This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JMMCT.2020.3044011, IEEE Journal on Multiscale and Multiphysics Computational Techniques

Non-Self-Adjacent Ray Classes for Parallelizable Shooting Bouncing Ray Tracing Double Count Removal

Cam Key, *Student Member, IEEE*, Blake A. Troksa, *Student Member, IEEE*, Stephen Kasdorf, and Branislav M. Notaroš, *Fellow, IEEE*

***Abstract*—We propose and experimentally validate a new ray spawning and associated double count removal (DCR) technique for shooting bouncing ray tracing (SBR). This technique allows, for the first time, efficient parallelization of ray DCR, the major bottleneck and least parallel aspect of modern SBR ray-tracing relying on the ray-cone approximation (RCA). We define non-self-adjacent (NSA) ray classes on a recursively sampled icosahedron, guaranteeing removal of mutual adjacency data dependencies between rays that previously prevented efficient parallelization of ray double count removal and, by extension, SBR. Using a GPU-parallelized implementation of the technique, we demonstrate speedups of DCR over 300×, limited in our testing only by the available hardware. As DCR is the asymptotically dominant contributor to the computation time of SBR-RCA, with respect to the number of parallel processes available, the achieved speedup applies to parallel SBR-RCA as a whole.**

***Index Terms*—Electromagnetic propagation modeling, asymptotic high-frequency techniques, high-performance computing, parallelization, GPUs, ray tracing method, shooting bouncing rays techniques, double count removal, ray-cone approximation, ray classes, large-scale simulations.**

I. Introduction

Ray tracing is an old and simple computational electromagnetics (CEM) technique that has seen renewed interest in recent years due to increased computing power and demand for fast propagation modeling in electrically-large, complicated environments. A frequency-asymptotic technique, ray tracing is well-suited to the types of propagation problems to which classical full-wave techniques like method of moments (MoM), finite difference (FD), and finite element method (FEM) are least suited. As such, ray tracing fills an important gap in the toolkit of methods available to CEM researchers and practitioners for diverse applications including 5G planning, propagation modeling in tunnel environments, and received signal strength (RSS) mapping [1]-[4]. As ray tracing is applied to a broader suite of increasingly demanding applications, the efficiency and scalability of the technique is now, more than ever, paramount to its usefulness.

For electrically-large propagation environments with linear, homogeneous media, ray tracing techniques have predominantly relied on image theory (IT) or the shooting-bouncing rays method (SBR); see [5] for an overview of these methods. In both cases, rather than explicitly solving variational formulations of Maxwell's equations and resulting linear systems, as full-wave techniques do, ray tracing iteratively constructs an approximate solution by propagating rays, each representing radiation from a source over a differential solid angle, and recording their interaction with the environment constrained as modeled by high frequency approximations like the Fresnel coefficients and theory of geometric optics (GO). For an excellent historical and theoretical background, see [6].

Image theory computes the paths rays follow from a source to a given receiver by recursively reflecting a source over all boundaries visible from that source to produce a set of image sources—each image source then treated as a new source. This process is continued to some maximum number of reflections, $N_{reflections}$, at which point any valid paths from source to receiver with up to $N_{reflections}$ reflections can be computed from the set of image sources—see [6] for a good overview. The advantage of this approach is that all possible paths between source and receiver with $N_{reflections}$ or fewer are captured exactly, reducing phase error. However, the computational complexity of IT is $O\left({N_{faces}}^{N_{reflections}}\right)$, where N_{faces} is the number of flat surfaces used to represent material discontinuities in the propagation environment. Since, for modern problems, N_{faces} is large, IT quickly becomes computationally untenable, even for small numbers of reflections. We note, however, that some techniques like reflection spaces or illumination zones can somewhat reduce the computational cost of IT.

SBR overcomes the computational shortcomings of IT by instead choosing a set of ray directions and a fixed number of rays a priori, then propagating each ray through the environment until it has made $N_{reflections}$ reflections or some other stop criterion is met, e.g., the ray leaving some region of interest. This yields linear complexity with the number of rays, and, using domain partitioning methods like the binary space partition (BSP), logarithmic complexity with respect to the

Manuscript received January 21, 2020, revised May 11, 2020. This work was supported by the National Science Foundation under grant ECCS-1646562.

Cam Key, Blake Troksa, Stephen Kasdorf, and Branislav M. Notaroš are with the Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO 80523-1373 USA (e-mail: camkey@rams.colostate.edu, blake.troksa@gmail.com, skasdorf@rams. colostate. edu, notaros@colostate.edu).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JMMCT.2020.3044011, IEEE Journal on Multiscale and Multiphysics Computational Techniques

number of facets [7]. However, this does not produce a set of exact paths between source point and receiver point, necessitating a method to decide which rays' field contributions should be counted at a given receiver. This is typically resolved by applying either the ray-cone approximation (RCA) [8]-[9] or ray-tube launching (RTL) [10]. Note that we only consider flat, triangular facets in this paper.

RTL has the advantage of exactly tiling the sphere of possible initial ray directions with no overlap. However, RTL introduces cases where ray tubes are split when only part of a tube's subtended solid angle reflects from a given face. Handling such cases introduces computational overhead and heavily-conditional execution, yielding a ray count that grows dynamically with reflection order in a way that cannot quickly be predicted a priori. RTL can therefore not be parallelized in an efficient, synchronous manner, making it a poor choice for modern SBR applications where scalability on synchronous, parallel hardware like graphics processing units (GPUs) is critical [11].

RCA also suffers from a barrier to efficient and complete parallelization: double count removal (DCR). DCR is necessary when using RCA due to inherent overlap between ray cones in the three-dimensional (3D) domain [11]. If a receiver point falls within the overlap of two cones from the same source, the field contribution from that source may be counted twice, leading to significant error in the resulting received power [11]. This necessitates a method to either prevent such cases a priori or detect them and remove them during computation. Many DCR approaches have been proposed in the past, but none have been developed with scalability on modern parallel hardware in mind. In [12], the authors present a DCR method by which rays are described by a characteristic sequence of planes hit, such that two rays with the same characteristic sequence when arriving at the same receiver are duplicates, necessitating the removal of one. This requires a comparison of characteristic sequences between all rays that arrive at a receiver to detect identical characteristic sequences—leading to a worst case complexity of $O\left({N_{rays}}^2\right)$ and producing a mutual data-dependency between rays that prevents effective parallelization. This also suffers from additional computational overhead where multiple coplanar, adjacent facets are present and therefore need to be tracked as the same object to maintain uniqueness of the characteristic sequence of a unique ray. A similar method is described in [13] that relies on information about each ray's number of reflections, distance traveled, and angle of transmission to detect and remove double counts. This is essentially a continuous version of the characteristic sequence from [12], with which we identify ray paths by continuous-valued properties of their propagation paths rather than discrete indices. The method in [13] suffers from the same mutual data-dependency between rays that hinders the method in [12] from effective parallelization. The most common type of DCR is described well in [14], which uses explicit geometric calculations to determine if two rays that have arrived at the same reception sphere contain the reception point in the overlap of their ray cones, indicating a double count. This approach is fast and reliable for sequential execution, but, as with previous methods, suffers from a mutual data dependency between rays that hinders its parallel performance and scalability. A useful structured sampling method is described in [8] that constrains the number of neighboring rays for any given ray, limiting double count checks to a known set of neighbor rays by sampling recursively on the icosahedron. This is useful to reduce the worst-case complexity of DCR to $O\left(N_{rays}\right)$. However, the DCR method described in [8] still introduces a mutual data dependency between neighboring rays that prevents efficient parallelization. We elaborate on what we mean by a data dependency and why it makes efficient parallelization difficult in Section V.C.

