3D Res (2016) 7:26
DOI 10.1007/s13319-016-0102-y

3DR EXPRESS

Discrete Method of Images for 3D Radio Propagation Modeling

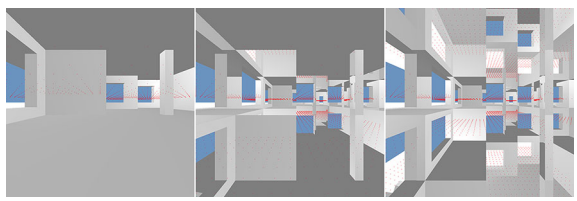
Roman Novak

Received: 17 June 2016 / Revised: 28 July 2016 / Accepted: 29 July 2016 / Published online: 6 August 2016

Abstract Discretization by rasterization is introduced into the method of images (MI) in the context of 3D deterministic radio propagation modeling as a way to exploit spatial coherence of electromagnetic propagation for fine-grained parallelism. Traditional algebraic treatment of bounding regions and surfaces is replaced by computer graphics rendering of 3D reflections and double refractions while building the image tree. The visibility of reception points and surfaces is also resolved by shader programs. The proposed rasterization is shown to be of comparable run time to that of the fundamentally parallel shooting and bouncing rays. The rasterization does not affect the signal evaluation backtracking step, thus preserving its advantage over the brute force ray-tracing methods in terms of accuracy. Moreover, the rendering resolution may be scaled back for a given level of scenario detail with only marginal impact on the image tree size. This allows selection of scene optimized execution parameters for faster execution, giving the method a competitive edge. The proposed variant of MI can be run on any GPU that supports real-time 3D graphics.

R. Novak (✉)
Department of Communication Systems, Jožef Stefan Institute, 1000 Ljubljana, Slovenia
e-mail: roman.novak@ijs.si

Graphical Abstract



Keywords Method of images · 3D radio ray tracing · Propagation prediction · Algorithm optimization

1 Introduction

The prediction of electromagnetic wave propagation is at the core of any wireless system design and essential for many services. When accompanied with detailed knowledge of the environment geometry, advanced ray-tracing techniques can take into account the majority of paths the real wavefront would traverse and model actual physical phenomena responsible for propagation of electromagnetic waves. Advanced channel characteristics, including delay spread and direction of arrival can be calculated from multipath traces, which are not readily available in pure stochastic propagation predictions.

Two computationally distinct radio ray-tracing approaches have been followed since early beginnings. The first, often seen as the brute force approach,

effectively traces a large number of rays from the transmitting source in all directions into the scene, while the second, known as the method of images, evaluates feasible propagation regions without resorting to a single ray granularity. The latter approach builds on the geometrical properties of a scene and is characterized by excellent prediction accuracy while modeling reflections and through-wall transmissions. In the following, we refer to the first group of algorithms as the shooting and bouncing rays (SBR) algorithms and to the second as the MI (method of images) algorithms.

We have seen numerous refinements of the method of images in the past two decades. In great majority, they are almost universally based on computational geometry for the source image tree construction, mathematically describing either regions of physically feasible ray paths or shapes of visible parts of surfaces involved in interactions with such paths. Undoubtedly, the advantage of algebraic treatment is high accuracy, even for distant areas. On the other hand, the SBR algorithms cast rays from the source in a limited set of directions, thus reducing accuracy with distance, but still produce valuable information on signal propagation in the areas of interest. The major advantages of such discretization are simplified numerical treatment of inherent problems and greater level of task independence, giving rise to efficient parallel implementations.

Here we introduce discretization into the MI as well. In order to speed up computationally demanding tasks involved in the three-dimensional source image tree creation process, finer-grained concurrency exploiting parallel nature of polygon rasterization is proposed. One could make an analogy between the rendered pixels and rays in a ray-shooting algorithm. The advantage of the MI is that it does not relay on the concept of a reception sphere to detect rays passing by the receivers, which is a common problem addressed by the SBR algorithms. We further compare the two approaches throughout the paper.

Discretization by rendering allows method of images to be efficiently run on a massively parallel architecture. By relying on the standard graphical interface, efficient implementations on today's conventional 3D accelerators are possible. OpenGL terminology is used in the paper, although the presented concepts are not limited to the chosen programming interface.

The accuracy of the radio ray tracing is not pursued here because we are dealing only with the algorithmic aspects of the technique that has minor influence on the prediction outcome. Both, computational geometry based method and the proposed rendering approach build identical visibility trees up to a given precision. Thus, no validation using field measurements is necessary. Moreover, the subject of prediction accuracy has been studied extensively in the past two decades with abundant literature available to the interested reader.

The contributions of this paper can be summarized as follows:

- Rasterization is effectively applied to the 3D radio signal prediction domain.
- Method of images, traditionally based on computational geometry, is discretized.
- Derived is the method's asymptotic run time on massively parallel architectures present in today's graphical 3D accelerators.
- Analogy is established between rendered pixels and shooting and bouncing rays.
- Demonstrated are performance benefits of the discretization over the shooting and bouncing rays.

In the following, Sect. 2 reviews the related work. Basic steps in the source image tree creation are explained in Sect. 3. The algorithm for the image tree construction and traversal is wrapped up in Sect. 4. We show that the parallel run-time complexity of the proposed method is close to that of the SBR algorithms executed at comparable spatial resolution in Sect. 5. Finally, the run-time performance is evaluated in Sect. 6, followed by the conclusion in Sect. 7.

2 Related Work

In global illumination domain, ray tracing is based on a physical optics approximation, where paths of light rays are followed through the scene, with processing of various sorts on surface intersections in order to reproduce reality as close as possible. Ikegami et al. [1] were among the first who showed the usefulness of ray-tracing technique for radio wave propagation prediction in 1991. A larger set of electromagnetic effects are typically dealt with at radio frequencies. For example, diffraction [2] and interference are

generally not considered in the global illumination problem, although some exceptions exist [3].

