

Towards Hybrid Spatial Structure Rendering Models Using Quantum Orthographic Ray Casting

Zirui Zhang

Department of Computer Science
Purdue University
West Lafayette, USA
zhan4192@purdue.edu

Abstract—Ray tracing is a fundamental technique in computer graphics used to render realistic images by simulating light interactions with objects, known as primitives. With the rise in popularity of quantum computing, new techniques have been developed to accelerate these computationally intensive tasks, with the potential of significant speedup over classical approaches. Traditional ray tracing relies on detailed ray casting and intersection tests between rays and scene primitives, which can be extremely time-consuming for complex scenes. Grover's algorithm for unordered search has demonstrated potential quadratic speedups in unstructured search problems, creating novel approaches to accelerate intersection detection in graphics.

This work aims to contribute to the practicality of quantum search techniques for hybrid ray tracing pipelines to accelerate intersection tests. We propose two hybrid approaches: a uniform spatial division method employing a 3D Digital Differential Analyzer (DDA) for rapid grid traversal and an octree traversal scheme that recursively refines the search space using hybrid leaf node detection. Our methods build upon existing quantum orthographic ray-casting algorithms to efficiently identify cells or primitives where intersections occur before applying classical refinement through detailed intersection tests using orthographic projection. Our approach extends previous quantum ray casting techniques by adapting them to complex spatial data structures and scene representations, offering theoretical runtime improvements in optimal cases. These results demonstrate how quantum computing can enhance traditional computational rendering pipelines. As quantum hardware continues to evolve, we hope these hybrid techniques will offer alternative solutions for realistic rendering, inspiring further interdisciplinary research for quantum computation in computer graphics.

Index Terms—quantum computing, ray casting, grover's algorithm, hybrid rendering, spatial structures.

I. INTRODUCTION

In recent years, Quantum computing has demonstrated promising potential for solving problems currently difficult for classical computers [1]–[4], notably optimization and search algorithms [5]–[8]. One such algorithm, Grover's algorithm for unordered search, provides a quadratic speedup in runtime over classical exhaustive search [9]. In recent years, the algorithm has been explored for its potential in computer graphics, such as its application in ray casting [10], [11].

The ray casting algorithm is a fundamental algorithm used in rendering techniques such as ray tracing to determine visible surface meshes by shooting rays from a virtual camera into the scene [12], [13]. In its simplest form, ray casting projects the ray into the virtual scene to interact with objects

or planes within the scene, often referred to as primitives. These intersections are used as the basis for rendering and shader computations. Depending on the shading model, the intersection point can simulate various lighting components like ambient, diffuse, or specular lighting, contributing to realistic rendering [14]–[18].

While there have been numerous prior works that explored quantum approach to ray-casting-based rendering [10], [11], [19]–[21], its application and practicality still remain largely unexplored. In this paper, we extend the current state-of-the-art quantum ray casting framework for orthographic projection to address its application in existing rendering pipelines that uses special spatial structures, demonstrating its potential in hybrid rendering models for complex scenes. The two primary approach explored are:

- **Uniform Spatial Division:** The scene is partitioned into $n \times n \times n$ uniform grid of cubes and utilizes a classical 3D Digital Differential Analyzer (DDA) for grid traversal [22]. This hybrid approach uses Grover's to determine the first intersection along the ray path. Once the initial intersection is identified, the classical method continues the intersection test and rendering calculations. This approach combines the runtime speed-up of Grover's with local processing on detailed primitives.
- **Octree:** Spatial hierarchical structure has often been used as an effective partitioning method in ray tracing [23]–[25]. Our approach combines the data structure to define and redefine search space recursively. When a ray interacts with an octree node, which is a cube, the leaf nodes of that node can be recursively fed back into a quantum ray casting algorithm to pinpoint the interesting primitive. This method uses quantum search on dynamic structures without sacrificing runtime.

All rendered scene and primitive in this work are produced using a basic orthographic camera model with physically based rendering (PBR) model [26], expanded upon in Sec. III. Additionally, a brief overview of ray tracing and rendering will be covered in the same section. It should also be noted that rendering is not a primary concern of this work, as any ray-based rendering and shading model including volume shading or global illumination can be applied [17], [27], [28].

II. RELATED WORK

Quantum computation has traditionally been used for tasks such as simulation [29], optimization [30], or cryptography [31], where its ability to solve specific problems faster than classical hardware is emphasized. However, in recent years, there have been a growing interests in applying quantum algorithms for computer graphics. Notably, for various rendering and environment simulation tasks [32]–[35].

Prior to its rise in current popularity, quantum computation for computer graphics have been theorized for potential in computer graphics, for speed-up in z-buffer calculation or radiosity [36]–[38]. While radiosity has lost its popularity in recent years, current state-of-the-art approach to hybrid rendering models still take ideas from fast z-buffer visibility occlusion calculation [39].

