Fast Tracing of Acoustic Beams and Paths Through Visibility Lookup

Fabio Antonacci, Marco Foco, Augusto Sarti, *Member, IEEE*, and Stefano Tubaro, *Member, IEEE*

***Abstract*—The beam tracing method can be used for the fast tracing of a large number of acoustic paths through a direct lookup of a special tree-like data structure (beam tree) that describes the iterated visibility information from one specific position. This structure describes the branching of bundles of rays (beams) as they encounter reflectors in their paths. For this reason, beam tracing is suitable for real-time acoustic rendering even when the receiver is moving. In this paper, we propose a novel technique that enables the fast tracing of a large number of acoustic beams through the iterative lookup of a special data structure that describes the global visibility between reflectors. The method enables the immediate generation of the beam tree corresponding to an arbitrary source location, which can then be used for path tracing through direct lookup. In practice, this technique generalizes the traditional beam-tracing method as it makes it suitable for real-time acoustic rendering not just when the receiver is moving but also when the source is moving. The method enables real-time modeling of acoustic propagation and real-time auralization in complex 2-D and 2-D$\times$1-D environments (e.g., vertical walls limited by horizontal floor and ceiling), which makes it suitable for applications of real-time virtual acoustics, immersive gaming, and advanced acoustic rendering. Some experimental results show the effectiveness of fast beam tracing with respect to the state of the art in acoustic beam tracing.**

***Index Terms*—Acoustic applications, acoustic beams, acoustic reflection, reverberation.**

I. Introduction

THE MOST widespread methods for the modeling of early acoustic reflections in complex environments are based on the geometric tracing of acoustic paths [1]. Starting from the spatial distribution and the acoustic properties of the reflectors and from the location and the radiation characteristics of source and receiver (i.e., microphone), this method aims at reconstructing the geometry of the acoustic paths as they propagate through the environment. From this information, it is possible to visualize the spatial distribution of the paths, estimate the sequence of reflections over time, and predict the spatial distribution of the acoustic pressure with a good degree of accuracy.

The literature is rich with techniques for efficiently tracing paths, many of which share a certain degree of similarity with ray tracing techniques developed for image rendering [2]–[4]. We recall, for example, the image source method [5]–[7] and radiosity [8]–[10]. Among the geometric solutions, however, particularly efficient is that of beam tracing [11]–[15]. This method implements the tracing of acoustic paths as a lookup process of a specific data structure that describes the branching of acoustic bundles of rays as reflections from walls take place. The beam-tracing approach, in fact, enables a real-time rendering of sounds in complex environments even when the listening points (receivers) are moving. Beam tracing was originally conceived by Hanrahan and Heckbert [11] for applications of image rendering, and then extended by Funkhouser *et al.* [16] to the problem of audio rendering. In this method, all reflectors are assumed to be piecewise planar, and all the rays originated from a source which hit the same planar region of the reflector are bundled up into beams. Every time a beam encounters a reflector, the portion of beam that illuminates[1] that reflector splits into a set of subbeams, each corresponding to a different planar portion of the encountered reflector. As they bounce around in the environment, beams keep branching out and attenuating until they die out. The beam-tracing method organizes and encodes this beam splitting/branching process into a specialized data structure called *beam tree*. The construction of the beam tree is based on an iterative visibility evaluation process, usually based on spatial subdivision. Once the beam tree is available, the path tracing becomes a very efficient process. In fact, given the location of the listening point (receiver), we can immediately determine which beams illuminate it through a lookup of the beam tree data structure. This enables a real-time auralization of the acoustic source even while the receiver is moving. However, moving the source requires the construction of a new beam tree, which is generally a computationally demanding task. As a consequence, the computational effort of the beam-tracing method turns out to be unevenly divided between the beam tracing and the path tracing phase [17].

What makes the beam-tracing method particularly interesting is the fact that paths are determined by looking up a data structure that describes in an organized fashion all possible paths that depart from a given source location. It is thus quite natural to wonder whether a similar method can be devised for the construction of a beam tree. This question can be more precisely formulated as follows: *is it possible to construct the beam tree corresponding to a specific source location by looking up a higher order data structure that describes the beam trees that depart from all possible source locations?*

We recall that the beam tree can be seen as a data structure that describes the regions that are *visible* (either directly or after reflections) from a given source location, as a function of the

Manuscript received July 17, 2006; revised January 24, 2008. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Vesa Valimaki.

The authors are with the Dipartimento di Elettronica e Informazione, Politecnico di Milano, 20133 Milan, Italy (e-mail: augusto.sarti@polimi.it).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASL.2008.920064

[1]Given the similarities between ray tracing in image rendering applications and path tracing in acoustic modeling applications, throughout this paper we will often use terms like *visibility*, *line of sight*, *illuminated* region, etc.

receiver position. What we now need is a data structure that describes the regions that are visible from a given source location, as a function not just of the viewing angles but also of the source location itself. This information, in principle, can be precomputed using the environment's geometry only.

Traditional solutions to the visibility evaluation problem address specific cases of visibility of a region from a point (regional visibility, useful in beam tracing), or visibility along a line (used in ray tracing) [18]. In the past few years, however, some works have appeared in the literature which deal with the evaluation of global visibility, i.e., the visibility of a region from a region (a rather complete interdisciplinary survey on the matter can be found in [19] and [20]). The literature proposes several effective representations that primarily differ in the way source and receiver locations are parameterized to describe visibility. In this parameter space (ray space or line space), a ray departing from a generic source is always represented as a point, and the set of rays (each identified by a source location and a listening direction) that hit a given reflector are represented by a (visibility) region.

In this paper, we will encode and organize the global visibility information in such a way to enable an immediate construction of a beam tree through an iterative visibility lookup process [21]. This will allow us to generalize the beam tracing method in such a way to model not just moving receivers but also moving sources in real-time. An in-embryo version of this approach was already presented in [21]. In this paper, this approach is generalized, as all geometric primitives involved in its development (rays, beams, reflectors, sources and receivers) are more correctly defined and described as oriented entities. This will have an impact not just on the symmetry and the elegance of the approach, but also on the implementation, which will turn out to be easier to handle. We also extend the validity of the 2-D beam tracing approach described in [21] to other types of environments, such as the Cartesian product of a 2-D and a 1-D environment (e.g., the 2-D×1-D environments described by vertical walls ending in perpendicular floor and ceiling).

The 2-D×1-D case is very significant in sound auralization applications because it well models propagation in complex building interiors, in urban outdoor environments, etc.

In Section II, we will define the visibility problems and revisit the tracing algorithms (ray and beam tracing) in accordance with the corresponding visibility definition. We will then define the key concept of global visibility.

In Section III the relationship existing between the global visibility and visibility from a point is shown. In particular, Section III illustrates that the global visibility can be profitably replaced with a collection of visibilities from the reflectors. The rest of Section III shows that the beam tree can be constructed through a recursive lookup on the global visibility data structure, which is equivalent to building a visibility tree.

Section IV provides an assessment of the achievable speedup obtained with the proposed approach. Section V proposes a simple example of implementation of the fast beam-tracing algorithm for applications of fast acoustic rendering. Conclusions and future developments are provided in Section VI.

II. Visibility and the Tracing Problem

As we are particularly interested in the characterization of beam tracing as a problem of visibility from a point in space, a more detailed discussion on what a beam is and what it implies is in order.

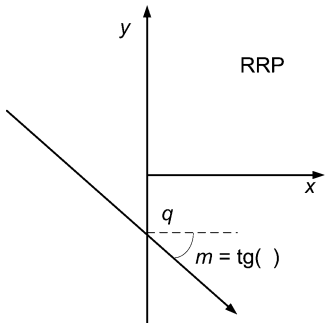


Fig. 1. RRP: The location of the intersection with the reference reflector and the angular coefficient of the ray at the intersection point are used as ray parameters.

The first step is to find a mapping from the geometric domain to the ray space that works as a minimal parametrization for rays. In general, a ray in a 2-D geometric domain can be readily specified by three parameters (two for the coordinates of the origin and one for the direction of the ray). This, however, turns out to be a redundant parametrization, as the rays that we are interested in are always bound to pass through some reflector (reference reflector); therefore, they can be more efficiently described by a pair of parameters. For example, we can adopt a parametrization based on the location and the angular coefficient of the ray at the point of intersection with the reflector. This parametrization is here referred to as the reference reflector parametrization (RRP), and is shown in Fig. 1.

Here, for the sake of simplicity and with no loss of generality, the frame is chosen so that the reflector will lie on the y axis. This way, a generic ray is defined by the equation $y = mx + q$ where q is the point of intersection between the ray and y axis, and $m = \tan\theta$ is the ray's angular coefficient, θ being the angle between the ray and the x axis.

Although the RRP is referred to a frame that is attached to a specific reflector, we will see that this choice does not represent a limitation for visibility evaluation purposes because of the iterative nature of the process itself.

A beam is a compact bundle of acoustic rays that originate from the same point (e.g., a source) and fall onto the same reflector (see [12]). It is completely defined by an origin and the (connected) illuminated region of an arbitrary cutting surface (typically the reflector of incidence). A beam can be visualized in ray space as the set of parameters corresponding to all of its rays. The region of the ray space corresponding to a beam is generally a portion of a 1-D curve. If we adopt the RRP, this region becomes a region of a line, as shown in Fig. 2.

Beams can be profitably arranged in a data structure called a beam tree. At the root of the tree is the source, while the children of node i refer to those subbeams that branch out from the reflection of beam i with a reflector in the environment. An example of a beam tree is shown in Fig. 3(a) and (b).