First, we recall briefly the basics of the method of images and its tree building and signal backtracking steps. The method is designed around the observation that rays reflected from a surface seem to originate from a fictitious transmission source, which can be found symmetrically on the other side of the surface along its perpendicular [4]. Such a fictitious source is called a source image. Multiple reflections are accounted for by considering existing images as new transmission points, which recursively leads to the image tree. Surfaces have a finite shape, limiting the reflected regions to polyhedron volumes, which helps in reducing the size of a tree. If the image tree describes only viable propagation paths, then it is referred to as a visibility tree.

Image trees are generally larger than visibility trees and require elaborate intersection tests between the ray paths and scene objects in the second backtracking step. Note that a backtracking is always required to re-create signal paths, even in a strict visibility tree, at least to establish the length of a path and all the reflection coefficients affecting the radio signal power.

Image trees are not limited solely to reflected paths. Through-wall transmissions can be dealt with successfully by extending the source mirroring onto walls that are identified in the so-called transmission regions [5]. On the other hand, corner diffractions can be adequately handled by the method of images only in 2D propagation scenarios [6]. Namely, a corner acts as a source of rays, generally introducing an infinite number of new sources unless it is represented by a single point. Therefore, in 3D diffraction must be accounted for by some other means.

Soon after its introduction, proposals to reduce the over dimensioned image tree emerged. In the following, we restrict ourselves to the published works that support full 3D environment modeling through the entire computation and to the ones being most relevant to our proposal.

The elaborate method of regions [4] constrains physically feasible paths by introducing spatial regions in the shape of convex polyhedra into the image tree construction. Computing viable reflection or transmission regions translate to the polyhedra intersection problem, the solution of which involves intensive computational geometry. Simplified version of the spatial regions treatment can be found in [7], where the geometry of objects is restricted to horizontal walls of arbitrary shape and strictly rectangular vertical walls.

Instead of bounding regions, [8] deals with a set of visible surfaces contained within such feasible regions. Surfaces are represented by polygons as seen from the source image after the surface corners have been projected to the viewing plane. In order to extract polygons describing only visible parts of a surface, each projection is processed by a sweep-line algorithm, followed by the well-known graph-theoretic polygon subtraction to account for hidden parts of a surface. The reflection visibility window is represented by yet another polygon in the computation, generally hindered by the treatment of many special cases.

Visibility tree in [9] is only partially reduced image tree because it is based on a polar-sweep of 2D space. When applied to 3D scene in the second backtracking step, paths described by the tree may or may not give rise to actual paths and further checks are still needed. Further, being a hybrid 2D/3D method, ground, floors and ceilings must be treated separately.

Further attempts to reduce image tree size while keeping the support for full 3D computation involve various pre-processing steps on the input geometry, such as dividing surfaces into tiles and using the tile center as the ray interaction point [10], or pre-computing intermediate values needed in the highly repetitive intersection tests, such as angular relations between the scene objects [11].

The above improvements strive to reduce the computational complexity of the problem. On the other hand, despite the increasing availability of parallel hardware, solving the problem in a parallel way has been largely avoided. The reason can be related to the fact that the imaging technique is not as parallel as the SBR technique [12], for which various hardware accelerators have been intensively studied, such as Cell architecture [13], SaarCOR processor [14], and FPGAs [15].

The straight-forward way to parallelize the method of images is to partition the image tree and use some form of dynamic load balancing, such as splitting the area database in regions and using work-pool concept [16]. The authors of [16] characterize their approach as being fine-grained. However, compared to our proposal, it is still rather coarse and better suited for

parallel computers in general and less for the GPU-like massively parallel architectures.

The closest to the fine-grained parallelization presented here is the rasterization of beams in [17], with beams being considered equivalent to the feasibility regions. Unfortunately, Schmitz et al. designed custom rendering pipeline that can be applied only to a 2D version of propagation prediction and cannot be extended to the third dimension.

GPUs has been studied extensively in the context of shooting and bouncing rays [18–20]. Recent work includes ray launching based on the NVIDIA OptiX framework [21, 22], GPU-based kd-tree accelerated beam tracing [23], channel modelling using a 3D game engine [24], and others. By aggregating rays that interact with the same surfaces, the SBR algorithm effectively converges to the method of images. Thus, the SBR is closely related and it serves as the evaluation reference.

3 Rendering Image Tree Branches

Two tasks prevail in the 3D image tree construction. First, a form of spatial ordering has to be established between the scene objects in one way or another to constrain viable propagation regions and determine surface visibility. Second, shapes of regions or visible parts of surfaces have to be recognized and maintained. Convex polyhedra intersection and polygons subtraction are two common techniques proposed to solve these two problems with high precision.

Similar problems arise in the rendering of an image onto the computer screen, which is represented internally by a discrete framebuffer. Here, visibility of a scene object and its shape are resolved by combining perspective projection with the hidden surface removal. By using the so called z-buffering, spatial ordering is achieved in a discrete way on a pixel-by-pixel basis. One can identify surfaces and their affected areas hit by a spherical wavefront from a 2D rendering, as illustrated in Fig. 1. First, a 3D scene consisting of three wall-shaped objects (a) is shown together with a view frustum from the source of transmission. The actual rendering (b) exposes visible surfaces hit by the line-of-sight radio waves. Six frustums with 90-degree field of view should be stacked together in a cube to cover the entire space.