This paper proposes an efficient method of double count removal in SBR ray tracing that is highly parallelizable and removes the last major bottleneck to efficient parallel scaling of SBR applied to CEM. We take a similar sampling approach as [8] to limit potential double counts for each ray to a set of known neighbor candidates and maintain an $O\left(N_{rays}\right)$ worst case run time, but introduce a new DCR method that does not suffer from the mutual data dependency between rays that prevents effective parallelization of previous DCR methods. We introduce non-self-adjacent (NSA) classes of rays on the structured icosahedral and octahedral samplings such that no two rays in the same class are neighbors. When only one NSA class is processed at a time, no ray has mutual data dependency with any other ray currently being processed, removing the major barrier presented by previous methods to effective parallelization of SBR. Due to the structure of the sampling we use and the way we define the NSA classes, information from at most six neighbor rays needs to be checked for double counting at the time a given ray is processed. The number of neighbor rays that need to be checked and their indices is known a priori for any ray. The NSA classes we introduce have useful properties like symmetry, asymptotic inter-class isotropy, and simple definition yielding easy implementation. We present a four-class NSA formulation on the icosahedron, maintaining complete non-self-adjacency at the minor expense of inter-class isotropy. We also present two three-class NSA formulations: one on the icosahedron maintaining inter-class isotropy but only asymptotic NSA, and one on the octahedron, maintaining inter-class isotropy and full NSA at the expense of decreased global sampling regularity.

In the rest of the paper, we introduce these NSA classes and associated DCR methodology. We begin with a review of the icosahedral sampling technique, followed by a description of the introduced NSA classes, along with their definition, useful properties, and relative advantages. We next discuss application of the introduced NSA classes to highly-scalable DCR, offering a theoretical discussion of the asymptotic correctness of our simple DCR method in terms of sampling the SBR image space. We introduce the image space with motivating examples to facilitate this theoretical discussion. We then present speedup, computation time, and scaling results demonstrating efficacy of the proposed method, achieving over 300× speedup. We conclude by further outlining the potential of the new DCR technique using NSA classes for efficient and scalable SBR.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JMMCT.2020.3044011, IEEE Journal on Multiscale and Multiphysics Computational Techniques

II. Mathematical Definitions

To facilitate simple discussion of NSA classes, we make a variety of useful definitions and assumptions while noting a few important consequences. We begin denoting by R the set of all rays to be processed and by $r_i \in R$ the i^{th} ray in R. We assume here that each ray is unique, or formally that $r_i \neq r_j, i \neq j$. We then define the total number of rays, $N_{rays} = |R|$. We also denote by K the set of ray classes, and by $C_i \in K$ a specific ray class. A ray class is a set of rays. The total number of classes is $N_{classes} = |K|$. For all classes, we enforce completeness $\bigcup_{i=1}^{N_{classes}} C_i = R$, and independence $C_i \cap C_j = \emptyset, i \neq j$. In general, we denote the neighborhood (set of neighbors) of r_i as N_i with only the constraint that $r_i \notin N_i$. The most useful choice of N_i for RCA is the set of spherical Voronoi neighbors of r_i, denoted here V_i. However, we maintain generality in the choice of neighbors wherever we use N_i. We formally define the NSA property as

$$\{r_i | r_i \in N_j, r_i \in C_k, r_j \in C_k\} = \emptyset, \ \forall C_k \in K, \tag{1}$$

and similarly, the asymptotic NSA property as

$$\lim_{N_{rays}\to\infty} \frac{|\{r_i | r_i \in N_j, r_i \in C_k, r_j \in C_k\}|}{|\{r_i | r_i \in C_k\}|} = 0, \ \forall C_k \in K. \tag{2}$$

III. Non-Self-Adjacent Ray Classes

NSA ray classes are those that satisfy (1). Structured DCR methods like ours or [8] limit the DCR data dependency to a known neighbor set. For such DCR methods, ray classes that satisfy (1) guarantee that no rays within a given class are dependent, allowing all members of a class to be processed in parallel. Any ray class can satisfy (1) with the correct neighbor sets, most simply and least usefully $N_i = \emptyset, \forall r_i \in R$. In competition with this, the specific choice $N_i = V_i, \forall r_i \in R$ is geometrically correct for detecting SBR double counts but constrains the possible classes that satisfy (1) for a given sampling pattern. Satisfaction of (2) gives an easy solution to this problem. For ray classes with $N_i = V_i, \forall r_i \in R$ that satisfy (2) but not (1), we can ignore possible double counts between neighbors in the same class, allowing members of the same class to be processed in parallel while introducing only minimal error. In practice, this is done by excluding from the neighbor set of a ray any Voronoi neighbors that share the class of that ray. Because of this, ray classes based on $N_i = V_i, \forall r_i \in R$ that satisfy only (2) are almost as useful as those that satisfy (1). We show three useful ray class definitions based on $N_i = V_i, \forall r_i \in R$ that satisfy (2) or (1).

Among the possible methods to define NSA ray classes for SBR, the simplest is to assign each ray to its own class. Since, for removal of the SBR DCR data dependency, we require ray classes to be processed sequentially, assignment of each ray to its own class is equivalent to fully sequential SBR. This may seem trivial but reveals an important consideration for the number of rays per class: if the number of rays per class is less than the number of rays our given hardware can process in parallel, then ray classing presents a computational bottleneck. To maximize the minimum value of N_{rays} for which this bottleneck occurs, it is desirable to choose the minimum number of ray classes possible—the fewer ray classes, the more rays per class. In choosing the minimum number of ray classes, it is easy to see that neither one class nor two classes can give us the necessary NSA property. For $N_{classes} = 1$, (1) is not satisfied unless $N_i = \emptyset, \forall r_i \in R$, otherwise a ray and its neighbors are in the same class. For $N_{classes} = 2$, (1) is not satisfied unless neighbors of a ray are themselves never neighbors, or in other words, the graph, G, constructed by connecting each $r_i \in R$ to its neighbors contains no topological triangles. G with no topological triangles can exist in general, but for the most useful case of $N_i = V_i, \forall r_i \in R$, G is the spherical Delaunay triangulation of R, which contains *only* triangles for $N_{rays} > 2$.

For (1) to hold when $N_i = V_i, \forall r_i \in R$, neighbors of any $r_i \in R$ cannot be in the same class as r_i and no adjacent neighbors can be in the same class as each other. This requires, at minimum, $N_{classes} = 3$ to fully satisfy (1). Since V_i lie on a topological circle around r_i we require

$$|V_i| \ mod \ 2 = 0, \forall r_i \in R, \tag{3}$$

or in other words, an even number of neighbors for each ray. A few regular neighborhoods with varying neighbor counts are shown in Fig. 1.

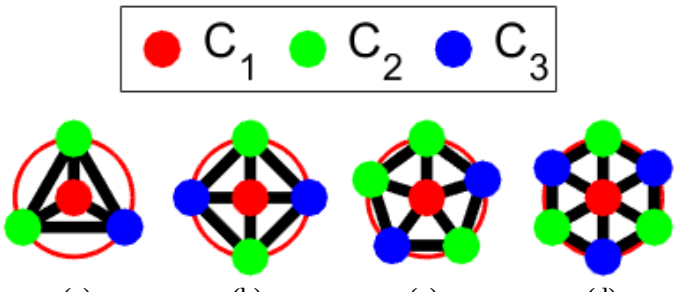

Fig. 1. Examples of uniform local topology with three classes: (a) triangular, (b) square, (c) pentagonal, and (d) hexagonal. Neighbors of central cell lie on a topological circle.

For global sampling uniformity, we desire the Voronoi cells of all rays to be identical, regular polygons. In the Euclidean plane, we could simply tile with either squares (Fig. 1b) or regular hexagons (Fig. 1d) and the class patterns from Fig. 1 to satisfy sampling uniformity and (1). However, satisfying sampling uniformity on the sphere is only possible for the five platonic solids, offering at most 20 sample points in the case of the dodecahedron (sampled on vertices) or the icosahedron (sampled on face centroids). This motivates methods like the icosahedral subdivision approach in [8] that, more generally speaking, sample at the vertices of high-frequency geodesic polyhedra to maximize sampling uniformity in a structured way.