It is quite apparent from the above discussion that beams can be cast only with the knowledge of the source location and the environment's geometry. When the receiver is specified as well, the paths linking source and receiver can be determined. Every time a beam falls onto the receiver, we have a path

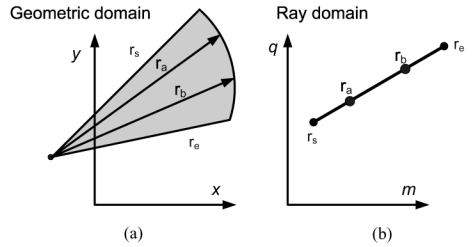


Fig. 2. A beam is completely characterized by its origin and the set of directions of its rays: r_s and r_e are the start and the end rays of the beam, while r_a and r_b are two rays inside the beam (a). Using the RRP, the parameters of all rays of the beam lie on a portion of a line in ray space. The points corresponding to rays r_s, r_e, r_a and r_b are shown.

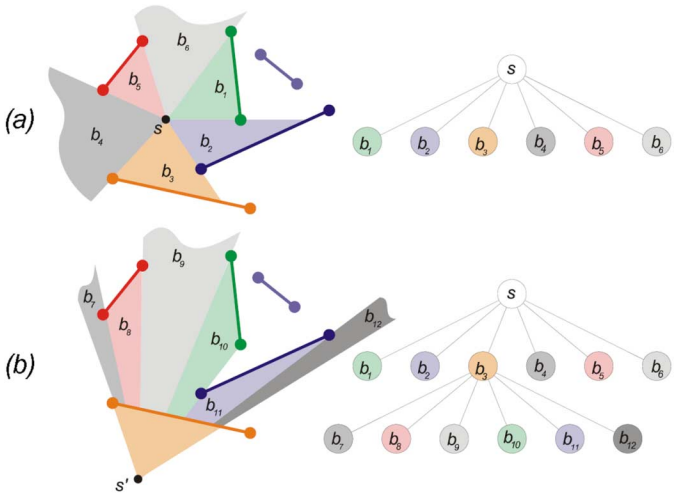


Fig. 3. Recursive beam tracing and beam tree building. Step (a) Beams are emitted from the source location. Some of them fall onto walls and others meet a reflector at infinity. The corresponding beam trees are shown on the right-hand side. At the root of the beam tree is the source. Step (b) Reflection of beam b_3. In the beam tree this corresponds to generating the children of node b_3.

linking source and receiver. The path can be readily determined using the sole information stored in the beam tree. As already explained in Section 1, however, there is a significant gap between the computational effort required for tracing beams and that for finding paths as the former needs an evaluation of visibility while the latter only needs a beam-tree lookup (see [17] and [16]). Different solutions have been carried out to overcome this problem. A very popular one is represented by the binary space partitioning technique [22], [23]. This particular approach was used not just for beam tracing but also for visibility tests in image-source methods [24].

In order to turn beam tracing into a lookup process, we need to have a closer look at the global visibility evaluation problem and, in particular, we need to characterize the process of beam splitting at a reflector.

At any time during the iterative beam-tracing process, a beam is always defined by an origin and by the region of the reflector that is illuminated by that beam. It is thus quite natural to refer to the reflector that defines the current beam of the iterative beam-tracing process as the *current* or *reference* reflector and adopt the RRP as the reference frame. In fact, the current beam (originated from a previous reflection) travels from the (virtual) source *through* the reference reflector and eventually hits other reflectors. These new reflectors decide how the beam will split. Our goal is to pre-evaluate the visibility of the next reflectors from the reference reflector in an offline phase, and just look it up as we trace beams in space.

We would be tempted to pose the problem in the following terms: *which portions of which reflectors are visible from an arbitrary point in space* through *the current reflector?*

This way of formulating the problem, however, results in having to construct visibility functions that, in the 2-D tracing case, depend on three parameters (two source location coordinates and one angle of view). Fortunately, having chosen the RRP our ray space turns out to be 2-D. This fact can be intuitively explained by the fact that visibility does not depend on where along the line of sight the source is. Similar ideas were exploited in [25] and [26] to reduce a 5-D plenoptic function to a 4-D visibility function (lumigraph).

In order to derive the iterative visibility lookup process, it is important to define the concept of "oriented" reflector as a planar region in space that reflects only on one side. In the 2-D case, this can be seen as a segment, only one side of which is reflective. Let us consider two planar oriented reflectors s_i and s_j. The orientation of such reflectors is specified by unit vectors $\mathbf{n}_i$ and $\mathbf{n}_j$, respectively, which are perpendicular to the segment, and pointing outward from the reflective face. We will say that a ray oriented like the unit vector $\mathbf{n}$ illuminates the reflector s_j when it passes through it and $\mathbf{n} \cdot \mathbf{n}_j < 0$ (incidence condition). A similar definition can be given for the incidence condition of a beam: we will say that a beam illuminates the reflector s_j when all of its rays pass through it and all of the rays $\mathbf{n}(m,q)$ of the bundle satisfy the incidence condition $\mathbf{n}(m,q) \cdot \mathbf{n}_j < 0$, where m and q are the angular coefficient and the location of the ray at the reference reflector (see definition of RRP, Fig. 1). It is not difficult to realize that if the incidence condition is satisfied by one of the rays of the beam, then it is satisfied by all the rays of the beam.

Slightly more general is the case of incidence condition of an arbitrary beam passing through the reference reflector s_i and the reflector s_j. In this case, we can check the incidence condition by checking whether s_i and s_j *face* each other. This can be readily verified by checking whether $\mathbf{n}_i \cdot \mathbf{n}_j < 0$, as long as the lines where the two reflectors lie do not cross on any of the two reflectors.

In order to define the visibility region *of* s_j *from* s_i, we first define the *visibility of* s_j and the *visibility from* s_i, and then we combine them together through region intersection.

In accordance to the definition of the RRP, our reference frame is defined so that the x axis will be aligned with the unit vector $\mathbf{n}_i$, and the reflector s_i lies on the y axis. We can generally assume that the ray's orientation $\mathbf{n}$ is always in agreement with the orientation of the reference reflector s_i, i.e., $\mathbf{n} \cdot \mathbf{n}_i > 0$ (ray's admissibility condition). This is a consequence of the fact that we are considering the visibility *from behind* the reference reflector as, during the iterative tracing of beams, the source is always a virtual one (see end of Section II).

If the reference reflector were of infinite extension, the region of visibility from it would be the whole ray space

$$\mathcal{R} = \{(m,q) : -\infty < m < \infty, \ -\infty < q < \infty\}.$$

Our reference reflector s_i, however, is of finite extension $q_1^{(i)} < q < q_2^{(i)}$, therefore the region of visibility from that reflector is

$$\mathcal{R}_i = \left\{(m,q):\ -\infty < m < \infty,\ q_1^{(i)} < q < q_2^{(i)}\right\}.$$

The region of visibility of the reflector s_j with respect to the reference frame specified before is given by the coordinates (m,q) corresponding to all possible rays $\mathbf{n}(m,q)$ that intersect the reference line $x=0$ and illuminate s_j

$$\mathcal{R}^j = \{(m,q):\ \mathbf{n}(m,q)\cdot\mathbf{n}_i > 0,\ \mathbf{n}(m,q)\cdot\mathbf{n}_j < 0\}.$$

We will show later that this region is generally shaped like a wedge in ray space, and that for a double-faced reflector we end up with a region shaped like a double wedge (see Fig. 5).

The region of visibility of reflector s_j from reflector s_i is given by the set of rays passing through s_j and s_i and belonging to

$$\mathcal{R}_i^j = \Big\{(m,q) : q_1^{(i)} < q < q_2^{(i)},\ \mathbf{n}(m,q)\cdot\mathbf{n}_i > 0, \mathbf{n}(m,q)\cdot\mathbf{n}_j < 0\Big\}.$$

The set $\mathcal{R}_i^j$ corresponds, quite clearly, to the intersection of the two regions

$$\mathcal{R}_i^j = \mathcal{R}_i \cap \mathcal{R}^j.$$

The evaluation of the global visibility consists of computing all regions of visibility of any oriented reflector from any oriented reference reflector (the two sides of a reflector are treated like two separate reflectors).

In the next section we will show how global visibility can be used for an iterative on-the-fly construction of the beam tree. In particular, we will see that a source can also be represented in ray space, as the set of all possible rays cast from the source location. This set corresponds to a line in ray space. The visibility of a reflector from the source is thus the result of the intersection between the line that represents the source in ray space, and the visibility region of the reflector. This means that the visibility of a reflector from a point can be defined as the result of a lookup process on the global visibility.

III. Visibility

The goal of this section is to describe the iterative process for constructing a beam tree starting from the global visibility and the source location. We will first specialize our ray space through a particular choice of RRP that facilitates the tracing of beams. For reasons that should already be quite apparent (see Section II) we will refer to this specific choice of ray space as the dual (ray) space.