3.1 Rendering Reflections

According to the method of images, the reflection from a point source virtually originates from its image located symmetrically to the reflecting plane. Rendering a mirror image basically uses the exact same principle, only that the entire view frustum is mirrored. The reflection region is further bound on its sides by the shape of the reflecting surface and in front by the reflecting plane. Stencil and plane clipping operations in graphics shaders perform equivalent tasks, effectively limiting the rendered image to the view within the reflection region. In Fig. 1, the visible parts of all surfaces in the reflection region of the wall with a window (c) are revealed in framebuffer (d). First, a stencil in the shape of the wall's front surface is set up by redrawing the front surface over image (b). We need to redraw the object over the resolved depth buffer in order to define stencil only for the visible pixels. Next, the view frustum from (a) is mirrored over the reflecting plane and the scene is drawn over the stencil mask while being also clipped off at the reflecting plane. The result of the above sequence of operations is depicted in framebuffer (d), which is augmented with the original view to be more concise. An actual implementation would draw only the shaded areas.

The above technique of rendering reflections can be applied to reflections within reflections, allowing stepping through the image tree using the recursive algorithm presented in Sect. 4. The technique is not constrained by the prescribed surface primitive. Namely, stencil supports arbitrary shaped surfaces, including those having holes and consisting of multiple parts. Being less restrictive in surface definition also implies less image tree nodes. Using the above rendering approach to identify scene objects encountered by the reflected spherical wavefront follows the laws of geometric optics and is accurate down to a single pixel.

3.2 Rendering Double Refractions

Modeling refraction phenomena in a similar way is more challenging and some compromises must be made. At the core of the problem is the distortion of a spherical wavefront as it enters different medium of transmission. As the radio rays pass into different medium, the shape of a surface hit by the distorted

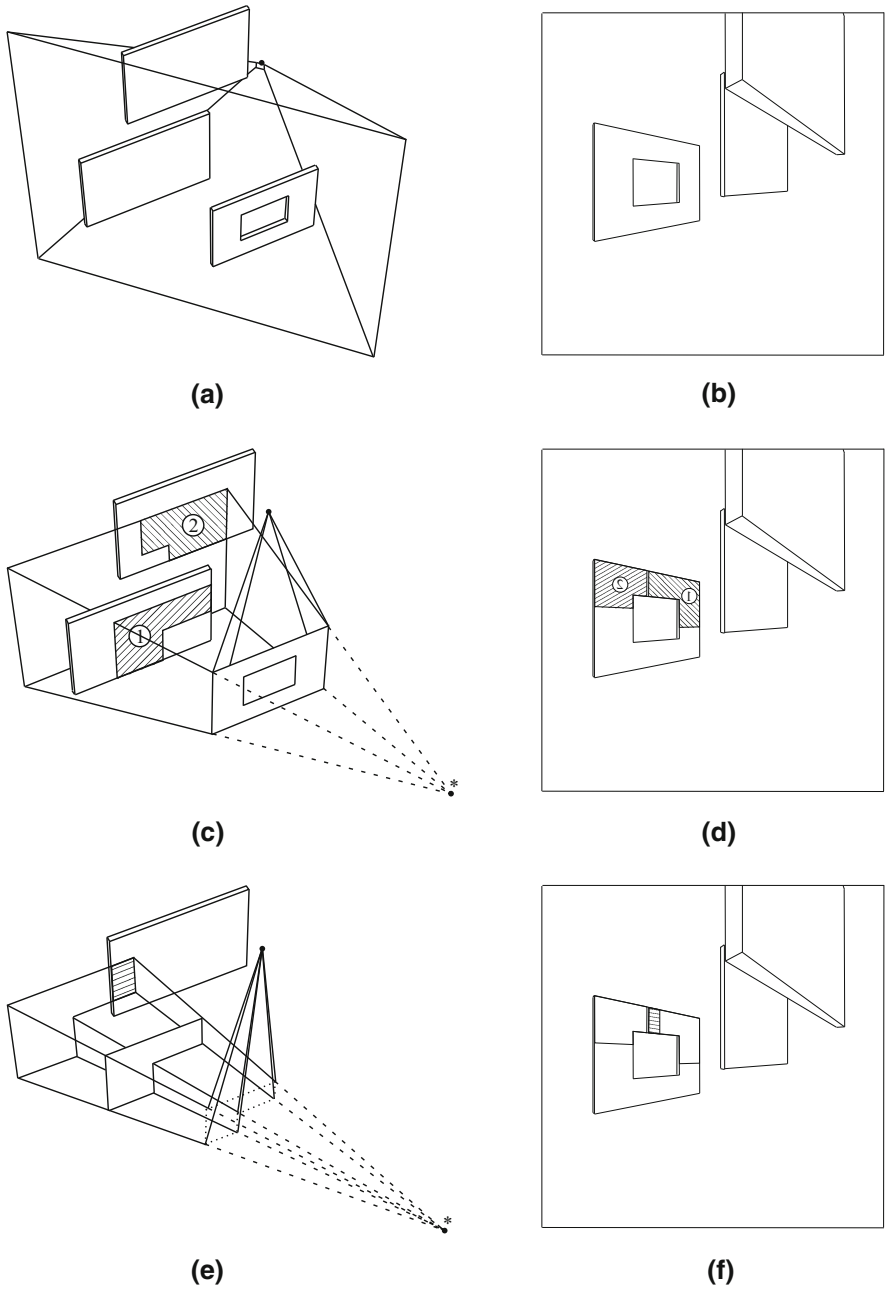


Fig. 1 Rendering of surfaces hit by a spherical wavefront; a view frustum in the sample scene (**a**) defines line-of-sight surfaces in framebuffer (**b**). Surfaces hit by the reflected (**c**) and later double-refracted (**e**) wavefront are revealed in framebuffers (**d**) and (**f**) using standard stencil test, z-buffering and plane clipping

wavefront or the volume in which these rays spread cannot be described by a simple polygon or convex polyhedron. Thus, proposals that incorporate transmission branches in the image tree normally model objects as slabs with predefined thickness, such as walls, for which double refraction on the front and on the back surface to some extent cancels out the error. A straight line of the through-the-object rays is usually assumed, which is a good approximation only for a thin slab located far from the source. In order to keep focus of the paper on the algorithmic aspects and minimize influence on the calculated signal, we make the same assumptions and, in the following, only briefly discuss a potential improvement.