A. Three Classes in Icosahedral Topology

Unfortunately, geodesic polyhedra with icosahedral

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JMMCT.2020.3044011, IEEE Journal on Multiscale and Multiphysics Computational Techniques

symmetry never emit a topology that can satisfy (1) with $N_{classes} = 3$; they contain 12 vertices with $N_i = 5$, necessitating $N_i \cap N_j \neq \emptyset, i \neq j$ in some cases. However, this defect need only occur at the edges of the original (pre-subdivision) icosahedron. The number of samples that lie on the original icosahedral edges grows linearly with the number of subdivisions while the total number of sample points grows quadratically. The asymptotic NSA property (2) is therefore satisfied with $N_{classes} = 3$. We show a simple method here.

We can easily define a set of possible sample points on any triangle and three associated classes that satisfy (1) when only points on that triangle, t, are considered. We denote by $\{a_t, b_t, c_t\}$ the set of vertex locations of the triangle, and by $N_{divisions}$ the desired number of subdivisions (an edge of t is split into $N_{divisions}$ new edges). Each sample point on the triangle is then given by

$$s_{i,j}^t = a_t + \frac{i}{N_{divisions}}(b_t - a_t) + \frac{j}{N_{divisions}}(c_t - a_t), \tag{4}$$

with indices defined by

$$i, j \in \mathbb{N}_0, i, j \leq N_{divisions} + 1, i + j \leq N_{divisions} + 1. \tag{5}$$

The classes on t are then given by

$$C_k^t = \{s_{i,j}^t | (j - i)\ mod\ 3 = k - 1\}, k \in \{1,2,3\}. \tag{6}$$

Figure 2 shows these classes on a triangle for $N_{divisions} = 11$.

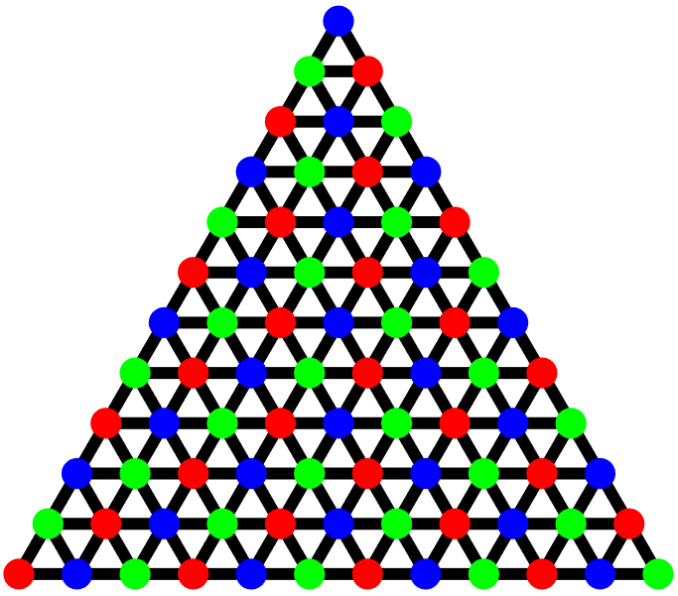

Fig. 2. NSA classes defined by (6) for $N_{divisions} = 11$.

Since (6) may assign different classes to a point lying on an edge depending on which adjacent triangle we consider, we require an extra step to maintain class independence $C_i \cap C_j = \emptyset, i \neq j$ and expand classes from (6) to the entirety of a geodesic polyhedron by a union of (6) over its triangles.

We denote by G the set of edges, by P the set of vertices, and by T the set of triangular facets of an arbitrary polyhedron with triangular faces. Each vertex $p \in P$ has a set of incident edges. We specify that a given p is a member of only one of its incident edges. Similarly, each edge $g \in G$ separates two triangular faces. We specify that points on a given g are a member of only one of the two triangles it separates. By these definitions, each point on the geodesic polyhedron is a member of one and only one $t \in T$. If $s_{i,j}^t \in t$, we say t is the parent triangle of $s_{i,j}^t$. The parent triangle of any sample point is unique. The classes on the geodesic polyhedron are then given by

$$C_k = \bigcup_{t \in T} C_k^t, \tag{7}$$

where indices are as defined in (5). Figure 3 shows these classes on the icosahedron with $N_{divisions} = 13$.

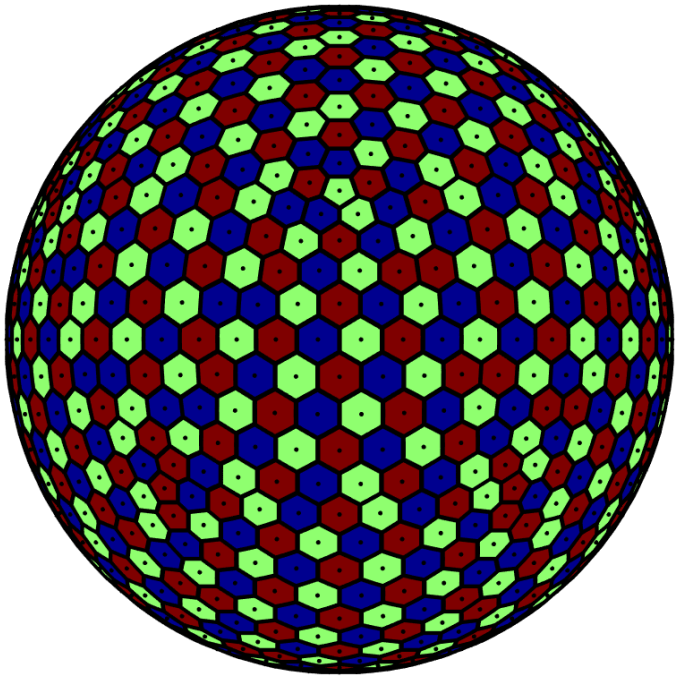

Fig. 3. Asymptotically-NSA classes on the icosahedron with $N_{divisions} = 13$ and $N_{classes} = 3$. Voronoi-adjacent rays in the same class are common here due to very low ray count.

This choice of classes and sample points offers an excellent foundation for parallel SBR-DCR. Although only (2) is satisfied, the points that violate (1) are constrained to those lying on the edges of the original icosahedron and their immediate neighbors. These points represent a proportion of the total N_{rays} that decreases linearly with increased $N_{divisions}$. This makes double counting between adjacent same-class neighbors inconsequential at the high ray counts typically used in most SBR applications.

B. Three Classes in Octahedral Topology

The presence of twelve points with five neighbors on geodesic polyhedra with icosahedral topology prevents such

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JMMCT.2020.3044011, IEEE Journal on Multiscale and Multiphysics Computational Techniques

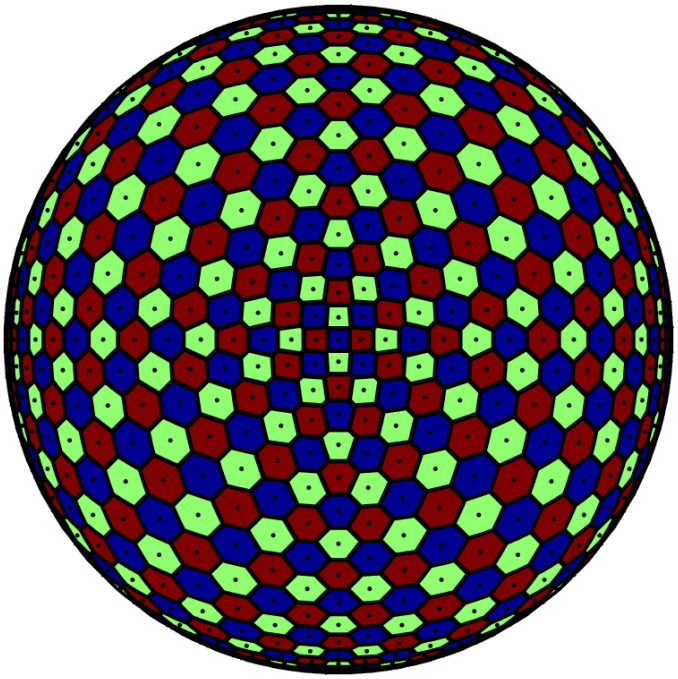

Fig. 4. Perfect NSA classes on the icosahedron with $N_{divisions} = 17$ and $N_{classes} = 3$. No Voronoi-adjacent rays are in the same class.