A. Specialized RRP

Let s_i be the reference reflector from which we want to compute the visibility (in a 2-D space). For reasons of efficiency that will become clearer later on, we choose a reference frame $(x^{(i)}, y^{(i)})$ that normalizes the reflector s_i through a rotation, a translation, and a scaling. We do so in such a way that the normalized reflector will lie on the $y^{(i)}$ axis between the coordinates -1 and 1. The first step is to perform a translation of the coordinate reference system so that the origin will coincide with the reflector's center. Let $(\bar{x}^{(i)}, \bar{y}^{(i)})$ and $(-\bar{x}^{(i)}, -\bar{y}^{(i)})$ be the endpoints of the reference reflector after the translation, and let α be the angle between the vector $(\bar{x}^{(i)}, \bar{y}^{(i)})$ and the x axis. The normalization of the geometric domain can be performed as

$$\begin{bmatrix} x^{(i)} \\ y^{(i)} \end{bmatrix} = \frac{1}{|\bar{y}^{(i)}|} \begin{bmatrix} \cos(\alpha \pm \pi/2) & \sin(\alpha \pm \pi/2) \\ -\sin(\alpha \pm \pi/2) & \cos(\alpha \pm \pi/2) \end{bmatrix} \begin{bmatrix} \bar{x}^{(i)} \\ \bar{y}^{(i)} \end{bmatrix} \tag{1}$$

where

$$\alpha = \operatorname{atan}\left(\frac{\bar{y}^{(i)}}{\bar{x}^{(i)}}\right).$$

Choosing $\alpha+\pi/2$ or $\alpha-\pi/2$, we select one of the two rotations that bring the segment on the $y^{(i)}$ axis. The reference reflector divides the space $(x^{(i)}, y^{(i)})$ into two halfspaces: $x^{(i)} > 0$ and $x^{(i)} < 0$.

The set of rays passing through s_i is described by the equation $y^{(i)} = mx^{(i)} + q$ where $-1 \le q \le 1$ and $-\infty < m < \infty$. Fig. 4 shows the reflector s_i referred to the normalized frame in the geometric domain (left). The corresponding region of visibility from the reference reflector is the horizontal strip (reference visibility strip) in the dual space (right).

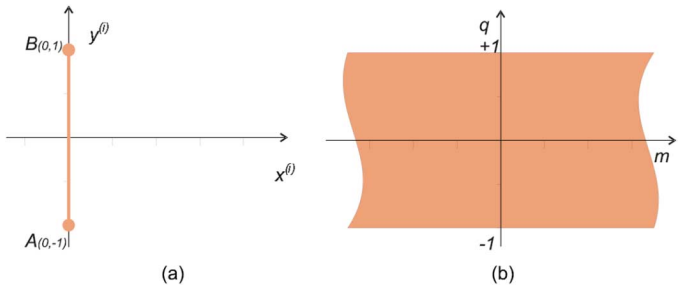


Fig. 4. (a) Specialized RRP and (b) the set of rays passing through the reference reflector in the (m,q) domain (visibility region from the reference reflector).

It is important to notice that, according to the image source principle, we should not consider occlusions on the part of reflectors that lie in the same halfspace of the virtual source. For example, we could have an oriented reflector placed between the virtual source and the reference reflector, facing the source. Although this configuration does not violate either the ray's admissibility condition or the incidence condition (see Section II), this is clearly a reflector to be discarded. As a convention, in the future we will use the rotation which brings the image source in the halfspace $x^{(i)} < 0$.

B. Geometric Primitives in Ray Space

As already seen above, the ray space turns out to play a similar role as the dual of a geometric space. The duality between geometric primitives, and the corresponding primitives in ray space, in fact, exhibits numerous symmetries that will become apparent in the following subsections.

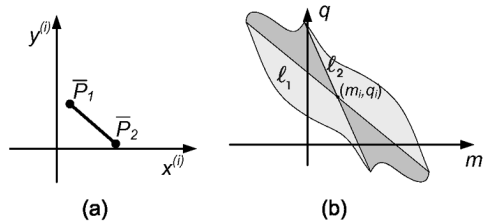


Fig. 5. Dual of a segment: the dual of the segment in (a) are the shaded regions of the ray space in (b).

1) Dual of a Point: Let us consider a point $\bar{P}^{(i)} = (\bar{x}^{(i)}, \bar{y}^{(i)})$ in the normalized geometric domain that is referred to reflector i. The lines passing through $\bar{P}^{(i)}$ are represented by the equation

$$q = \bar{y}^{(i)} - m\bar{x}^{(i)}. \tag{2}$$

The set of points that satisfy (2) form a line in the dual ray space.

2) Dual of a Reflector: A two-sided reflector (segment) is completely specified in the geometric domain if we fix its endpoints $P_1^{(i)}(x_1^{(i)}, y_1^{(i)})$ and $P_2^{(i)}(x_2^{(i)}, y_2^{(i)})$. The dual of such points in the ray space are the lines $l_1 : q = -mx_1^{(i)} + y_1^{(i)}$ and $l_2 : q = -mx_2^{(i)} + y_2^{(i)}$. The intersection point of the lines l_1 and l_2 in the ray space is determined as in (3):

$$m_i = \frac{y_2^{(i)} - y_1^{(i)}}{x_2^{(i)} - x_1^{(i)}}, \quad q_i = -\frac{y_2^{(i)}x_1^{(i)} - y_1^{(i)}x_2^{(i)}}{x_2^{(i)} - x_1^{(i)}}. \tag{3}$$

The dual of all intermediate points between P_1 and P_2 is represented by the set of all lines whose slopes are between those of l_1 and l_2. This set of lines can be seen as the dual of a nonoriented reflector (see Fig. 5).

When $x_1^{(i)} = x_2^{(i)}$ (i.e., the reflector is parallel to the reference one) the (m, q) coordinates of the intersection point go to infinity, which means that the lines l_1 and l_2 are parallel.

3) Dual of an Oriented Reflector: The dual of an oriented (one-sided) reflector is the set of rays that fall onto just one of the two faces of the reflector. In order to obtain this region of visibility, we follow a simple geometric procedure to assess which face of the reflector is hit by a generic ray. Each reflector is characterized by the vectors $\mathbf{v}_1^{(i)}$ and $\mathbf{v}_2^{(i)}$ which go from $P_2^{(i)}$ to $P_1^{(i)}$ and vice versa, respectively. A counterclockwise rotation of the vectors $\mathbf{v}_1^{(i)}$ and $\mathbf{v}_2^{(i)}$ of $\pi/2$ radians returns the new pair of vectors $\mathbf{v}_{p_1}^{(i)}$ and $\mathbf{v}_{p_2}^{(i)}$, each pointing to one of the two halfspaces singled out by the segment $\overline{P_2^{(i)}P_2^{(i)}}$.

In the next steps of the beam-tracing algorithm, using the incidence condition described in Section II, we determine which one of the two faces of the segment is hit by the ray $\mathbf{r}_d$. The conditions to test are

$$\mathbf{v}_{p_1}^{(i)} \cdot \mathbf{r}_d < 0, \quad \mathbf{v}_{p_2}^{(i)} \cdot \mathbf{r}_d < 0.$$

The correct side of the reflector is the one that satisfies the above condition (see Fig. 6).

In order to show how to derive the dual of the oriented reflector $P_1^{(i)}P_2^{(i)}$ with orientation $\mathbf{v}_{p_1}$, let us consider the configuration of the left-hand side of Fig. 7. The ray $\mathbf{r}_b$ hits the reflector on the face identified by $\mathbf{v}_{p_1}$, unlike $\mathbf{r}_a$, which hits it

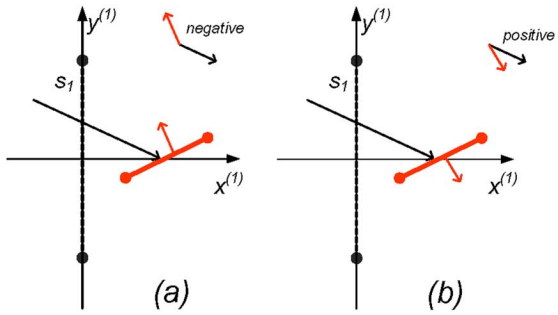


Fig. 6. Test on the ray direction to determine which face of the reflector is hit by the ray.

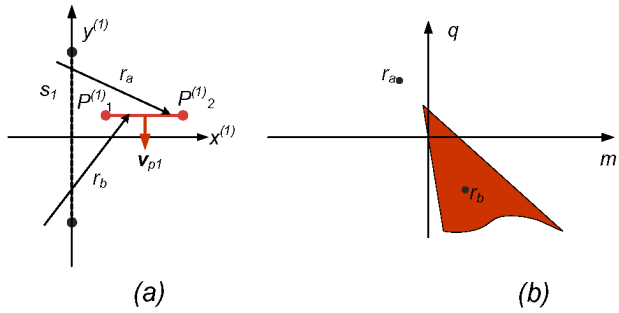


Fig. 7. Dual of an oriented reflector: the ray r_a is not included in the dual of the oriented reflector because it hits the reflector on the wrong side.

on the other side. The dual of the reflector $\overline{P_1^{(i)}P_2^{(i)}}$ with orientation $\mathbf{v}_{p_1}$ is shown on the right-hand side of Fig. 7.

Indeed, the dual of a two-sided reflector is the union of the duals of the one-sided reflectors corresponding to its two faces (each wedge-shaped).

4) Visibility Region: All rays originated from the reference reflector s_i form the region of visibility from that reflector. After normalization, this region takes on the strip-like shape described in Fig. 4. The rays originating from the reference reflector that hit a limiting reflector s_j, however, will only form a subset of this strip (see Fig. 4), as it is the intersection between the dual of s_j and the dual of s_i (reference visibility strip). As shown in Fig. 8, according to the geometric configuration of reflectors, different situations can be encountered in the (m, q) domain.

The intersection of the dual of s_j and the visibility strip is called visibility region of s_j from s_i. Once the source location is specified, the set of rays passing through s_j and s_i and departing from that location will constitute a subset of the visibility region of s_j. This is, in fact, a crucial point of our procedure because, as we will point out later on, this subset is, specifically, a (1-D) segment. This means that determining visibility from a point (beam) corresponds to a local linear scanning of a visibility region. The main advantage of the visibility approach to the beam tracing problem resides in the fact that we only need geometric information about the environment to compute the visibility regions. For this reason, the visibility information can be computed in advance, while the linear scanning is the only operation done on the fly.