In reality, due to Snell's law of refraction, a spherical wavefront is transformed into a hyperbolical

shape [25] after it passes through a plane interface into different propagation medium. An attempt to find a common viewpoint by extending refracted rays back toward the source would necessary fail. For the same reason, double refraction on parallel plane interfaces, with the signal re-entering the initial transmission medium on the second refraction, realigns the double refracted ray's direction with the incident ray's direction. However, as illustrated in Fig. 2, a small parallel displacement remains.

Because ray displacement depends on its incident angle θ_{inc}, an error-free through-wall view that would accurately depict scene hit by the double refracted wavefront is not possible. Nevertheless, a translation of the viewpoint in the direction opposite to the surface normal, i.e., Δ in Fig. 2, can reduce the misalignment angle. The misalignment angle ε between the double-refracted ray and the matching line-of-sight direction from the Δ-corrected viewpoint location is defined as

$$\varepsilon = \theta_{\text{inc}} - \tan^{-1}\left(r/(l + d - \Delta)\right) \tag{1}$$

with

$$r = l \tan\theta_{\text{inc}} + d \tan\theta_{\text{tr}}, \tag{2}$$

where l is the source to wall distance, d is the depth of a wall, θ_{inc} is the ray's angle of incidence and θ_{tr} is the angle of transmission for media refractive indices n_1 and n_2 in compliance with $\sin\theta_{\text{tr}} = (n_1/n_2)\sin\theta_{\text{inc}}$. Note that ε shown in Fig. 2 is for a viewpoint matching the source.

When rendering a refraction view through a wall, correction Δ would ideally be selected in a way to minimize the expected misalignment angle over the incident angles for the actually visible parts of the wall. As mentioned previously, finding the best translation for a given view is beyond the scope of this paper. The interested reader is referred to the forthcoming publication on the subject.

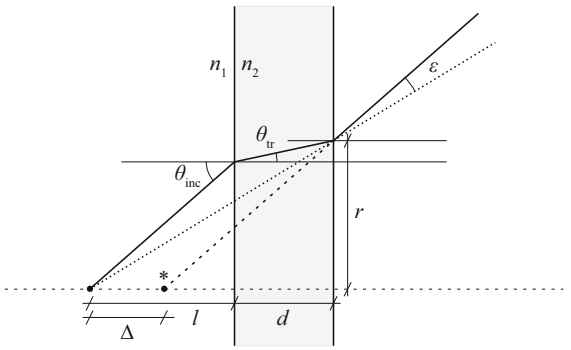


Fig. 2 Double refraction on parallel planes

In order to illustrate the concept of rendering a transmission region, we start with the content of framebuffer (d) in Fig. 1, where two visible surfaces has already been identified. Let the next branch in the image tree be a double refraction through the surface no. 1. In order to render the transmission region, we first define a stencil mask in the shape of this surface. We can do that by redrawing the surface using the depth buffer, the stencil buffer and the view frustum from the last step. When the transmission stencil is set up, the last frustum is redrawn, this time using front face of the wall no. 1 as the clipping plane. Rendering (f) in Fig. 1 shows the visible surface that is hit by the propagating wavefront after being reflected and double-refracted by the selected walls.

It has to be noted that the double refraction misalignment error affects only visibility of objects while the through-wall path of a ray in a signal backtracking procedure still remains exact. When all the intermediate planes, matching refraction coefficients, source and destination are known, re-creating exact path is simply a matter of geometry.

4 Image Tree Traversal

The framebuffer memory requirements are high and storing the entire image tree for later traversal would be prohibitively expensive. Thus, the signal evaluation step must be interleaved with the tree discovery, allowing the reuse of the framebuffer space.

Algorithm 1 Recursive image tree traversal

```
1: procedure step(type, surface, view, depth)
2:   set stencil test condition to stencil == depth − 1;
3:   increment stencil for visible surface pixels
4:     by redrawing surface in view;
5:   If type == REFLECT then
6:     mirror view over surface and set next_view;
7:   If type == REFRACT then
8:     translate view toward surface and set next_view;
9:   set stencil test condition to stencil == depth;
10:  draw next_view;
11:  identify visible receivers;
12:  accumulate signal in visible receivers;
13:  If depth < max_depth then
14:    identify visible surfaces;
15:    For each visible surface do
16:      step(REFLECT, surface, next_view, depth +1);
17:      step(REFRACT, surface, next_view, depth +1);
18:    end
19:  unwind stencil;
20: end
```

Initially, we draw a scene six times, each time using a frustum covering one-sixth of the space. The viewpoint of each of the six views, i.e., eye location, is at the source of transmission. We denote such views as baseline views. Next, line-of-sight receivers are identified. Signal is accumulated in the identified receivers and Algorithm 1 is called twice for each visible surface, once for the reflection branch and once for the double refraction branch of the image tree. The identification of visible receivers and the identification of visible surfaces are also part of the recursive step and explained shortly.