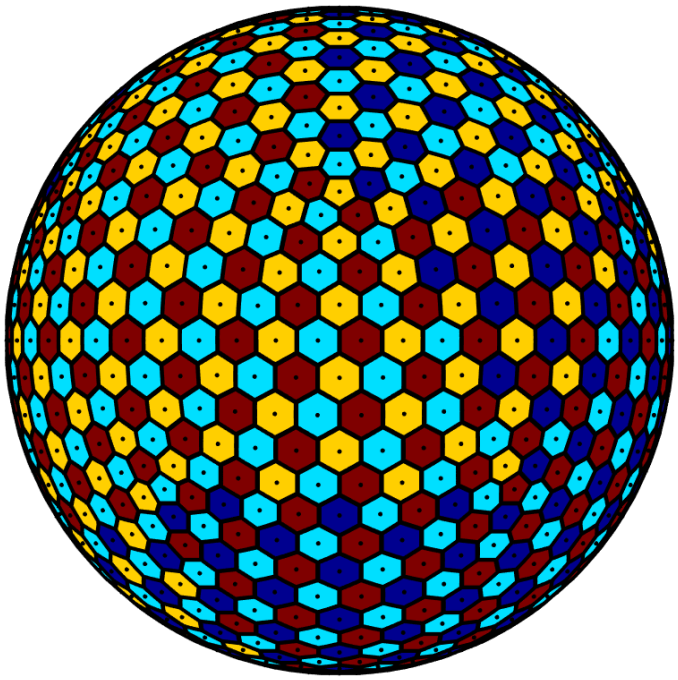

Fig. 5. Perfect NSA classes on the icosahedron with $N_{divisions} = 13$ and $N_{classes} = 4$. No Voronoi-adjacent rays are in the same class.

polyhedra from satisfying (3) at all vertices. Octahedral geodesic polyhedra, on the other hand, contain only points that satisfy (3), making them a good option for sampling where perfect non-self-adjacency is desired with the minimum number of classes.

Figure 4 shows (6) and (7) applied to the octahedron with $N_{classes} = 3$. We use an ordering of $\{a_t, b_t, c_t\}$ for each triangle that maintains class independence on edges between triangles regardless of parent triangle assignment. Many such orderings exist on the octahedron, so we do not specify one here. These octahedral classes have the advantage of fully satisfying (1) with only three classes, but at the cost of somewhat reduced sample uniformity compared to the icosahedron.

C. Four Classes in Icosahedral Topology

To achieve both high sampling uniformity and satisfaction of (1), we can define fully NSA classes on the icosahedron with $N_{classes} = 4$. We choose one of many four-color vertex colorings of the icosahedron, with colors corresponding to class indices $k \in \{1,2,3,4\}$. This assigns to each of the icosahedron's twelve vertices one of four classes, such that no adjacent vertices share a class.

On a given triangle, we again denote by $\{a_t, b_t, c_t\}$ the set of vertex locations, and now by $\{k_{a_t}, k_{b_t}, k_{c_t}\}$ the set of corresponding class indices. For simplicity, we define the vector $h = \langle k_{a_t}, k_{b_t}, k_{c_t} \rangle$, with $h(l)$ denoting its l^{th} entry. Sample points are again defined by (4) with indices defined by (5). However, instead of (6), the classes on t are now given by

$$C^t_{h(l)} = \{s^t_{i,j} | (j - i)\ mod\ 3 = h(l) - 1\}, l \in \{1,2,3\}. \quad (8)$$

Assigning parent triangles as before to maintain class independence, the four classes on the icosahedral geodesic polyhedron are again given by (7). Note that $N_{divisions}$ must be one less than an integer multiple of 3 to maintain class independence. Figure 5 shows (7) and (8) applied to the icosahedron with $N_{divisions} = 13$ and $N_{classes} = 4$. These classes fully satisfy (1) and have the same sampling uniformity as those from Fig. 3, but each class no longer samples the entire sphere.

IV. SBR as a Sampling of the Image Space

To facilitate a discussion of SBR DCR, we introduce the concepts of the environment space and image space. The environment space, E, is the physical space in which we are modeling propagation. Any point $e \in E$ is given by a scalar real-valued triplet with spherical coordinates $\langle \rho, \theta, \phi \rangle, \rho \in [0, \infty)$, $\theta \in [0, \pi]$, $\phi \in (-\pi, \pi]$. A ray, r, with initial direction $\langle \theta_0, \phi_0 \rangle$ follows a curve, s, through E, parametrized by d such that $s(d) = e$ is the point on s at which the ray has traveled d distance along s. The curve s is a straight line radiating from the origin if no reflections occur, a continuous path composed of line segments of reflections occur in homogeneous media, or a general continuous, curved path in inhomogeneous media. We consider only the first two cases here. The image space, Q, represents the space in which paths taken by rays follow straight lines radiating from the origin regardless of their reflections in the environment space. Note that we consider only reflections here, not transmission. Any point $q \in Q$ is also given by a real-valued triplet with spherical coordinates $\langle d, \theta_0, \phi_0 \rangle, d \in [0, D_{max}]$, $\theta \in [0, 2\pi]$, $\phi \in (-\pi, \pi]$, where d is the distance traveled in E for the ray with initial direction $\langle \theta_0, \phi_0 \rangle$. D_{max} gives the maximum propagation distance considered. The ray source is the origin

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JMMCT.2020.3044011, IEEE Journal on Multiscale and Multiphysics Computational Techniques

of both spaces. We define a map M such that $M(q) = e = s(d)$. Note that M is in general not invertible: $\exists e \in E\ s.t.\{q|M(q) = e\} = \emptyset$; we call such e occluded. Note that $E = Q$ in homogeneous media with no reflections and $D_{max} = \infty$. Note also that the concepts of E and Q apply to SBR as a whole and are not specific to the NSA or DCR methods we present in this paper.

For clarity, we give a few examples, shown in Fig. 6, in two dimensions of E and the associated Q. To produce these plots, we constrained $\phi = 0$ and uniformly distributed initial ray directions in θ. Since a given ray only samples E along a given path, in turn sampling Q only along a straight radial path from the origin, we interpolate Q between rays using RCA and assigning any q not on a ray path the properties of the nearest q on a ray path. This produces a piecewise approximation of M: $\widetilde{M}$. Rays were propagated for a fixed, constant distance. We chose $N_{rays} = 1000$ so no defects due to the $\widetilde{M}$ approximation are visible at the chosen figure resolution and propagation distance. To demonstrate the relationship between E and Q, we assign hues to $e \in E$ corresponding to θ and opacity increasing with ρ. Each $q \in Q$ is then assigned the hue and opacity of $\widetilde{M}(q)$.