5) Dual of Multiple Reflectors: Visibility Maps: When there are more than two reflectors in the environment, we must consider the possibility of mutual occlusions. When that happens,

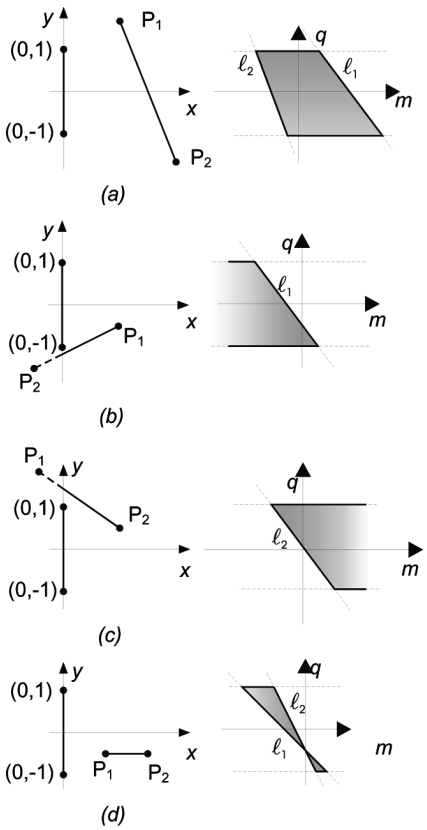


Fig. 8. Different shapes of visibility regions for beam tracing. (a) Refers to a segment which is completely visible from the reference reflector, (b) and (c) refer to segments which are not completely visible from the reference reflector. Reflector in (d) has both faces visible from different portions of the reference reflector.

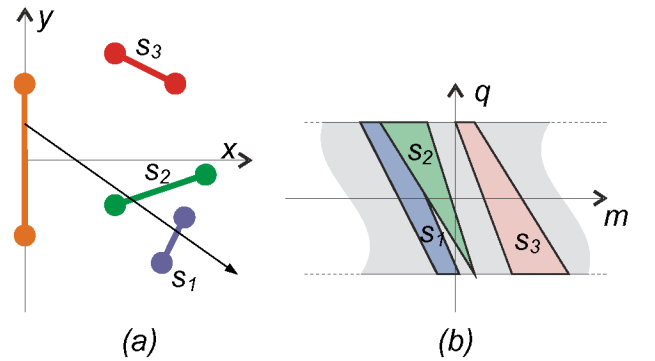


Fig. 9. Example of occlusion between reflectors and corresponding overlap of the visibility regions. (a) For some rays, the reflector s_1 is occluded by s_2. (b) The parameters of all occluded rays form a region of overlap between visibility regions in ray space. In order to determine which region overlaps which, it is sufficient to cast a test ray.

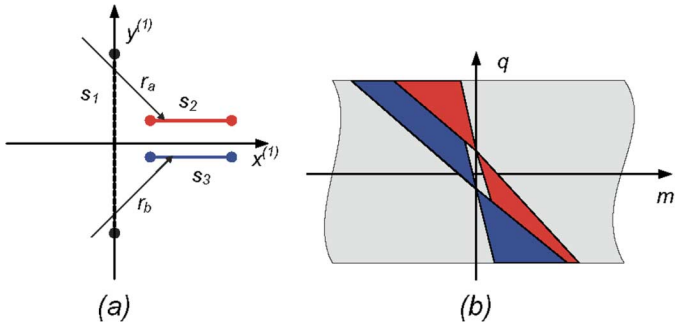


Fig. 10. Ambiguity in the occlusion between two nonoriented reflectors s_2 and s_3. For some rays (e.g., r_a) s_2 occludes s_3. For other rays (e.g., r_b) s_3 occludes s_2.

the relative visibility regions end up overlapping. Sorting out which reflector occludes which corresponds to determining which visibility region overrides which in their overlap. An example of this sort is shown in Fig. 9, where the two reflectors s_1 and s_2 are in a partially occluding configuration with respect to the reference reflector. In order to determine which visibility region overrides which in the overlap, we can simply shoot a test ray (whose coordinates in ray space (m, q) are picked within the overlap), find the order in which reflectors are hit by the ray, and sort the visibility regions in front-to-back order. The resulting collection of visibility regions constitutes the visibility diagram of the reference reflector: for each (m, q) pair, the visibility diagram returns the index of the first reflector hit by the (m, q) ray.

The collection of visibility maps of all the reflectors determines the global visibility.

A problem arises when the reflectors are in a configuration similar to that of Fig. 10. If s_1 is the reference reflector, we end up having an occlusion between s_2 and s_3, which needs to be solved. We immediately notice that, if we do not consider each reflector as a pair of oppositely facing oriented reflectors, the occlusion problem cannot be disambiguated: in fact for some rays s_2 occludes s_3, while for other rays s_3 occludes s_2. We know that a two-sided reflector corresponds to a double wedge in ray space. As each oriented side of the reflector corresponds to one of these two wedges, the ambiguity problem can now be overcome. The right-hand side of Fig. 10 shows the visibility regions of s_2 and s_3 sorted in front-to-back order.

6) Dual of a Beam: During the iterative tracing, the beam is completely specified by an origin (virtual source) and by its region of intersection with the reference reflector (which is the reflector's region that the reflected sound radiates from). However, starting from this information, we could define our beam as either a bundle of lines or as a bundle of oriented rays.

If the beam is defined as a bundle of lines passing through the origin and covering a preassigned angular region, i.e., a nonoriented beam (a double wedge), then its dual l is a segment in ray space, i.e., a subset of the line that represents the dual of the origin of the beam (see Fig. 11). If, on the other hand, the beam is defined as a bundle of oriented rays, i.e., an oriented beam (a single wedge), then its dual is an oriented (one-faced) segment. This was to be expected because there exists a complete duality relationship between geometric space and ray space, and we had already established that the dual of an oriented segment is an oriented beam (a single wedge).

The information of orientation (in geometric space or in ray space) is important whenever we need to check the incidence condition. For example, we know which one of the two wedges of a nonoriented beam crosses the reference reflector, but we need to check the incidence condition if we want to determine how the beam splits when it encounters the next reflectors. Notice that the incidence condition can be verified directly in

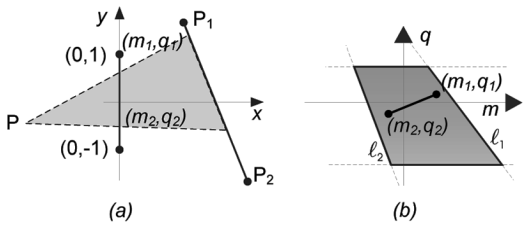


Fig. 11. Dual of a beam as the intersection of the reference strip, the visibility region of the limiting segment, and the dual l of the point $P(x, y)$. The lines (m_1, q_1) and (m_2, q_2) which limit the beam are transformed in the dual space into the two endpoints of the beam. Intermediate points in the dual space between (m_1, q_1) and (m_2, q_2) correspond in the geometry to intermediate lines between the end ones.

geometric space, in which case we will retain the orientation information of the beam; or in ray space, in which case we will retain the orientation information of the dual segment (see Section III-C).

The dual of a beam is a key concept for our tracing process because we intend to evaluate visibility in ray space. The fact that the dual of a beam is a segment, in fact, reduces the visibility evaluation problem to a 1-D lookup, which results in a tremendous speedup of the tracing process.

This can be readily seen, for example, when evaluating the splitting of a beam at a limiting reflector. We want to determine which of the rays that originate from $P(x^{(i)}, y^{(i)})$ and pass through the illuminated region of the reference reflector end up hitting a limiting reflector. We do so by scanning the intersection between reference strip and dual of the limiting reflector over the segment l.

The situation is summarized in Fig. 11: the segment $[q_1, q_2]$ limits the illuminated region of the reference reflector and the origin of the beam is in $P(x^{(i)}, y^{(i)})$ (left-hand side). The corresponding situation in the ray space is on the right-hand side: the lines (m_1, q_1) and (m_2, q_2) which limit the beam are transformed in the dual space into the two endpoints of the beam. Intermediate points in the dual space between (m_1, q_1) and (m_2, q_2) correspond in the geometry to intermediate lines between the end ones.

7) Summary of Dualities: The fact that ray space is seen as the dual of the geometric space means that a nonoriented ray (a line) maps onto a point. Similarly, an omnidirectional source/microphone (a point or, equivalently, a pencil of nonoriented rays) is represented by a (nonoriented) line. One different way to look at this duality is to consider a nonoriented reflector of infinite extension (again a line), which corresponds to a point (an omnidirectional beam). Similarly, an omnidirectional beam in ray space is represented by an infinite line (a two-sided reflector).

When dealing with limited extensions, things start getting more interesting. For example, the dual of a finite reflector (indeed a segment) is a beam and the dual of a beam is a segment, and there exists the same duality between oriented reflectors and oriented beams. This means that there is a distinction also between omnidirectional sources (bundles of outgoing oriented rays) and omnidirectional listening points (bundles of incoming oriented rays), which correspond to infinitely extended oriented lines. Of course this last duality works the other way around too.