While the framebuffer's color and depth components are overwritten repeatedly in Algorithm 1, the framebuffer's stencil component acts as a virtual stack. By incrementing stencil buffer values for a visible part of the surface that is currently targeted for a reflection or double refraction, next stencil is pushed on the virtual stack at the beginning of a recursive call. This is possible because a stencil is always contained within the stencil of the shallower recursion step. At the end of the recursion, one can simply pop or unwind the last stencil by drawing a full viewport quad over the existing stencil and requesting a stencil decrement. 8-bit stencil components are common today, enabling up to 2^8 interactions on each viable signal path.

Having set up a stencil, we mirror or translate the current view frustum, effectively computing the source image for the reflection or for the double refraction branch of the image tree. Next, the transformed view is actually drawn using surface color-coding and the most recent stencil. The color-coding is exploited a few steps later while identifying visible surfaces.

In order to establish the reception points visibility in the rendered view, it suffices to compare each reception point depth—while being observed from the source image location—with the already computed depth buffer and test the point's side relative to the frustum's front clipping plane. The computation actually mirrors the transformations involved in the drawing of a point but with no actual rendering. It can be implemented entirely in a custom shader acting on its own visibility framebuffer while being fed with the set of reception points and with the current depth buffer as a depth texture.

Once the location of a receiver is confirmed to be visible, signal is accumulated by retracing source images down to the transmission source. The signal accumulation procedure can be any standard technique of summing coherent rays taking into account antenna pattern, polarization, material composition, frequency, etc.

Visible surfaces in the current rendering need to be identified next in order to continue with the source image tree traversal. Because any rendering can produce erroneous pixels that protrude between triangle seams for the obstructed surfaces, invalid pixels should also be filtered out. By adopting color-coding of surfaces in the earlier rendering step, the task translates to computing a color histogram. A surface with the number of drawn pixels above the threshold is then recognized as visible. Note that the threshold can also be used to balance the prediction speed versus accuracy by ignoring smaller or far away surfaces. Finally, two recursive calls for each visible surface take care of subsequent reflections and transmissions.

5 Complexity Analysis

Discrete MI can be viewed as a ray-shooting algorithm, which motivates the following comparison of the proposed MI asymptotic behavior with that of an SBR algorithm. In order to establish a common ground between the two approaches, we first pair the non-masked pixels of each rendering step with rays in a ray-shooting algorithm.

Let the SBR algorithm initially launch r rays and the discrete MI use a framebuffer with p pixels. Ray equivalency is achieved when $r = 6p$. The fact that there is a variation in angular distribution between the imaginary through-pixel rays while the SBR algorithms commonly require uniform distribution of the initial rays over all directions is irrelevant, because we are only establishing problem size.

5.1 SBR Parallel Run Time

Central to any SBR algorithm is a search for the nearest point in which ray intersects with objects of the examined scenario. Trivial testing of a ray against every object in the scene is inefficient due to linear dependency on the number of scene objects. Optimized data representation of a scene is needed for, on

average, $O(\log n)$ search time, which is the theoretical lower bound given n scene primitives [26]. Further, m reception spheres must undergo similar intersection tests, effectively increasing the problem size to $n + m$. Therefore, the processing of initial rays up to the first interaction is of $O(r \log(n + m))$ time. Rays reflected from the same surface are observed as a group. Such a group is optimally processed in $O(s \log(n + m))$ time, where s is the number of reflected rays from a single surface. Similar conclusion can be drawn for a group of rays involved in a single through-wall transmission.

The above tasks can be efficiently parallelized on a standard GPU architecture. Assuming that c cores are available on a generic GPU, the parallel run times evaluate at $O(r/c \log(n + m))$ and $O(s/c \log(n + m))$ time for the above two cases.

5.2 MI Parallel Run Time

Efficient rendering of views in the proposed discrete MI needs to deal with similar optimization problems. A search for the nearest point in the direction of a through-pixel ray is replaced by a hidden surface removal technique, which is a fundamental procedure in computer graphics. Conceptually, one could use the ray-tracing solution for hidden surface removal, which limits the parallel rendering time to $O(s/c \log n)$, with s now being equal to the stencil area pixel count or to the number of pixels in the baseline views. In practice, hardware z-buffer would probably be used to resolve visibility of a pixel. In that case, visibility culling algorithms are needed to approximate the above run-time complexity, such as view-frustum culling, stencil culling, back-face culling and occlusion culling. Given that multi-pixel visibility tests in culling algorithms are faster than ray-by-ray intersection tests, a smaller constant factor is expected in the above complexity. Although simple geometry tests exist for removing objects not intersecting the view frustum and for removing primitives facing away from the viewpoint, occlusion culling needs more sophisticated approach. Here, highly efficient occlusion culling algorithms for special environments, such as densely obstructed building interiors [27], can help match or even surpass the SBR run-time complexity.

The rendering of a view is followed by the identification of visible receivers. The visibility test described in Sect. 4 cannot be directly compared to the SBR handling of reception spheres, which is tightly interwoven with the ray propagation through the sphere-extended scene. However, the procedure is closely related to the shadow mapping in computer graphics. This allows identification of visible receivers entirely on a GPU using specially designed shader, which does actual visibility check using a shadow sampler and a projective depth texture lookup. Thus, reception point visibility can be established in $O(m/c)$ parallel time.

Next time-consuming step is identification of visible surfaces based on a surface histogramming. Computing a histogram in parallel has its limits because many pixels map to the same bin, which implies some serialization of updates. Using scatter-based technique from [28] the parallel run-time is still kept at $O(p/c)$. Further, $O(p/c)$ is also parallel run time complexity of drawing and unwinding the stencil buffer.

5.3 Comparison

From the above, the discrete method of images comes close to the SBR algorithm with respect to the processing time needed for reflections and diffractions, i.e., $O(s/c \log(n + m))$ versus $O(s/c \log n + m/c + p/c)$. The way receivers are handled gives some advantage to the SBR asymptotic behavior only if the number of receivers is prevailing factor in the problem size, but at the cost of path accuracy, as discussed in the next section. Assuming ray equivalency, r and p are equal up to a constant. Thus, the same conclusion applies when comparing initially shot rays with the six baseline views.