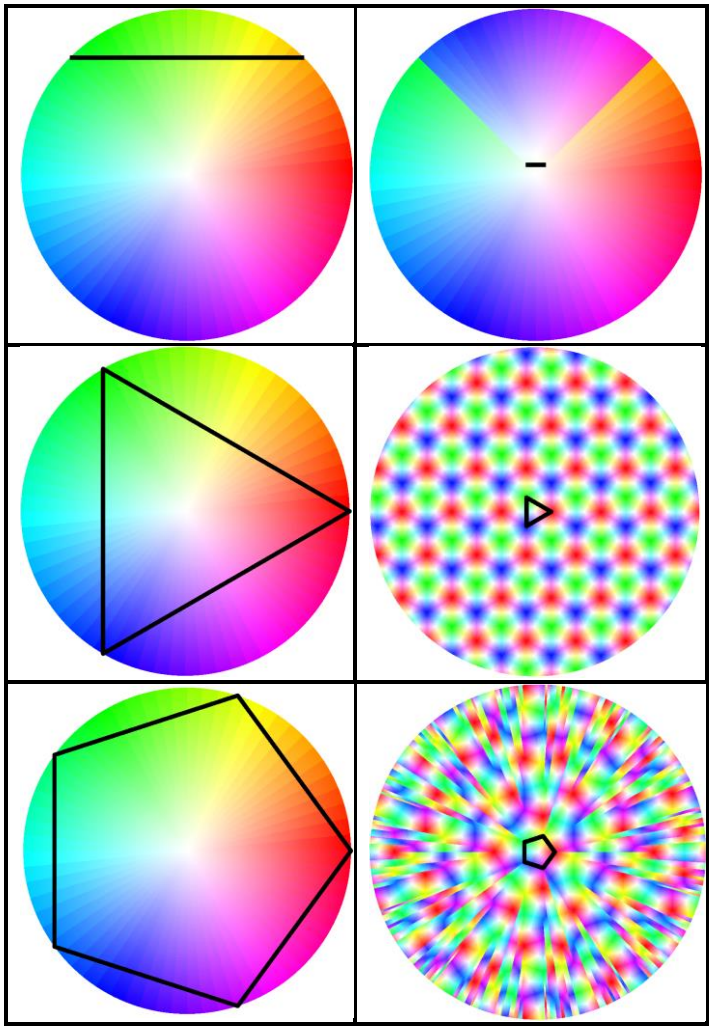

Fig. 6. Examples of environment and associated image spaces. Reflectors are shown in black. The left column shows three examples of environment spaces: a single plane, a triangle, and a pentagon. The right column shows the associated image spaces.

We quantify the geometric error introduced by approximating the mapping M as

$$\varepsilon_{geometric} = \frac{3}{4\pi D_{max}^4} \int_Q \left\| M(q) - \widetilde{M}(q) \right\| dq. \tag{9}$$

Note that the 4th rather than 3rd power in (9) comes from normalizing with respect to D_{max} in addition to the volume of integration. We define geometric convergence of SBR as the property that

$$\lim_{N_{rays} \to \infty} \varepsilon_{geometric} = 0. \tag{10}$$

SBR has geometric convergence if, for an arbitrary region, Ω, in θ, ϕ on the sample sphere surface,

$$\lim_{N_{rays} \to \infty} |\{r|\langle \theta_0, \phi_0 \rangle \in \Omega\}| = \infty. \tag{11}$$

It is easy to show from (4) that the sampling patterns in Section III enforce (11) and therefore (10).

V. Efficient, Parallel Double Count Removal

A. The Proposed Method

To present our DCR method, we first make some definitions for clarity. We have a set of observation points O with $o \in E, \forall o \in O$ and $N_{observations} = |O|$. We denote by $N_{reflections}$ the maximum number of reflections considered for all rays. The goal of SBR is to compute the field at all observation points due to a set of source points. We consider only one source point at a time, combining fields at observation points by superposition when multiple source points are present. To compute from which rays contributions are considered at a given $o \in O$, we use the dynamically-sized sphere intersection method from [14]. We choose α from [14] for a given $r_i \in R$ as the maximum angle between r_i and any $r_j \in V_i$. This prevents any gaps between ray cones, allowing errors only in the form of overlap (double counts) between neighbors.

Our DCR technique is more straightforward than those in [8]-[14] and can be summarized simply when implementation details are ignored: We process only one ray class and only the n^{th} reflection for rays in that class at a time, recording any ray-observation pairs for sphere intersections that occur between the n^{th} and $(n+1)^{th}$ reflection. We only keep a ray-observation pair containing r_i and o_j if no neighbors of r_i are members of pairs containing o_j. Note that, for NSA classes like those in Section III.A that satisfy (2) but not (1), we do not consider Voronoi-neighboring rays in the same class as neighbors for the purpose of DCR. This introduces an error that is asymptotically negligible, as discussed in Section III.A.

This DCR method is extremely simple, and with the NSA classes from Section III, highly parallelizable and scalable.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JMMCT.2020.3044011, IEEE Journal on Multiscale and Multiphysics Computational Techniques

We give a comparison to existing DCR methods as well as a pseudocode example for one possible implementation in Section V.C. First, however, we consider the glaring omission we make in defining such a simple method. Our method introduces an obvious error that previous methods have mitigated with more-complicated techniques. Where neighboring rays hit different, non-coplanar facets but intersect the same observation sphere with the same reflection count, our method will detect a false double-count not detected by more-rigorous DCR methods. A simple example of the type of false double count detected by our method is shown in Fig. 7. If rays r_1 and r_2 are neighbors, only one of their field contributions will be counted at o after having reflected one time, even though these reflections were from different, non-coplanar facets. The field contributions of r_1 and r_2 in this case represent different image sources, so both should be counted.

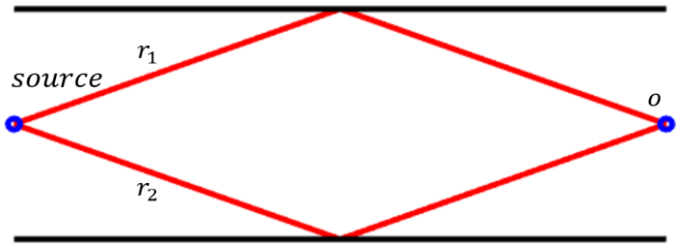

Fig. 7. A simple example of how a false double count may be detected by our method. If the two rays are neighbors, our method will count only one of their contributions at o, despite the rays representing unique images.

Production of false double counts, superficially, seem like a major flaw with our DCR method. However, we show here that the proportion of false double counts drops asymptotically to 0 with increased N_{rays}, and by extension the asymptotic correctness of our DCR method.

B. Asymptotic Correctness of the Proposed Method

To show asymptotic correctness of our method, it suffices to show the proportion of neighboring rays that hit the same triangular facets in the same order after having traveled some finite maximum distance D_{max} approaches 1 asymptotically as $N_{rays} \to \infty$. Satisfaction of this property can be shown using the notion of the image space as follows.

Denote by EG the set of points in E on facet edges and $QG = \{q | M(q) \in EG\}$. Projecting QG in the d direction onto the unit sphere gives QG'. If the domain contains a finite number of reflecting facets and D_{max} is finite, QG' partitions the unit sphere into a finite number of polygonal regions. Note that these partitions correspond to the largest possible polygonal cone boundaries of ray tubes in RTL after splitting if rays are only traced until D_{max}. By (11) and the observation that the region boundaries subtend only an infinitesimal solid angle, the proportion of neighboring rays that hit the same triangular facets in the same order by their D_{max} approaches 1 and our DCR method introduces a proportion of false double counts that decreases to 0 as $N_{rays} \to \infty$.

Convergence is also apparent from (11) and the perspective of image theory. Since (11) implies the solid angle subtended by each ray cone decreases asymptotically toward zero as $N_{rays} \to \infty$, the probability of neighboring rays hitting different facets at their first reflection (necessary but not sufficient for a false double count) also decreases toward zero. In the case where rays hit the same facet at their first reflection, the resulting reflected rays can be considered to radiate from the same image source. The rays' second reflection can then be treated as a first reflection, yielding an inductive proof of convergence for arbitrary reflection count or D_{max}.