TABLE I
PRIMITIVES IN THE GEOMETRIC DOMAIN AND THEIR CORRESPONDING REPRESENTATION IN RAY SPACE

Geometric space	Ray space
Omnidirectional bundle of non-oriented rays	Non-oriented ray or two-sided infinite reflector
Omnidirectional bundle of outgoing rays (source)	one-sided infinite reflector
Omnidirectional bundle of incoming rays (receiver)	one-sided infinite reflector
beam (double wedge)	two-sided reflector
two-sided reflector	beam (double wedge)
oriented beam (single wedge)	one-sided reflector
one-sided reflector	oriented beam (single wedge)
(oriented) beam splitting	(one-sided) reflector fragmentation
incidence relation	incidence relation
intersection (illumination condition)	intersection (illumination condition)

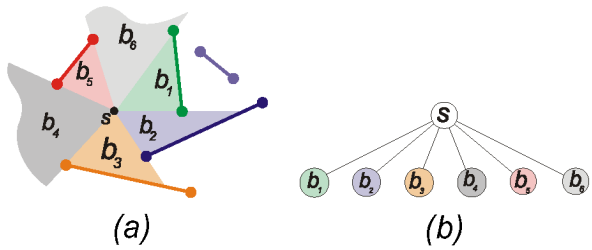


Fig. 12. Beams traced from (a) the source location and (b) the corresponding beam tree.

The above duality is not limited to just geometric primitives, but it also extends to geometric relations. For example, the incidence relation between an oriented beam and an oriented reflector corresponds to the same incidence relation between the dual of the beam (reflector) and the dual of the reflector (beam).

Table I summarizes some geometric primitives and relations and their representation in the (m, q) ray space.

C. Visibility From a Source in Ray Space

Now we have all the information that it takes to explain the iterative construction of a beam tree. We will do so using an example.

Consider the configuration of reflectors shown in Fig. 12(a). The first step of the algorithm consists of determining how the complete pencil[2] of rays produced by the source is partitioned into beams. This is done by evaluating the visibility from the source using traditional beam tracing. Notice that we could also evaluate this visibility directly in the ray space, but this would entail the definition of a reference frame that is not attached to any reference reflector (very first iteration), with consequent complications in the incidence conditions.

This initial splitting process produces two classes of beams: those that are limited by a reflector and those that are not. The beams and the corresponding beam tree are shown in Fig. 12(a) and (b), respectively. Beams b_1, b_2, b_3, and b_5 are limited by a reflector, while beams b_4 and b_6 continue their propagation towards infinity; therefore, they are no longer considered in the next iterations.

[2]A *pencil* is the set of all rays that are bound to pass through a preassigned point.

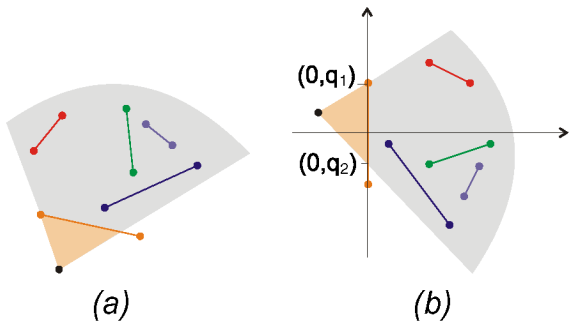


Fig. 13. Beams traced from (a) the source position and (b) the corresponding normalization.

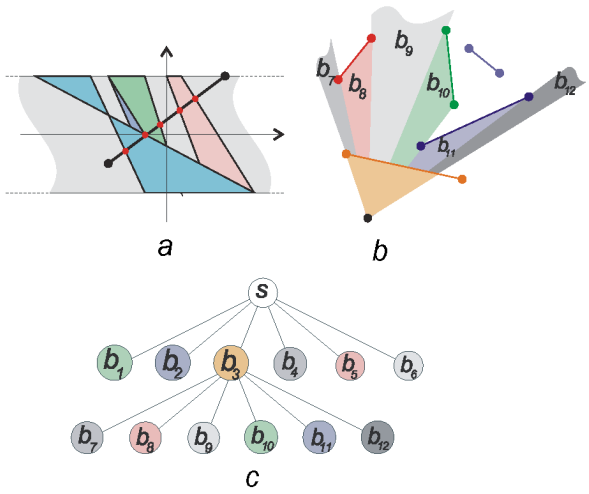


Fig. 14. (a) Beam subdivision for the set of rays in Fig. 13. (b) Traced beams in the unnormalized geometric domain. (c) Beam tree data structure.

Let us consider the splitting process applied, for example, to beam b_3 (similar considerations will apply to the other beams). As a first step, we determine the image source of s by mirroring the source about the reference reflector s_3. In order to use the visibility diagram of the reference reflector, we need to normalize the source location. The two steps (generation of the image source and normalization) are shown in Fig. 13(a) and (b), respectively. As explained in Section III-B4, the generation of the beam corresponds to the intersection of the visibility diagram with the line l dual of the image source position. The line l is made of a number of segments, each corresponding to a portion of the line l lying on a visibility region of the visibility diagram. The beam-tracing subdivision of the example in Fig. 13(b) is illustrated in Fig. 14. In (a), the beam subdivision is accomplished in the (m, q) domain, while in (b) the beams corresponding to the segments in (m, q) domain are traced. In (c), the resulting beam tree structure is illustrated. Beams b_7, b_9, and b_{12} proceed to infinity and correspond to unlabeled segments in the visibility diagram, while beams b_6, b_8, and b_{12} are blocked by reflectors.

A number of parametrizations of beams can be devised. The vectors are oriented in order to guarantee that a point inside the beam is always on the right of these vectors, as shown in Fig. 15.

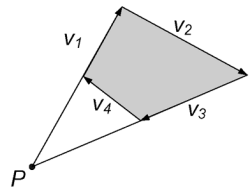


Fig. 15. Beam parametrization by memorizing the vectors containing the region contained in the beam.

The proposed parametrization has the advantage of speeding up the test that checks the presence of the receiver in the traced beams.

D. Beam Tracing as an Iterative Lookup

The iterative procedure detailed in the previous section is carried out for all the beams limited by a reflector. To summarize, we start with defining an image source for each limiting reflector of the previous iteration. For each one of these reflectors, we carry out the splitting process. This works by normalizing the geometric domain with respect to it, and by considering the intersection between the visibility diagram of the reflectors with the dual of the current image source (angular sector of the image source). This operation generates a number of subbeams. Some of them proceed to infinity, others are blocked by reflectors and therefore they originate a new branching. As new beams are born, the beam tree data structure gets updated. The recursive procedure stops when the beam tree reaches a preassigned order or reflection or when the beams die out (i.e., when they are attenuated below a preassigned threshold of magnitude).

E. Path Tracing as an Iterative Lookup

As explained before, path tracing is here intended as a point-to-point (source-to-receiver) visibility check. Once the receiver location is specified, a simple iterative procedure looks up the beam tree. Each node in the tree corresponds to a beam, and the presence of the receiver in the beam is checked through a simple geometrical test [16]. Different implementations of the test can be devised, according to the beam's parametrization. Using the parametrization in Fig. 15, the presence test consists only of verifying that the receiver is on the right-hand side of all the vectors $v_i, i = 1, \ldots, 4$ that parametrize the beam. This testing is carried out by computing the scalar products between the vector pointing to the receiver and each one of the vectors $v_i, i = 1, \ldots, 4$.

IV. Experimental Results

In this section, we discuss some implementation aspects of the aforementioned fast tracing algorithm.

This section is divided into three subsections. We will first consider how we can account for reflections from ceiling and floor under some assumptions on the environment's geometry. We will then provide the results of some test experiments, observing the computational time for the visibility diagram, the beam tree, and the filter update.

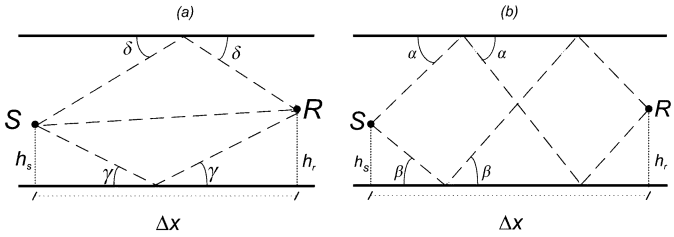


Fig. 16. Reflections on floor and ceiling when the number of reflection n is odd or even ($n = 1$ or $n = 2$).

A. Modeling Reflections in 2-D×1-D Environments

In many visibility computation problems, it is quite simple to extend to the 3-D case solutions that are originally conceived in 2-D (see for example [28]). The extension of 2-D beam tracing to a 3-D environment, however, is not a trivial problem. What prevents us from generalizing the results in that direction is the fact that concepts of duality cannot be directly extended from 2-D to 3-D. Similar considerations are made in [18] and [29].

A direct extension of the concepts described in the previous section is quite straightforward if we assume that the 3-D environment is a separate Cartesian product between a 2-D floormap and a 1-D vertical distribution of interfaces (separable environment). An example of this kind of environment is the generalized cylinder: floor and ceiling are parallel and perpendicular to walls. It is also easy, however, to accommodate transmissions between different floors in a building that preserves the same floormap, or to accommodate a ceiling-floor material distribution (reflectivity) that changes with the floormap location.