6 Performance Evaluation

The performance evaluation tests were designed to check the run-time characteristics of the algorithm in comparison to the shooting and bouncing rays alternative on the same rendering hardware. The discrete method of images was implemented as an indoor wireless planner tool, closely following the concepts outlined in this paper.

As a reference, in-house SBR tool based on the ray-density normalization is used. The tool has been already proven in several projects with telecommunication industry. It is highly optimized GPU-based ray tracer using the NVIDIA OptiX ray-tracing engine, which is adapted to the radio propagation

environment. Scene objects are kept in a bounding volume hierarchy entirely on a GPU, with rays generated and traced through the scene in parallel threads. The tool was selected over the commercially available prediction tools because it could be fully customized in order to match the total electric field phasor computations in both tools. Further, diffraction and scattering have been disabled as these effects are not supported by the imaging technique. Knowing all the intricate details of the implementation, no shortcuts, such as dividing surfaces into tiles and other closely-guarded trade secrets of commercial tools, could bias the comparison.

The measurements have been performed on several indoor scenarios, with the following results referring to the office scenario shown in Fig. 3. The prediction plane is located at 1.1 m above the ground with 3 reception points per meter. The transmitting dipole operates at Wi-Fi frequency of 2.5 GHz. It is positioned near the office entrance at 2.2 m above the ground and at 45-degree inclination.

First, we compare the performance of ray-equivalent setups as defined in Sect. 5. The number of rays for the SBR graph in Fig. 4 was fixed by the icosahedron tessellation frequency of 12, resulting in 167,772,162 initial rays. Thus, the ray-equivalent framebuffer size for the rasterization graph was set to 5,288 by 5,288 pixels. As expected, the running time increases exponentially with the number of interactions considered on the signal path, i.e., with the number of ray segments for the SBR and with the image tree depth for the MI implementation. The rasterization approach was faster for the shallower depths up to 5 interactions, while the SBR implementation was a clear winner at the higher number of interactions. The faster exponential growth of the ray-equivalent MI can be largely attributed to the time-consuming memory transfers between the rasterization GPU and the CPU, the latter being responsible for the overall algorithm flow and signal accumulation. Further, the MI under the test had a limited capability of hidden surface removal, with only frustum and back-face culling actually implemented. On the other hand, the SBR run entirely on a GPU, including the signal accumulation, and was not subject to such limitations. Note that a GPU-only MI would be possible; however the solution could not be used on any rendering hardware that supports 3D graphics.

Another limiting factor is that the proposed reception point visibility check is linear in the number of points vice versa the logarithmic dependency in the SBR implementation, where the reception spheres are merged with the scene in a single acceleration structure. Then again, building such a structure adds to the initialization time, shown as the higher vertical offset of the SBR graph in Fig. 4.

The above tests favor SBR over the proposed MI at ray equivalency, at least for deeper traces. However, one must take into account the exceptional path accuracy of the imaging technique to have an objective comparison. A faster computation of the discrete method of images is possible because of the different sources of prediction errors between the two approaches. While some errors have non-algorithmic origins, such as inaccurate description of the

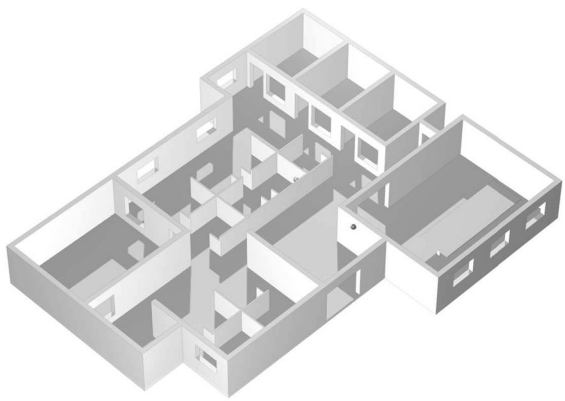


Fig. 3 Office test scenario with transparent ceiling

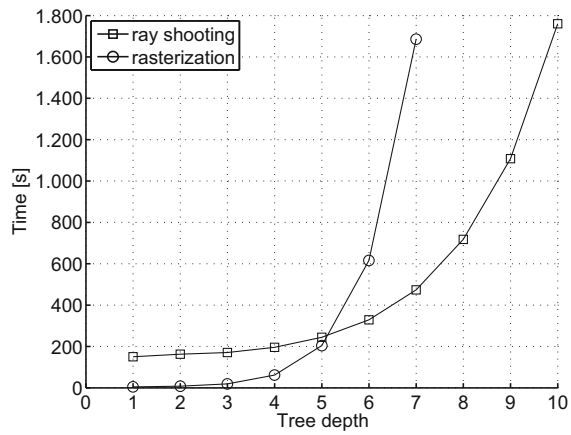


Fig. 4 Performance of optimized GPU-only SBR versus a mixed GPU/CPU discrete MI at ray equivalency

environment geometry, others can be attributed to the choice of input parameters. The angular separation of rays launched from the transmission source in the SBR type of algorithms influences spatial precision of computation, i.e., the ability of a ray to intersect distant objects. Diverging rays can miss smaller objects entirely and thus affect prediction at distance. Another closely related source of prediction errors is the use of a reception sphere to detect rays contributing to the total received power. Ray-paths that intersect a sphere are only approximate for a given reception point. Moreover, multiple hits by rays from the same wavefront require special treatment, with implemented ray-density normalization being one of them, which introduces further errors in the estimate of a real signal.