Many subsequent works have proposed quantum applications in computer graphics through acceleration in visibility computation, supersampling, and photon mapping [20], [40]. The majority of the proposed algorithms rely on theoretical quadratic speed-up in ordered search of primitives, as the number of primitives grows larger in complex scenes (e.g [10]).

However, many of these approaches suffer from various constraints, such as data representation [40], large sample size [41], or limited scene complexity [10]. Additionally, many of the proposed approaches sacrifice runtime for feasibility, leading to degradation of practicality in implementation. Out of the many currently existing models, the one proposed by Carolina Alves et al. [10], for ray casting in orthographic projection, while still constrained, demonstrates promising results.

The method proposed by Carolina Alves et al. uses a Grover’s unordered search with a modified oracle for ray-object intersection, demonstrated by Alg.1.

Algorithm 1 Oracle for ray (x, y) intersection with N primitives

- 1: $|Hint\rangle \leftarrow |1\rangle$; $|Vint\rangle \leftarrow |1\rangle$;
 - 2: $min_X: |p\rangle |c\rangle = mX |p\rangle |0\rangle^{\otimes nc}$
 - 3: Negate $Hint$ if $x \geq c$; revert $|c\rangle$
 - 4: $max_X: |p\rangle |c\rangle = MX |p\rangle |0\rangle^{\otimes nc}$
 - 5: Negate $Hint$ if $x \leq c$; revert $|c\rangle$
 - 6: $min_Y: |p\rangle |c\rangle = mY |p\rangle |0\rangle^{\otimes nc}$
 - 7: Negate $Vint$ if $y \geq c$; revert $|c\rangle$
 - 8: $max_Y: |p\rangle |c\rangle = MY |p\rangle |0\rangle^{\otimes nc}$
 - 9: Negate $Hint$ if $y \leq c$; revert $|c\rangle$
 - 10: Flip the primitive’s coefficient sign if $intersect = 1$:
 $|p\rangle = (-1)^{intersect} * |p\rangle$
 - 11: Revert $|Hint\rangle$, $|Vint\rangle$, and $|intersect\rangle$
-

This approach, while limited to integer coordinate and orthographic projection, remains efficient in terms of qubit count for complex scenes with large quantity of primitives, requiring nc qubits, where $nc = \lceil \log_2(\max(max_X, max_Y)) \rceil$. The issue with this algorithm, as the original authors also

points out within their work, is that circuit depth can grow extremely quickly even for limited primitives, due to Grover’s iterations.

To ensure the practicality of the algorithm, the search space over all primitive count must be either limited or optimized, leading to potential increase in classical runtime overhead. Our work builds upon the proposed quantum ray casting model while adhering to its constraints, utilizing primitive modification and spatial data structure.

III. BACKGROUND

This section provides an overview of the key concepts and algorithms that underlie this paper, starting with the fundamentals of ray casting and ray tracing techniques in computer graphics. A quick overview of orthographic projection and shading is provided, which is essential for rendering the scene. Finally, a brief overview of the spatial data structure is provided, which is the basis for the hybrid rendering models. Finally, closing with Grover’s algorithm and oracle modification based on prior work.

A. Ray Casting and Ray Tracing

The ray casting algorithm in computer graphics is where rays are projected from a virtual camera into a scene of primitives to determine which objects or surfaces are intersected. It functions as the first step to ray tracing, which is a broader framework that expands upon the visibility computation. Ray tracing, unlike ray casting, re-cast rays on intersections to account for reflection, refraction, and shadows to render realistic images. Ray casting serves as the basis model for ray-tracing-based rendering pipelines.

B. Orthographic Rendering

Orthographic projection is one of the two primary rendering approaches in computer graphics [42], [43]. Unlike perspective projection, orthographic projection casts rays into the scene as parallel rays, demonstrated in Fig. 1. This projection method has the benefit of simplifying many mathematical and computational challenges involved in ray-object intersection tests, as it removes perspective distortion and allows the camera ray to be cast from multiple locations in parallel. Our work builds on the original quantum ray casting algorithm designed for orthographic cameras.

C. Spatial Data Structures

Efficient spatial data structures such as voxel grids [22], uniform grids [44], and octrees [23] are essential for accelerating 3D rendering. This structure allows the scene to be partitioned into smaller regions, reducing the number of intersection tests required by ray casting. It serves as the basis for the practicality of this work, as current and near-term quantum hardware offers limited advantages in ray casting a scene in its entirety. In this work, we explore approaches that use two different spatial partitioning methods to integrate quantum search techniques with classical ray tracing pipelines. Detailed methodologies for each approach are discussed in the following section.