C. Pseudocode and Comparison to Existing DCR Methods

To understand our DCR approach with NSA classes and why it allows for efficient parallelization, it is useful to understand why existing approaches make this more difficult. Generically speaking, existing DCR methods attempt to apply some function $DCR(r_i, o, c)$, to determine whether a ray causes a double count at a given observation o for a given context c and, if so, resolve that double count in some data structure that tracks ray-observation intersections. All ray-observation intersections remaining after DCR are counted in the final field computations for the corresponding observation. Our method is no different in this regard. Consider the case, however, where r_1, r_2, and r_3 are mutual neighbors; i.e. $r_1, r_2 \in N_3, r_1, r_3 \in N_2, r_2, r_3 \in N_1$. As noted in Section III, the correct neighbor choice $N_i = V_i, \forall r_i \in R$ yields the spherical Delaunay triangulation for G, so cases like this occur for every ray regardless of the sampling method chosen. Consider also that o falls in the overlap of all three rays' cones. Only one of these rays should be counted, although each of the three is equally valid under RCA. For a given context, we have three potential instances of DCR to process: $DCR(r_1, o, c)$, $DCR(r_2, o, c)$, and $DCR(r_3, o, c)$. To correctly resolve this situation by counting only one of the three rays, the outputs of the three processes must be consistent, e.g. if r_1 is counted, r_2 and r_3 cannot be counted. This requires that, for instance, computation of $DCR(r_2, o, c)$ and $DCR(r_3, o, c)$ is dependent on the result of $DCR(r_1, o, c)$, so the three processes cannot complete execution simultaneously. This is equally true if DCR constitutes a simple comparison of characteristic sequences [12] as it is for geometric computations between rays [14]. The problem lies in how synchronization mechanisms like mutexes that allow such data dependencies to be handled in a parallel execution environment delay process completion; a given thread must wait for others on which it is dependent. Such parallelization approaches are inefficient, since processor cycles are wasted while waiting, or, in more complicated approaches, while switching between threads.

To further illustrate the problem presented by adjacent ray data dependencies, we give below two examples of pseudocode, one for our DCR approach with NSA classes, and another for a generic DCR approach without NSA classes. For both examples, we assume that neighbor sets are defined and known ahead of time, as in our method or e.g. [8], since this is already a common approach in recent literature to limit the data dependency to only a small neighbor set. We also assume that indices of observations intersected by a given ray, r_i, are recorded in a hitlist denoted HL_i. We denote a generic observation point index as idx. There are many ways to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JMMCT.2020.3044011, IEEE Journal on Multiscale and Multiphysics Computational Techniques

manage this information, but we consider this the simplest and most illustrative.

```
1 for k ∈ [1..N_classes] do
2   for i ∈ [1..|C_k|] do in parallel
3     for r_j ∈ N_i do
4       conflicts ← {idx|idx ∈ HL_i, idx ∈ HL_j}
5       for idx ∈ conflicts do
6         HL_i ← HL_i\{idx}
```

Using NSA classes as above, only non-neighboring rays are processed in parallel, so $conflicts$ can be readily computed for each parallel instance. Note that lines 3 through 6 are effectively an implementation of $DCR(r_i, o, c)$. Consider, in contrast, if we use no NSA classes. We must somehow resolve cases like $r_1, r_2 \in N_3$, $r_1, r_3 \in N_2$, $r_2, r_3 \in N_1$. One way to do this could be:

```
1 for i ∈ [1..N_rays] do in parallel
2   for r_j ∈ N_i do
3     if i > 1 then
3       while not available_j do
4         wait
5     conflicts ← {idx|idx ∈ HL_i, idx ∈ HL_j}
6     for idx ∈ conflicts do
7       HL_i ← HL_i\{idx}
8 available_i ← true
```

Rays processed in parallel may now be dependent on each other, so we define the Boolean variable $available_j$ to keep track of whether double count removal has been completed for r_j. Here, lines 2 through 8 are effectively an implementation of $DCR(r_i, o, c)$. The process for r_1 can execute immediately, but other threads must wait until their dependencies are resolved. In fact, in this implementation, most threads will spend most of the total computation time waiting.

For simplicity, both examples intentionally ignore context. This is appropriate for our DCR method, but not for existing methods. Context is information other than ray index and observation index that identifies a unique field contribution. For instance, in [12], c is a characteristic sequence of facets hit by a ray before registering a hit for an observation. To consider context in general, updates to $conflicts$ would require comparison of e.g. characteristic sequences [12] or geometry information [14], adding a layer of complexity and reducing performance. Our DCR method avoids this by defining c as the number of reflections taken by a ray before encountering an observation sphere. We then only calculate hits for the n^{th} reflection of all rays simultaneously, resetting hitlists before the $(n+1)^{\text{th}}$ reflection. Since all entries in HL_i correspond to the same c, our method allows c to be ignored during DCR. A simple example of how we order our DCR method relative to other SBR processes is presented below.

```
1 for n ∈ [1..N_reflections] do
2   initialize HL_i to empty ∀i ∈ [1..N_rays]
3   for k ∈ [1..N_classes] do
4     for i ∈ [1..|C_k|] do in parallel
5       trace r_i to n^th reflection
6       compute sphere intersections
7       fill HL_i
8   DCR
```

VI. Results and Discussion

To demonstrate the scalability of the proposed NSA class-based parallel DCR method and its practical advantages over inherently sequential approaches, we produced an efficient GPU-based implementation. As a baseline, we also produced an efficient but fully sequential CPU-based implementation of the method. The CPU-based implementation performs the same operations from Section V, but processes only one ray at a time, rather than rays in each class in parallel. Both implementations used the same parallel, GPU-based SBR ray propagation, sphere intersection, and field computation implementations, the computation times of which were included in the total computation time. We show results using the 3- and 4-class icosahedral schemes from Section III, denoted Ico3 and Ico4, respectively. Our intention in presenting results for both Ico3 and Ico4 here is to demonstrate the bottleneck introduced by NSA classes does not occur for 3-class or 4-class schemes over the range of typical parameters tested (as low as $N_{rays} = 10^3$). Note that, since our DCR technique requires fewer operations to detect and handle double counts than existing methods, its use as a sequential benchmark here likely underestimates the computation time of most existing DCR approaches. Also note that, with good implementation, no pre-process step is required for management of ray class designations. Each ray's class can be determined in constant time from its parent triangle index and its indices within that triangle. All results were produced on a mid-range (as of 2019) consumer workstation equipped with an Intel i7-3770 3.4 GHz CPU and an NVIDIA GeForce GTX 1060 6GB GPU with 1280 CUDA cores. A 4×4×1000-meter waveguide was used as the propagation environment. Since our initial implementation is targeted to CUDA-enabled GPUs, we are not able to include a strong scaling plot (i.e., scaling with respect to core count) since threads are automatically distributed to GPU streaming multiprocessors in the CUDA paradigm, offering us little control over how many are used simultaneously. We hope to present a strong scaling plot in future work once we have an efficient CPU implementation.

Figure 8 shows the computation time taken only by DCR for both the sequential and parallel implementations with respect to increasing N_{rays}. We chose to test a wide range of N_{rays} values that we believe is representative of the range of ray counts used for most practical applications. We see vastly improved performance and scaling of parallel DCR over the sequential implementation, with parallel DCR outperforming sequential for all N_{rays} tested and a maximum observed speedup over 300×. We observe the largest speedups for the highest N_{rays} tested, with the speedup for lower ray counts likely constrained by host-device communication overhead below $N_{rays} = 10^6$.

Figure 9 shows the fraction of the total computation time taken by DCR for each approach with respect to N_{rays}. The sequential example takes roughly 50% of the total computation time by 100 million rays. The parallel examples, meanwhile, take less than 1% of the total time. Measuring the time proportionality of DCR is useful because it offers a simple, relative comparison of DCR to other important steps of the SBR algorithm. Encouragingly, the results of Fig. 9

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JMMCT.2020.3044011, IEEE Journal on Multiscale and Multiphysics Computational Techniques

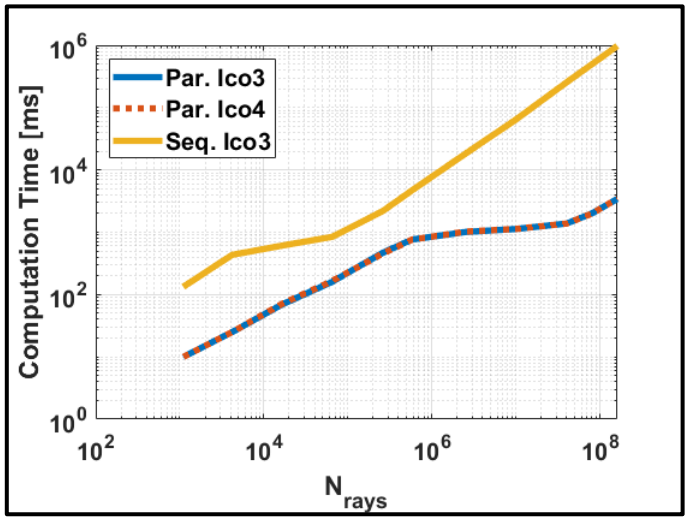

Fig. 8. Computation time of sequential vs. parallel DCR with respect to N_{rays}. Other parameters were constant: $N_{reflections} = 20$, $N_{observations} = 500$.