On the other hand, the proposed discrete method of images decouples these two sources of imprecision. While the spatial precision is directly related to the chosen framebuffer size, the ray-path is exact. Therefore, ray-equivalent setup, as defined in Sect. 5, is not a prerequisite for a comparable signal prediction quality. While one cannot decrease the number of launched rays without significantly affecting the result in the SBR algorithm, there is only a minor penalty if lower than ray-equivalent number of pixels is used in the discrete method of images. Fig. 5 plots the size of the image tree for the scenario in Fig. 3, i.e., the number of reflections and transmissions, achievable under different framebuffer sizes and tree depths. Note how the tree size changes only marginally for larger framebuffers. The converging tree size implies that for

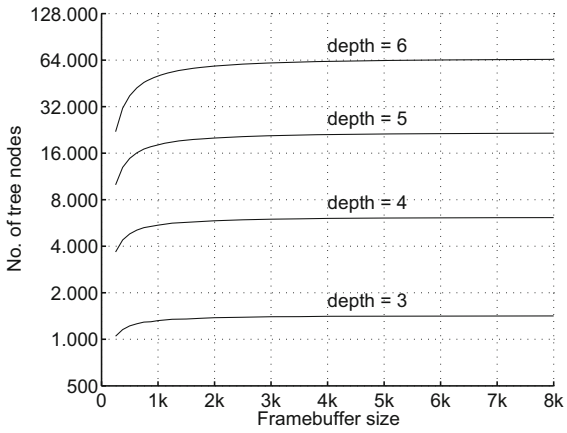


Fig. 5 Spatial resolution quantified through the image tree size at various depths and framebuffer sizes

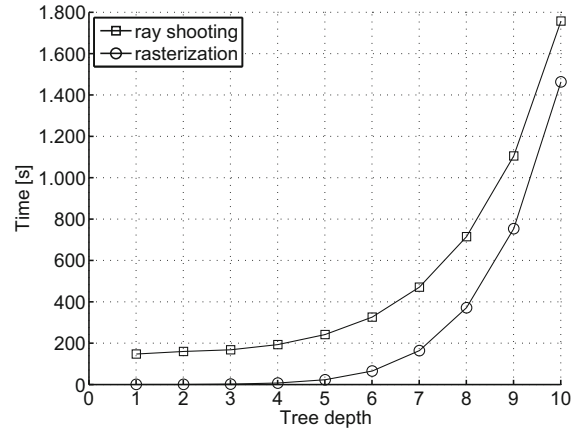


Fig. 6 Performance of the rasterization approach versus the ray shooting at comparable prediction accuracy; 167 million rays were launched initially as opposed to 4 megapixel rasterization

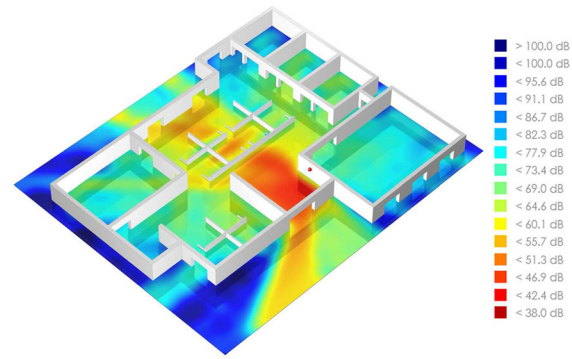


Fig. 7 Predicted signal loss in dB; the prediction was run for 3 reception points per meter at 1.1 m height above the ground and with 45-degree inclined dipole at Wi-Fi frequency near the office entrance (*red dot*). Six interactions per ray were considered. (Color figure online)

a given scenario increasing the rendering resolution does not produce much more views with previously undetected details. Further, because ray-path precision does not depend on the number of pixels used, one can achieve a comparable prediction quality at lower framebuffer size.

As shown in Fig. 6, the implemented discrete MI stays below the SBR running time for depths up to 10 interactions if framebuffer size is reduced from 5k (26,214,400 pixels) to 2k (4,194,304 pixels).

The surface plot in Fig. 7 shows the predicted signal loss for the scenario under consideration using 4 megapixel rasterization. Six interactions were simulated. Using larger framebuffer size showed only

minor visual differences at the prediction plane edges, amounting to less than 10 dB maximum deviations.

It should be noted that the running time of the proposed method is linear in the number of reception points. Increasing this number slows down the proposed algorithm faster than it slows down the SBR implementation with its logarithmic dependency.

7 Conclusion

This paper has introduced a discretization into the well-known method of images for radio wave propagation modeling. The source image tree being built is exact visibility tree, thus no further validation of discovered signal paths is necessary. The algorithm has low starting overhead and significant level of concurrency. Further, by tuning spatial precision to a scenario under consideration it can outperform the SBR alternative for reasonable configuration settings while retaining the MI prediction quality. In the proposed form it can be run on most of today's 3D graphical processors. GPU-only implementation is also foreseen in order to avoid remaining unnecessary memory transfers.