Separable environments have been already treated in literature; see for example [18] and [29]. The extension of the 2-D beam tracing to 2-D × 1-D environments passes through the generation of a number of image sources by the recursive reflection of the source on floors and ceilings. This "path multiplication" stage is repeated for each acoustic path. Each image source generated in this stage corresponds to a reflective path between source and receiver and bouncing on floors and ceilings. Let H be the height of walls, Δx be the distance between source and receiver along the horizontal path, and h_s and h_r be the height of the source and the receiver, respectively. In order to determine the angles between the ray and the ceiling and the ray and the floor, we need to distinguish between the cases when n is even or odd, n being the total number of times the ray bounces on floor and/or ceiling. The notation used to represent the angles formed by the rays and the floor and between the rays and the ceiling is described in Fig. 16 for the cases of n being odd $(n = 1)$ or even $(n = 2)$. Angles α to δ are provided as follows:

$$\alpha = \operatorname{atan}\left(\frac{H(n-1) + h_r - h_s}{\Delta x}\right) \quad n \text{ even} \tag{4}$$

$$\beta = \operatorname{atan}\left(\frac{H(n-1) + h_s - h_r}{\Delta x}\right) \quad n \text{ even} \tag{5}$$

$$\gamma = \operatorname{atan}\left(\frac{Hn - h_s - h_r}{\Delta x}\right) \quad n \text{ odd} \tag{6}$$

$$\delta = \operatorname{atan}\left(\frac{H(n-2) + h_s + h_r}{\Delta x}\right) \quad n \text{ odd}. \tag{7}$$

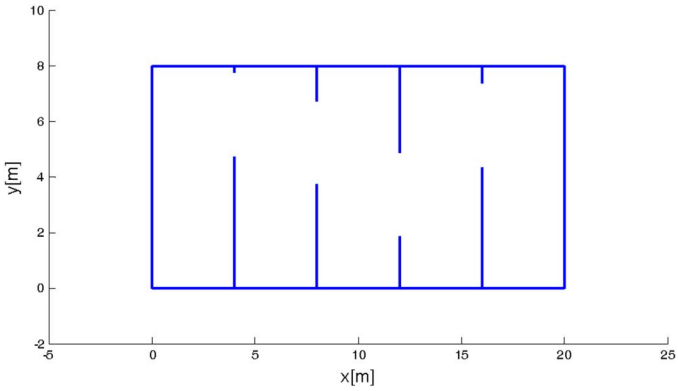


Fig. 17. Example of variable-geometry test environment. The 8 m×4 m rooms are all connected by randomly located apertures.

B. Experimental Results and Performance Evaluation

In order to evaluate the effectiveness of the proposed approach we tested the algorithm on several types of environments of controlled complexity.

One type of controlled-complexity environment is shown in Fig. 17. The environment is made from a variable number of 8 m×4 m rooms connected together by an access that is randomly placed on the 8-m side. An N-room will thus have $4N$ reflective walls. We tested the algorithm with 20, 40, 80, 120, 320, and 640 such walls. This environment's configuration will be later referred to as "variable geometry."

The second type of environment is similar to the previous one but the location of the apertures is now fixed at the center of the 8-m walls. This configuration will be later referred to as "fixed geometry." We tested the algorithm with fixed geometry, again, with 20, 40, 80, 120, 320, and 640 such walls.

It is important to underline that with these two types of modular environments only a small number of walls are, in fact, visible from a generic source location. In order to evaluate the effectiveness of the algorithm in situations of complete visibility, we also conducted a number of simulations on a convex polygonal environment like that of Fig. 18, with an increasing number of walls: 5, 10, 20, 30, 80, 160, 320, and 480.

The tests were conducted with an Intel Mobile Pentium processor equipped with 1 GB of RAM, in order to:

- measure the building time of visibility diagrams;
- compare the beam tree's building time of visibility-based beam tracing with that of traditional beam tracing based on [11];
- measure path tracing time;
- measure the number of traced paths.

1) Visibility Diagram Building Time: The building time of visibility diagrams is shown in Fig. 19 together with its cubic fitting curve for the variable-geometry environment.

The fact that the fitting function for the building time of visibility diagrams is cubic has a theoretical explanation in the determination of occlusions between reflectors. If the number of reflectors is N, while computing the visibility diagram of reflector i, the occlusion between all the reflectors different from i must be considered, and this results in a quadratic function of the number of reflectors, therefore, global visibility depends on the cubic power of the number of reflectors. The same fitting

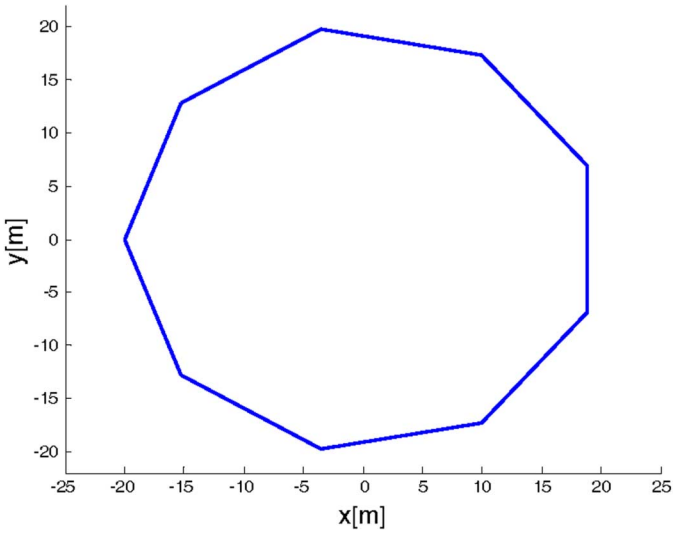


Fig. 18. Example of convex polygonal environment.

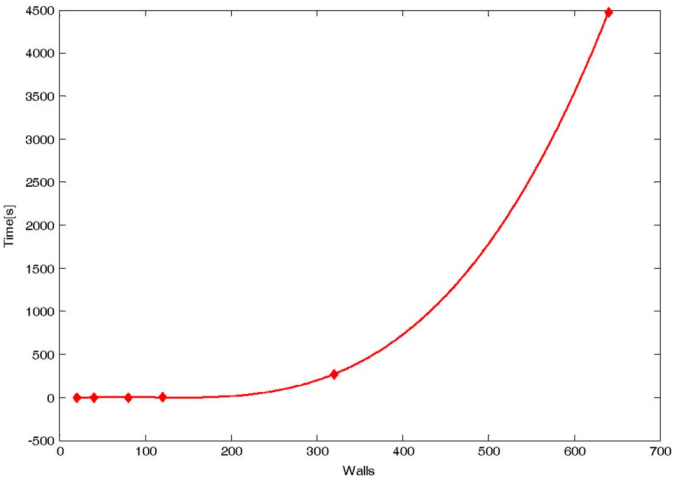


Fig. 19. Building time of visibility diagrams in the case of variable-geometry environments.

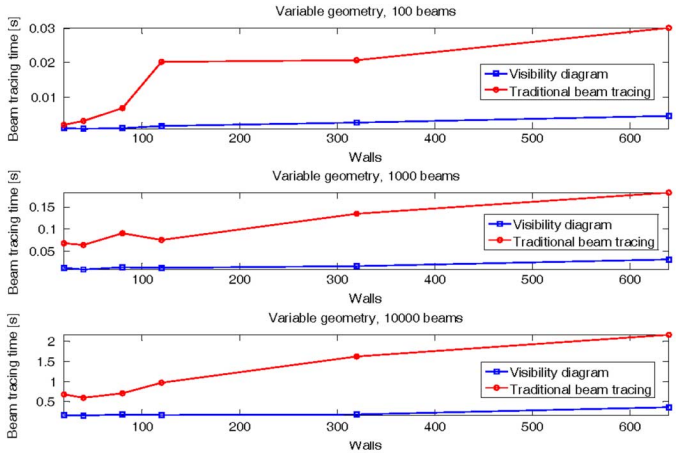


Fig. 20. Beam tree building time for variable geometry for traditional beam tracing (circles) and visibility-based beam tracing (squares) for 100 beams (top), 1000 beams (center), and 10 000 beams (bottom). The proposed approach greatly outperforms traditional beam tracing especially when the number of traced beams is very large.

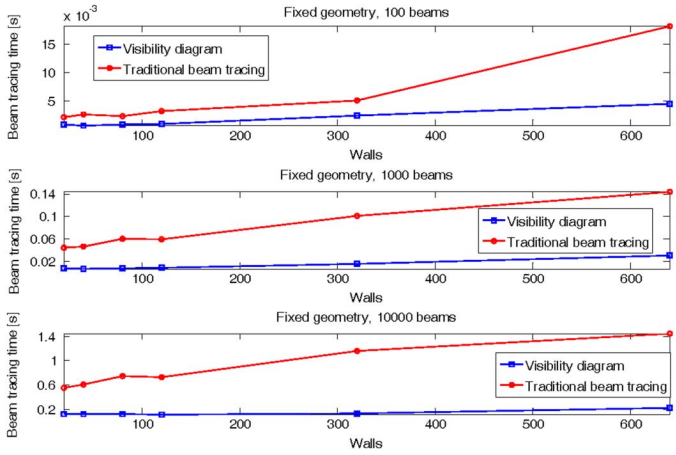


Fig. 21. Beam tree building time for fixed geometry for traditional beam tracing (circles) and visibility-based beam tracing (squares) for 100 beams (top), 1000 beams (center), and 10 000 beams (bottom). The same conclusions as in Fig. 20 can be drawn.

was repeated for fixed-geometry and convex-polygonal environments. The results turn out to be similar to Fig. 19, since the time spent in evaluating the visibility depends only on the number of reflectors and not on their configuration.

2) Beam Tree Building Time: The beam tree building time is intended as the time spent by the algorithm in tracing a preassigned number of beams. In order to assess the impact of the proposed approach on this specific parameter, in comparison with a traditional beam-tracing approach, we stopped the algorithm when 100, 1000, or 10 000 beams were traced. As expected, our approach turns out to outperform traditional beam tracing, particularly when the number of traced beams is very large. As the beam tree building time strongly depends on the source location, we conducted many tests by placing the source at the center of each one of the rooms of the modular environments. The beam tree building time was then computed as the average of the beam tree building times over all such simulations. The results of this experiment are shown in Figs. 20 and 21, which confirm that the visibility-based tracing greatly outperforms traditional beam-tracing methods.