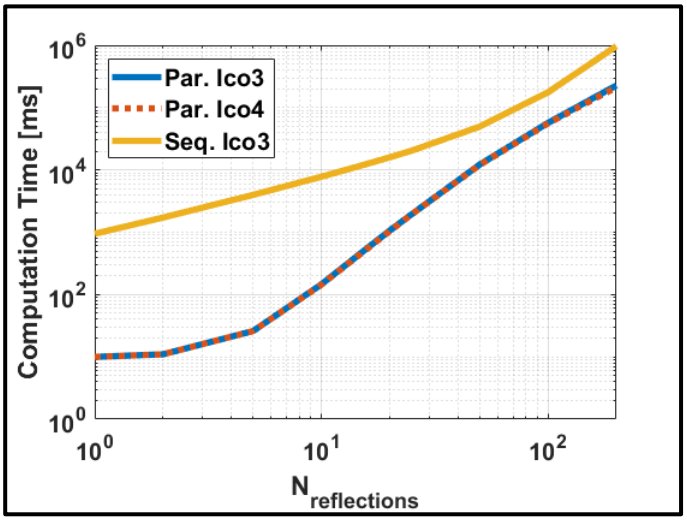

Fig. 10. Computation time of sequential vs. parallel DCR with respect to $N_{reflections}$. Other parameters were constant: $N_{rays} = 2{,}505{,}000$, $N_{observations} = 500$.

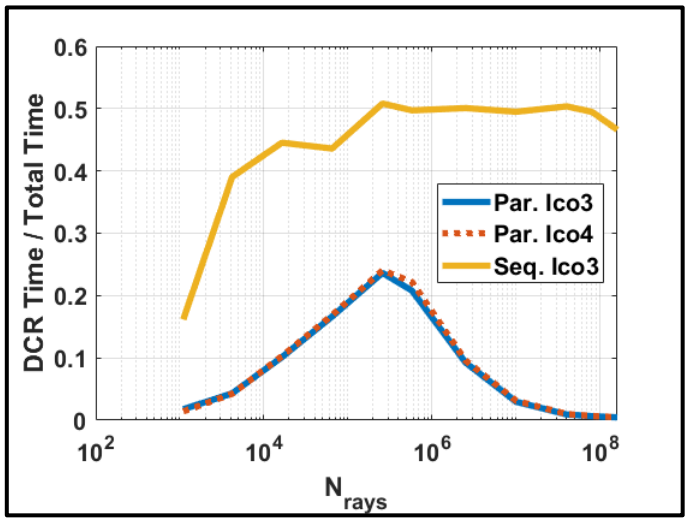

Fig. 9. Proportion of total SBR computation time taken by sequential vs. parallel DCR with respect to N_{rays}. Other parameters were constant: $N_{reflections} = 20$, $N_{observations} = 500$.

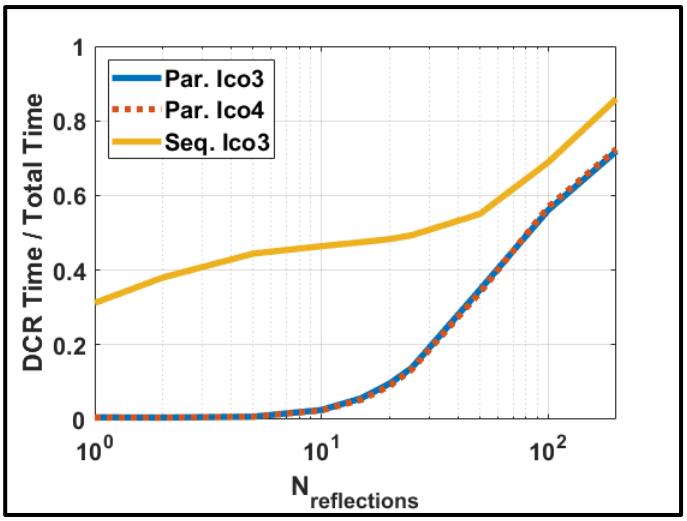

Fig. 11. Proportion of total SBR computation time taken by sequential vs. parallel DCR with respect to $N_{reflections}$. Other parameters were constant: $N_{rays} = 2{,}505{,}000$, $N_{observations} = 500$.

indicate that time taken by our parallel implementation is asymptotically non-dominant with respect to increasing ray count. Furthermore, the fact that Ico3 and Ico4 agree almost perfectly shows that no bottleneck is introduced by NSA ray classing over the wide range of ray counts tested.

We also note that the time proportionality peak in Fig. 9 around $N_{rays} = 10^6$ lends evidence to our belief that non-asymptotic effects like communication overhead constrain the speedup in Fig. 8 below this value.

Figure 10 shows similar results to figure 8, but with respect to the maximum number of reflections simulated for any given ray. We again chose a range of values that we consider typical for most practical applications. The parallel examples are once again faster in all cases, even at high reflection orders, with a maximum observed speedup over 100× for the parameter values tested. We note that the observed speedup becomes lower at higher reflection orders. We believe this is due to memory limitations of our GPU hardware at high reflection orders necessitating host-device communication.

Figure 11, analogous to Fig. 9, shows the proportion of the total computation time taken by each example. Although it appears in Fig. 11 that asymptotic behavior of the time proportionality has begun to dominate (we observe a linear trend on the semilog scale by around $N_{reflections} = 100$), this is unlikely to be the case. The proportion of the total time taken by DCR is limited to 1, so the observed trend is misleading (all three curves must level out at some point). As with Fig. 10, we believe the reduced efficiency at higher reflection orders can be attributed to memory limitations of our GPU hardware and associated host-device communication overhead.

Figures 12 and 13 are analogous to figures 8 and 9 but with respect to the number of field observation points. The parallel examples tested for Fig. 12 achieve a maximum observed speedup over 10,000×, although this is for very low $N_{observations}$. At high $N_{observations}$, the observed speedup levels out to about 10× on our test hardware. Like Figs. 10 and 11, we believe the reduced efficiency in Figs. 12 and 13 for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JMMCT.2020.3044011, IEEE Journal on Multiscale and Multiphysics Computational Techniques

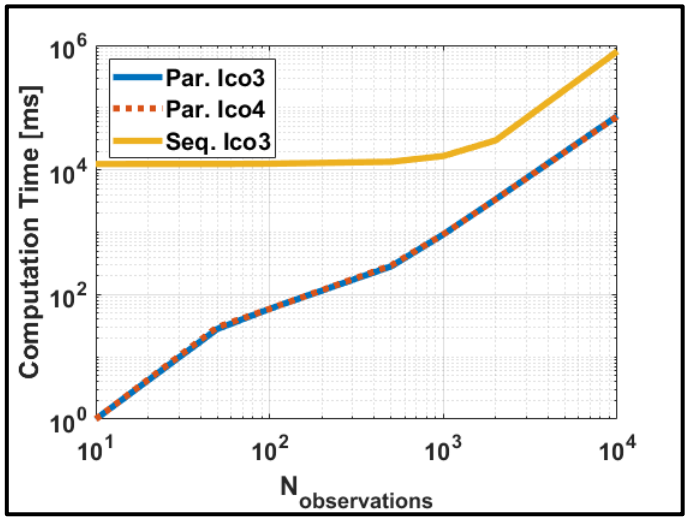

Fig. 12. Computation time of sequential vs. parallel DCR with respect to $N_{observations}$. Other parameters were constant: $N_{rays} = 2{,}505{,}000$, $N_{reflections} = 20$.