References

1. Ikegami, F., Takeuchi, T., & Yoshida, S. (1991). Theoretical prediction of mean field strength for urban mobile radio. *IEEE Transactions on Antennas and Propagation*, *39*(3), 299–302.
2. McNamara, D. A., Pistorius, C. W. I., & Malherbe, J. A. G. (1990). *Introduction to the uniform geometrical theory of diffraction*. Norwood, MA: Artech House, Antennas and Propagation Library.
3. Stam, J. (1999). Diffraction shaders. In *Proceedings of the 26th Annual Conference on computer graphics and interactive techniques (SIGGRAPH), Los Angeles, CA* (pp. 101–110).
4. Kimpe, M., Leib, H., Maquelin, O., & Szymanski, T. H. (1999). Fast computational techniques for indoor radio channel estimation. *Computing in Science & Engineering*, *1*(1), 31–41.
5. McKown, J. W., & Hamilton, R. L. (1991). Ray tracing as a design tool for radio networks. *IEEE Network: The Magazine of Global Internetworking*, *5*(6), 27–30.
6. Athanasiadou, G., Nix, A., & McGeehan, J. (2000). A microcellular ray-tracing propagation model and evaluation of its narrow-band and wide-band predictions. *IEEE Journal on Selected Areas in Communications*, *18*(3), 322–335.
7. Fortune, S. (1996). A beam-tracing algorithm for prediction of indoor radio propagation. In *Selected papers from the workshop on applied computational geormetry, towards geometric engineering* (pp. 157–166).
8. Maurer, J., Drumm, O., Didascalou, D., & Wiesbeck, W. (2000). A novel approach in the determination of visible surfaces in 3D vector geometries for ray-optical wave propagation modelling. In *Proceedings of the 51st IEEE vehicular technology conference (VTC 2000-Spring), Tokyo, Japan* (Vol. 3, pp. 1651–1655).
9. Agelet, F., Formella, A., Hernando Rabanos, J., de Vicente, F., & Fontan, F. (2000). Efficient ray-tracing acceleration techniques for radio propagation modeling. *IEEE Transactions on Vehicular Technology*, *49*(6), 2089–2104.
10. Rautiainen, T., Hoppe, R., & Wolfle, G. (2007). Measurement and 3D ray tracing propagation predictions of channel characteristics in indoor environments. In *IEEE 18th international symposium on personal, indoor and mobile radio communications (PIMRC), Athens, Grece* (pp. 1–5).
11. Ashour, M., Micheal, S., Khaled, A., el Shabrawy, T., & Hammad, H. (2014). A preprocessing dependent image theory based ray tracing algorithm for indoor coverage solution. In *Proceedings of the wireless communications and networking conference (WCNC), Istanbul, Turkey* (pp. 299–304).
12. Yun, Z., & Iskander, M. (2015). Ray tracing for radio propagation modeling: Principles and applications. *IEEE Access*, *3*, 1089–1100.
13. Benthin, C., Wald, I., Scherbaum, M., & Friedrich, H. (2006). Ray tracing on the cell processor. In *IEEE symposium on interactive ray tracing (RT), Salt Lake City, UT* (pp. 15–23).
14. Schmittler, J., Wald, I., & Slusallek, P. (2002). SaarCOR: A hardware architecture for ray tracing. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on graphics hardware, Saarbrucken, Germany* (pp. 27–36).
15. Woop, S., Schmittler, J., & Slusallek, P. (2005). RPU: A programmable ray processing unit for realtime ray tracing. *ACM Transactions on Graphics*, *24*(3), 434–444.
16. Athanaileas, T. E., Athanasiadou, G. E., Tsoulos, G. V., & Kaklamani, D. I. (2010). Parallel radio-wave propagation modeling with image-based ray tracing techniques. *Parallel Computing*, *36*(12), 679–695.
17. Schmitz, A., Rick, T., Karolski, T., Kuhlen, T., & Kobbelt, L. (2011). Efficient rasterization for outdoor radio wave propagation. *IEEE Transactions on Visualization and Computer Graphic*, *17*(2), 159–170.
18. Aila, T., & Laine, S. (2009). Understanding the efficiency of ray traversal on GPUs. In *Proceedings of the conference on high performance graphics (HPG), New Orleans, LA* (pp. 145–149).
19. Catrein, D., Reyer, M., & Rick, T. (2007). Accelerating radio wave propagation predictions by implementation on graphics hardware. In *Proceedings of the 65th IEEE vehicular technology conference* (pp. 510–514).
20. Tao, Y., Lin, H., & Bao, H. (2010). GPU-based shooting and bouncing ray method for fast RCS prediction. *IEEE Transactions on Antennas and Propagation*, *58*(2), 494–502.
21. Parker, S. G., Bigler, J., Dietrich, A., Friedrich, H., Hoberock, J., Luebke, D., et al. (2010). OptiX: A general purpose ray tracing engine. *ACM Transactions on Graphics*, *29*(4), 66.

22. Schiller, M., Knoll, A., Mocker, M., & Eibert, T. (2015). GPU accelerated ray launching for high-fidelity virtual test drives of VANET applications. In *Proceedings of the international conference on high performance computing & simulation (HPCS), Amsterdam, Netherlands* (pp. 262–268).
23. Tan, J., Su, Z., & Long, Y. (2015). A full 3-D GPU-based beam-tracing method for complex indoor environments propagation modeling. *IEEE Transactions on Antennas and Propagation*, *63*(6), 2705–2718.
24. Cadavid, A., Ibarra, D., & Salcedo, S. (2014). Using 3-D video game technology in channel modeling. *IEEE Access*, *2*, 1652–1659.
25. Rappaport, C. (2011). A novel, non-iterative, analytic method to find the surface refraction point for air-coupled ground penetrating radar. In *Proceedings of the 5th European conference on antennas and propagation (EUCAP), Rome, Italy* (pp. 1786–1789).
26. Szirmay-Kalos, L., & Márton, G. (1998). Worst-case versus average case complexity of ray-shooting. *Computing*, *61*(2), 103–131.
27. Luebke, D., & Georges, C. (1995). Portals and mirrors: Simple, fast evaluation of potentially visible sets. In *Proceedings of the 1995 symposium on interactive 3D graphics, Monterey (I3D), CA* (pp. 105–ff).
28. Scheuermann, T., & Hensley, J. (2007). Efficient histogram generation using scattering on GPUs. In *Proceedings of the 2007 symposium on interactive 3D graphics and games (I3D), Seattle, WA* (pp. 33–37).