Notice that the traditional beam tracing exhibits an irregular behavior in Fig. 20, for a limited number of walls, when 100 beams are traced. This anomaly is due to the irregular shape of the environment. In fact, the average order of the beam tree that is necessary for tracing 100 beams is higher for 80 reflectors than it is for 160 reflectors.

Beam tracing greatly benefits from visibility diagrams particularly when reflectors are in heavily occluded configurations. In this case, in fact, only a limited number of visibility regions turn out to show up in each visibility diagram. When, on the other hand, each reflector sees all the other reflectors, the benefit from using visibility diagrams is reduced. Fig. 22 shows the beam-tracing building time for 100, 1000, and 10 000 beams in the case of a convex polygonal environment. In this case, as expected, all visibility regions are present in the various visibility

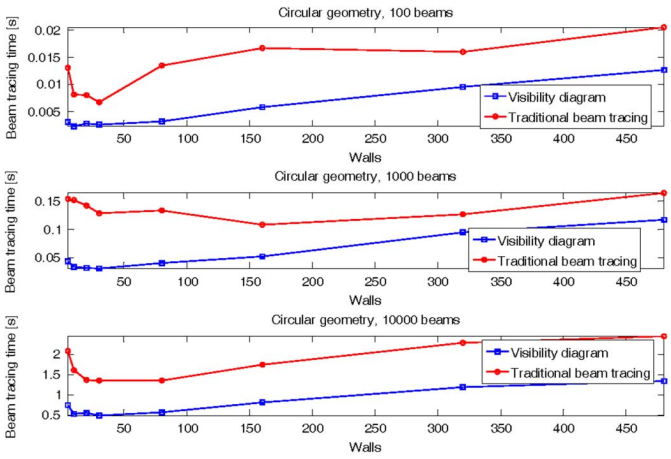


Fig. 22. Beam tree building time in the case of convex polygonal geometry for traditional beam tracing (circles) and visibility-based beam tracing (squares) for 100 beams (top), 1000 beams (center), and 10 000 beams (bottom).

diagrams with no overlaps; therefore, the advantages are here severely reduced.

In [16], a similar test was conducted. The approach presented in [16] is 3-D in nature. However, most of the environments in which their tests are conducted (pp. 749–750) comply with the 2.5-D assumption. For a comparable number of polygons, the beam tree tracing time turns out to be always longer than in our approach. As an example we can consider the "Room" environment in [16] characterized by 20 polygons: there, the computational time for tracing 1939 beams in approximately 180 ms. By Fig. 20, we can observe that for a variable-geometry environment of 20 polygons, the tracing time for 1000 beams is lower than 20 ms. When 30 000 beams are traced, the beam tree construction process presented in [16] needs approximately 3 s to update the data structure. When we consider a 20-wall variable-geometry environment, the time required to build the 10 000 beams in our approach is lower than 0.3 s. The above comparisons, however, are not exact, since the two algorithms are tested on different workstations and an implementation of [16] is not available.

Furthermore, in [16], we can observe that the computational time turns out to grow more than linearly as the number of walls increases.

3) Path Building Time: We also performed a set of experiments to confirm that the time needed for finding paths starting from a previously traced beam tree is unaffected by the choice of method used for determining the beam tree. As the number of paths depends on both source location and receiver location, in order to assess the general behavior of the path tracing step, we placed the source in the center of the modular environment and conducted several simulations with the receiver placed in each one of the rooms. This way, the average time obtained in these tests would be quite independent of the specific receiver location. We measured the path tracing time versus the number of reflectors when 10 000 beams are traced in a fixed-geometry environment. The simulation results confirmed that the differences in the path tracing times are negligible.

V. Example of Auralization System

In the previous section, we provided an analysis of the computational cost of the proposed solution and we showed that the advantages are quite significant, particularly from the computational standpoint. In order to test the possibility of updating the beam tree in real-time from the user's standpoint, we devised and developed a simple auralization experiment that enable the acoustic rendering of moving sources in environments with complex floormaps. The experiment's aim is not to produce realistic results (neither diffraction nor diffusion are here accounted for and some approximations are introduced), but only to check the adequacy of the responsiveness of the source motion interaction.

The interested reader can find more complete accounts on the state-of-the-art for auralization systems in [30], [31], and [24]. Our goal here is just a brief description of some critical stages involved for the auralization process.

The first step in the definition of our auralization process is the implementation of the head related transfer functions (HRTF). As the HRTF varies continuously with the direction of arrival (DOA) of the signal, each reflection should be separately filtered. From the computational standpoint, this operation turns out to be too demanding; therefore, we group together the DOAs into angular intervals and generate a separate audio stream for each interval. Each one of these angular intervals is attributed a separate tapped delay line [a finite-impulse response (FIR) filter] whose taps are computed considering all path lengths and the relative attenuations.

The filter bank whose parameters are generated at the previous step constitutes the auralization algorithm, which generates all the directional streams coming from an anechoic (dry) source. These streams are then mixed together to produce the stereo (headphone) auralization based on the HRTF. In our implementation, we used OpenAL [32] to mix 16 virtual "equivalent sources" placed in a circle around the listener (to simulate 16 discretized DOAs). The listener's head orientation is accounted for only in this last step. In order to account for HRTF in 3-D space, virtual equivalent sources should be placed in a sphere rather than a circle. However, a correct sampling of a sphere involves a huge number of points. As a consequence, reflections coming from floor and ceiling have been clamped to the azimuth plane, while their delays remain unaltered. This is, indeed, an approximation, which is, however, of less importance for the goals of this experiment.

Given the acoustic paths, the only information we need for computing the distribution of the reflections is their path length. Let k be an index that identifies the acoustic path and L_k the corresponding path length. Let f_s be the sampling frequency and c be the speed of sound. The time of arrival of the reflection related to acoustic path k is

$$n_k = \frac{L_k}{c} f_s. \tag{8}$$

If the walls of the environment all have the same reflection coefficient R, then the amplitude of the reflection associated to the acoustic path k is

$$a_k = \frac{R^j}{L_k} \tag{9}$$

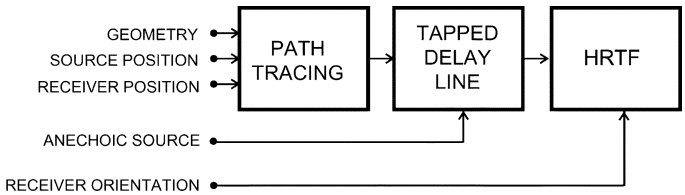


Fig. 23. Auralization system. Given the location of source and receiver and the geometric information of the environment, we can compute the acoustic paths that link source and receiver. In order to complete the auralization algorithm, all we need is the anechoic signal and the head orientation.

where j is the total number of reflections encountered by acoustic path k. The times of arrival of the echoes and their magnitude are computed for each acoustic path. The system can also quite effortlessly accommodate environments with different reflection coefficients for each wall. The resulting auralization system is shown in Fig. 23.

We ran some experiments in order to test whether the proposed technique is suitable for real-time tracing and auralization on low-cost computers. That platform was a simple laptop computer, which proved largely adequate for real-time operation with thousands of beams and paths, where both sources and receiver were moving. Although the acoustic plausibility of the results is limited by the modeling approximations (lack of diffraction and diffusion and other simplification), the spatial location of a moving source was correctly rendered, whether or not the visual feedback was enabled.

One final note concerns the costs of this particular implementation of our approach. We need to distinguish between filtering update costs (tracing costs) and auralization costs. Tracing costs have been thoroughly described in Section IV-B and they apply to this situation perfectly. As for the auralization costs, we need to consider the cost of a tapped delay line and that of the 16 parallel convolutions that implement the direction-dependent HRTFs. The HRTFs are implemented using the computational power of the sound card; therefore, all that is left to compute is the cost of the tapped delay line. Just to give an example, if we consider an environment with high level of mutual occlusion on 160 reflectors, and if we limit the iterations to a total of 10 000 beams, the average number of echoes (taps) that need be accounted for in the tapped delay line is about 225. The corresponding computational time for filter update (tracing) is 5×10^{-4} s.

VI. Conclusion and Future Work

In this paper, we proposed a novel approach to beam tracing that enables the on-the-fly construction of beam trees through a direct lookup of precomputed visibility maps. The impact of this solution to simulations of environmental acoustics, in fact, turned out to be significant, as it enables a real-time auralization in the presence of not just moving receivers but also moving sources. The proposed technique proved its effectiveness particularly when the floormap exhibits a significant level of complexity (numerous occlusions). The method proved suitable for advanced gaming applications and, in general, for all those applications that entail immersive auralization.

We are currently working on two levels of generalization for this approach. The first allows us to account for acoustic propagation phenomena such as diffraction and diffusion, with the goal of making the simulations more physically plausible. The second level will make the method suitable for modeling propagation in nonseparable 3-D environments.

References

[1] U. Krockstadt, "Calculating the acoustical room response by the use of a ray tracing technique," *J. Sound Vibrations*, vol. 8, no. 1, pp. 118–125, 1968.

[2] A. Glassner, R. Cook, E. Haines, P. Hanrahan, P. Heckbert, and L. Speer, *An Introduction to Ray Tracing*, 4th ed. London, U.K.: Academic, 1987.

[3] E. Veach and L. J. Guibas, "Metropolis light transport," in *Proc. 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'97)*, Los Angeles.

[4] J. Vian and D. V. Maercke, "Calculation of the room response using a ray tracing method," in *Proc. ICA Symp. Acoust. Theater Planning Performing Arts*, Toronto, ON, Canada, 1986, pp. 74–78.