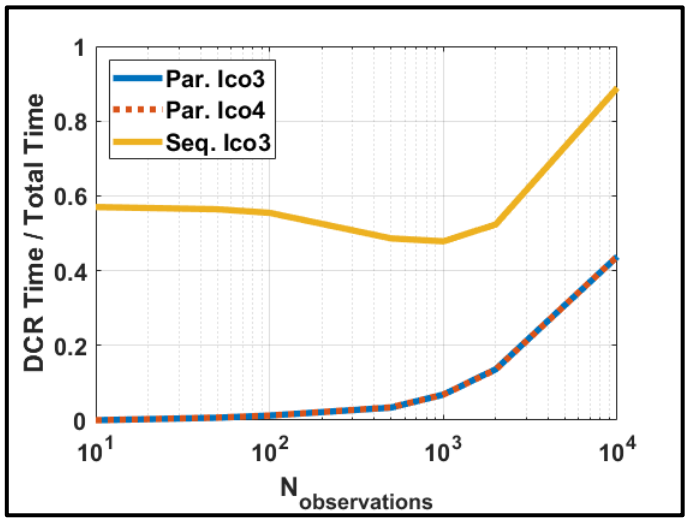

Fig. 13. Proportion of total SBR computation time taken by sequential vs. parallel DCR with respect to $N_{observations}$. Other parameters were constant: $N_{rays} = 2{,}505{,}000$, $N_{reflections} = 20$.

high $N_{observations}$ is due to host-device communication overhead.

VII. Conclusion

This paper has introduced non-self-adjacent ray classes for efficient, parallelizable shooting-bouncing-ray tracing double count removal. Unlike previous DCR methods, the approach made possible by the NSA ray classes introduced in this paper can take advantage of modern, parallel computing hardware, e.g., GPUs, that was not available in ray tracing's theoretical infancy. Predominantly geometric aspects of SBR like ray path computation and ray intersection tests have long been efficiently parallelizable, and most modern SBR approaches have taken advantage of this. However, the parallel approach to DCR enabled by the present work removes the last and final barrier to fully parallel, large-scale SBR simulations. This is crucial as problem sizes continue to grow, necessitating highly parallel and efficient CEM algorithms.

References

[1] F. Hossain, T. Geok, T. Rahman, M. Hindia, K. Dimyati, S. Ahmed, C. Tso, and N. Abd Rahman, "An efficient 3-D ray tracing method: prediction of indoor radio propagation at 28 GHz in 5G network," *MDPI Electronics*, vol. 8, no. 3, pp. 286-306, Mar. 2019.

[2] B. Troksa, C. Key, F. Kunkel, S. Savic, M. Ilic, and B. Notaros, "Ray tracing using shooting-bouncing technique to model mine tunnels: theory and verification for a PEC waveguide," *ACES Journal*, vol. 34, no. 2, Feb. 2019.

[3] N. Sood, L. Liang, S. V. Hum, and C. D. Sarris, "Ray-tracing based modeling of ultrawideband pulse propagation in railway tunnels", *Proc. IEEE APS/URSI Int. Symp.*, pp. 2383-2386, Jul. 2011.

[4] L. Kanaris, A. Kokkinis, M. Raspopoulos, A. Liotta, and S. Stavrou, "Improving RSS fingerprint-based localization using directional antennas," *Antennas and Propagation 8th European Conference*, pp. 2174-2177, Aug. 2014.

[5] M. Catedra and J. Perez, Cell Planning for Wireless Communications, Norwood, MA, USA: Artech House, 1999.

[6] D. McNamara, C. Pistorius, and J. Malherbe, Introduction to the Uniform Geometrical Theory of Diffraction, Norwood, MA, USA: Artech House, 1990.

[7] T. Ize, I. Wald, and S. Parker, "Ray tracing with the BSP tree," *Proc. IEEE Symp. on Interactive Ray Tracing*, pp. 159-166, 2008.

[8] S. Seidel, T. Rappaport, "Site-specific propagation prediction for wireless in-building personal communication system design," *IEEE Trans. on Veh. Technol.*, vol. 43, no. 4, pp. 879-891, Nov. 1994.

[9] S. Chen and S. Jeng, "An SBR/image approach for radio wave propagation in indoor environments with metallic furniture," *"IEEE Trans. on Antennas and Propag.*, vol. 45, no.1, pp. 98-106, Jan. 1997.

[10] C. Yang, B. Wu, and C. Ko, "A ray-tracing method for modeling indoor wave propagation and penetration," *IEEE Trans. on Antennas and Propag.*, vol. 46, no. 6, pp. 907-919, Jun. 1998.

[11] Z. Yun and M. Iskander, "Ray tracing for radio propagation modeling: Principles and applications," *IEEE Access*, vol. 3, pp. 1089-1100, Sep. 2015.

[12] Z. Yun, M Iskander, and Z. Zhang, "Development of a new shooting-and-bouncing ray (SBR) tracing method that avoid ray double counting," *IEEE APS Int. Symp. Dig.*, vol. 1, pp. 464-467, Jul. 2001.

[13] D. Didascalou, T. M. Schäfer, F. Weinmann, and W. Wiesbeck, "Ray density normalization for ray-optical wave propagation modeling in arbitrarily shaped tunnels", *IEEE Trans. Antennas Propag.*, vol. 48, no. 9, pp. 1316-1325, Sept. 2000.

[14] N. Noori, A. A. Shishegar, E. Jedari, "A New Double Counting Cancellation Technique for Three-Dimensional Ray Launching Method", *Proc. IEEE Antennas and Propag. Society Inter. Symp.*, pp. 2185-2188, 2006.

Cam Key (S'16) was born in Fort Collins, CO in 1996. He received his B.S. (2018) and his Ph.D. (2020) in Electrical and Computer Engineering from Colorado State University. His current research interests include uncertainty quantification, error prediction, and optimization for computational science and engineering; computational geometry, meshing, data science, machine learning, artificial intelligence, remote sensing and GIS, and novel applications of numerical methods across disciplines.

Blake Troksa was born in Boulder, CO in 1996. He received his B.S. (2018) and his M.S. (2019) in Electrical and Computer Engineering from Colorado State University. He is currently working as a software development engineer in the area of cloud computing. His research interests include hardware acceleration and distributed systems.

Stephen Kasdorf Received B.S. (magna cum laude) degrees in 2019 in both electrical engineering and applied physics from Colorado State University. He is currently working towards a PhD in electrical engineering at Colorado State University. His research interests include high frequency asymptotic electromagnetics methods such as ray optics and physical optics, as well as their hybridization with traditional numerical techniques.

Branislav M. Notaroš (M'00-SM'03-F'16) received the Dipl.Ing. (B.S.), M.S., and Ph.D. degrees in electrical engineering from the University of Belgrade, Belgrade, Yugoslavia, in 1988, 1992, and 1995, respectively. From 1996 to 1999, he was Assistant Professor in the School of Electrical Engineering at the University of Belgrade. He was Assistant and Associate Professor from 1999 to 2006 in the Department of Electrical and Computer Engineering at the University of Massachusetts Dartmouth. He is currently Professor of Electrical and Computer Engineering, University Distinguished Teaching Scholar, and Director of Electromagnetics Laboratory at Colorado State University. Dr. Notaroš serves as General Chair of the 2022 IEEE International Symposium on Antennas and Propagation and USNC-URSI National Radio Science Meeting and is Associate Editor for the IEEE Transactions on Antennas and Propagation. He serves as Vice President of Applied Computational Electromagnetics Society (ACES) and as Vice-Chair of USNC-URSI Commission B. He was the recipient of the 2005 IEEE MTT-S Microwave Prize, 1999 IEE Marconi Premium, 2019 ACES Technical Achievement Award, 2015 ASEE ECE Distinguished Educator Award, 2015 IEEE Undergraduate Teaching Award, and many other research and teaching international and national awards.