[5] J. Allen and D. Berkley, "Image method for efficiently simulating small-room acoustics," *J. Acoust. Soc. Amer.*, vol. 65, no. 4, pp. 943–950, Apr. 1979.

[6] J. Borish, "Extension of the image model to arbitrary polyhedra," *J. Acoust. Soc. Amer.*, vol. 75, no. 6, 1984.

[7] U. Kristiansen, A. Krockstad, and T. Follestad, "Extending the image method to higher-order reflections," *J. Appl. Acoust.*, vol. 38, no. 2–4, pp. 195–206, 1993.

[8] N. Tsingos and J. Gascuel, "A general model for simulation of room acoustics based on hierarchical radiosity," in *Visual Proc. ACM Comput. Graphics, SIGGRAPH'97*, Los Angeles, CA, 1997, p. 149.

[9] C. Goral, K. Torrance, D. Greenberg, and B. Battaile, "Modeling the interaction of light between diffuse surfaces," in *Proc. ACM Comput. Graphics, SIGGRAPH'84*, Minneapolis, MN, 1984, vol. 18, no. 3, pp. 213–222.

[10] G. R. Moore, "An approach to the analysis of sound in auditoria," Ph.D. dissertation, Dept. of Architecture, Univ. of Cambridge, Cambridge, U.K., 1984.

[11] P. Heckbert and P. Hanrahan, "Beam tracing polygonal objects," in *Proc. ACM Comput. Graphics, SIGGRAPH'84*, Minneapolis, MN, 1984, vol. 18, pp. 119–127.

[12] N. Dadoun, D. Kirkpatrick, and J. Walsh, "The geometry of beam tracing," in *Proc. ACM Symp. Comput. Geometry*, Sedona, AZ, Jun. 1985, pp. 55–61.

[13] M. Monks, B. Oh, and J. Dorsey, "Acoustic simulation and visualisation using a new unified beam tracing and image source approach," in *Proc. 100th Audio Eng. Soc. Conv. (AES 96)*, Los Angeles, CA, 1996.

[14] U. Stephenson and U. Kristiansen, "Pyramidal beam tracing and time dependent radiosity," in *Proc. 15th Int. Congr. Acoust.*, Trondheim, Norway, Jun. 1995, pp. 657–660.

[15] J. P. Walsh and N. Dadoun, "What are we waiting for? The development of Godot," in *Proc. 103rd Meeting Acoust. Soc. Amer.*, New York, Apr. 1982, vol. 71, no. SI, p. S5.

[16] T. Funkhouser, I. Carlbom, G. Elko, G. Pingali, M. Sondhi, and J. West, "Beam tracing approach to acoustic modeling for interactive virtual environments," in *Proc. ACM Comput. Graphics, SIGGRAPH'98*, Orlando, FL, Jul. 1998, pp. 21–32.

[17] T. Funkhouser, P. Min, and I. Carlbom, A. Rockwood, Ed., "Real-time acoustic modeling for distributed virtual environments," in *Proc. ACM Comput. Graphics SIGGRAPH'99*, Los Angeles, CA, 1999, pp. 365–374 [Online]. Available: citeseer.ist.psu.edu/funkhouser99real-time.html, Addison Wesley Longma, [Online]. Available

[18] V. Koltun, Y. Chrysanthou, and D. Cohen-Or, "Hardware-accelerated from-region visibility using a dual ray space," in *Proc. 12th Eurographics Rendering Workshop*, London, U.K., 2001, pp. 205–216 [Online]. Available: citeseer.ist.psu.edu/koltun01hardwareaccelerated.html

[19] D. Cohen-Or, Y. Chrysanthou, and C. Silva, "A survey of visibility for walkthrough applications," in *Proc. EUROGRAPHICS'00, Course Notes*, Interlaken, Switzerland, 2000 [Online]. Available: citeseer.ist.psu.edu/cohen-or00survey.html

[20] F. Durand, "3D visibility: Analytical study and applications," Ph.D. dissertation, Univ. Joseph Fourier, Grenoble, France, Jul. 1999.

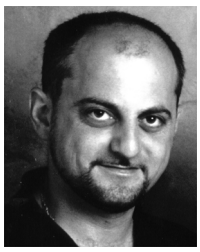
[21] M. Foco, P. Polotti, A. Sarti, and S. Tubaro, "Sound spatialization based on fast beam tracing in the dual space," in *Proc. 6th Int. Conf. Digital Audio Effects (DAFX-03)*, London, U.K., Sep. 2003, pp. 198–202.
[22] S. Teller and C. Sequin, "Visibility preprocessing for interactive walkthroughs," in *Proc. ACM Comput. Graphics, SIGGRAPH'91*, Los Alamitos, CA, 1991, vol. 25, no. 4, pp. 61–69.
[23] S. Teller, C. Fowler, T. Funkhouser, and P. Hanrahan, "Partitioning and ordering large radiosity computations," in *Proc. ACM Comput. Graphics, SIGGRAPH'94*, Orlando, FL, Aug. 1994, pp. 443–450.
[24] D. Schröder and T. Lentz, "Real time processing of image sources using binary space partitioning," *J. Audio Eng. Soc. (JAES)*, vol. 54, no. 7–8, pp. 604–619, 2006.
[25] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen, "The lumigraph," *Comput. Graphics* ser. Annu. Conf. Series, vol. 30, pp. 43–54, 1996 [Online]. Available: citeseer.ist.psu.edu/gortler961umigraph.html
[26] M. Levoy and P. Hanrahan, "Light field rendering," *Computer Graphics* ser. Annu. Conf. Series, vol. 30, pp. 31–42, 1996 [Online]. Available: citeseer.ist.psu.edu/levoy961ight.html
[27] B. Gardner and K. Martin, "HRTF Measurements of a Kemar Dummy-Head Microphone," Tech. Rep. 280, May 1994 [Online]. Available: citeseer.ist.psu.edu/gardner94hrtf.html, MIT Media Lab Perceptual Computing
[28] T. Leyvand, O. Sorkine, and D. Cohen-Or, "Ray space factorization for from-region visibility," *ACM Trans. Graphics (TOG)*, vol. 22, no. 3, pp. 595–604, 2003.
[29] J. Bittner and P. Wonka, "Visibility in computer graphics," *Environ. Planning B: Planning Design* vol. 30, no. 5, pp. 729–756, Sep. 2003 [Online]. Available: http://www.cg.tuwien.ac.at/research/publications/2003/Bittner-2003-Vis/
[30] L. Savioja, J. Huopaniemi, T. Lokki, and R. Väänänen, "Creating interactive virtual acoustic environments," *J. Audio Eng. Soc.*, vol. 47, pp. 675–705, Sep. 1999.
[31] J. Huopaniemi, L. Savioja, and T. Takala, "Diva virtual audio reality system," in *Proc. Int. Conf. Auditory Display (ICAD'96)*, Palo Alto, CA, Nov. 1996, pp. 111–116.
[32] [Online]. Available: http://www.openal.org2001, Creative Technology, Ltd. Creative OpenAL Programmers Reference 1.0

Fabio Antonacci was born in Bari, Italy, in 1979. He received the "Laurea" degree in April 2004.

Since 2004, he has been with Image and Sound Processing Group, Dipartimento di Elettronica ed Informazione, Politecnico di Milano, Milan, Italy. His research interests are mainly related to array signal processing (specifically to acoustic source separation and localization) and acoustic modeling (complex propagation modeling using fast beam tracing).

Marco Foco received the M.S. degree in computer engineering from the Politecnico di Milano, Milan, Italy, in 2003.

In 2003, he joined the Politecnico di Milano as a Junior Researcher, where he is focused on advanced acoustic rendering solutions. He recently established his own company (Codemachine), which focuses on real-time processing techniques.



Augusto Sarti (M'04) was born in 1963. He received the "Laurea" degree (cum laude) in 1988 and the Ph.D. degree in 1993 in electrical engineering from the University of Padua, Padua, Italy, with research on nonlinear communication systems. His graduate studies included a joint graduate program with the University of California, Berkeley, where he spent more than two years doing research on nonlinear system theory.

In 1993, he joined the Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milan, Italy, where he is currently an Associate Professor. His research interests are in the area of Digital Signal Processing, with particular focus on sound analysis and processing, image analysis, and computer vision. He coauthored more than 150 scientific publications on international journals and congresses as well as numerous patents in the image and sound processing area. He cofounded the Image and Sound Processing Group (ISPG) at the Dipartimento di Elettronica e Informazione, Politecnico di Milano, which has contributed to numerous national and international research Projects.

Stefano Tubaro (M'02) was born in Novara, Italy, in 1957. He completed his studies in electronic engineering at the Politecnico di Milano, Milan, Italy, in 1982.

He then joined the Dipartimento di Elettronica e Informazione, Politecnico di Milano, first as a Researcher of the National Research Council, then as a faculty member (Associate Professor in 1991, Full Professor in 2004). He initially worked on problems related to speech analysis, motion estimation and compensation for video analysis and coding, and vector quantization applied to hybrid video coding. More recently, his research interests have focused on image and video analysis for the geometric and radiometric modeling of 3-D scenes, advanced algorithms for video coding, and sound processing. He authored more than 150 scientific publications in international journals and congresses. He coauthored two books on digital processing of video sequences. He is also a coauthor of several patents in the area of image processing. He coordinates the research activities of the Image and Sound Processing Group (ISPG) at the Dipartimento di Elettronica e Informazione, which is involved in several research programs funded by industrial partners, the Italian Government, and by the European Commission.