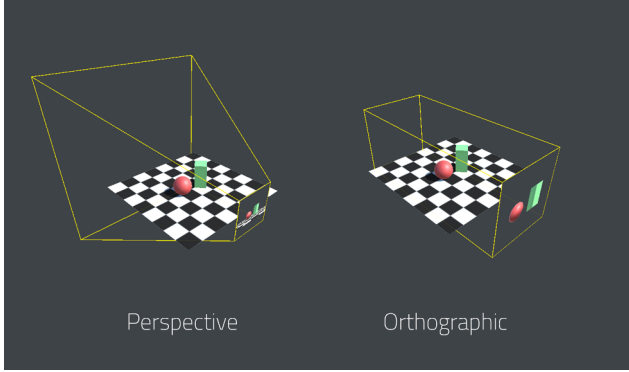


Fig. 1: Comparison of perspective projection (left) and orthographic projection (right), with near clipping plane, far clipping plane, and view frustum outlined in yellow. Checkerboard floor is not rendered in orthographic projection due to the floor plane being parallel to the camera.

IV. UNIFORM SPATIAL DIVISION PIPELINE

Uniform spatial division is a commonly used technique in graphics for accelerating ray-object intersection tests. By subdividing a scene into regular grids of cubic cells or voxels, the ray-casting algorithm search space is limited to the subset of cells that the ray traverses. In our hybrid quantum-classical approach, we use this spatial partitioning to reduce the search space for Grover’s algorithm while staying within the constraints of the original work, by limiting the search space to grids of positive integers. Subsequent detailed intersection computation, including shadow rays and reflected rays, are calculated classically.

A. Grid Partitioning and Traversal

Our work assumes a uniformly divided scene, where the 3D space is split into an $n \times n \times n$ grid of equally sized cubes. Each cell in the grid is assigned a unique identifier, similar to the primitive IDs used by Carolina Alves et al. Scene primitives in this case are associated directly with the cell or cells they occupy. This discretization simplifies the intersection problem while reducing the size of the search space. Instead of dealing with continuous geometric primitives such as a sphere, the algorithm now searches over a finite set of axis-aligned cube.

To traverse the uniform grid efficiently, we use a 3D Digital Differential Analyzer (3D-DDA) model [22]. The algorithm calculates the parameter value at which a ray crosses the boundaries between grid cells, which produces a sequence of candidate cells ordered along the ray’s direction. It is important to note that 3D-DDA is not calculated using the quantum ray-casting, only as a method of checking intersection, as it does not keep track of distance between camera and point of intersection.

The 3D-DDA algorithm initializes the starting point by mapping the ray’s origin onto the grid, in this case, the respective cell of the camera. In the classical case, the algorithm then computes the distance the ray must traverse along its

direction to reach the next cell boundary along each axis. The traversal loop incrementally steps through the grid cells until an intersection is found or until the ray exists the search grid.

B. Grover’s Algorithm

Grover’s algorithm is a quantum search algorithm that amplifies the probability amplitude of a marked item from an unsorted search space in $O(\sqrt{N})$ iterations, offering a quadratic speedup compared to classical unordered search methods [9]. At its core, Grover’s algorithm relies on an oracle that identifies the target states by flipping their phase and a diffusion operator that amplifies the probability amplitude of those marked states. In the context of our work, each quantum state encodes either a primitive or a cell from the 3D scene, and the goal is to mark those primitives that intersect a given ray.

In the original implementation by Carolina Alves et al. [10], the oracle is implemented as a reversible quantum circuit that performs geometric intersection tests. Specifically, the oracle computes the minimum and maximum boundaries of a primitive along each spatial dimension, (x, y) in the 2D case, or with an implicit z axis in the 3D case, comparing these boundaries with the ray’s origin and direction. If the ray’s coordinate falls within the computed bounds in every dimension, the oracle flips the phase of the corresponding quantum state. This phase inversion marks the state, which flags the primitive as a candidate for further intersection tests.

By applying Grover’s diffusion operator after the oracle, the amplitudes of the marked states are amplified, which increases the probability that a measurement will result in one of the states corresponding to an intersected primitive. This modified oracle is the basis for the quantum ray casting framework, allowing a fast reduction for narrowing down the search space prior to detailed classical intersection tests and rendering.

C. Algorithm Modifications

Our work uses similar boolean logic as the original quantum ray casting algorithm [10], with modifications to step size between primitives. Whereas the original algorithm required knowing the number of primitives for constructing the truth table used to depict the binary representation of the primitives, our work assumes a $n \times n \times n$ number of primitives that equally subdivides the scene.

As a demonstration, in the simplest 2D case, assume a 1×4 equally subdivided scene with 4 total primitives p_i . Each primitives are placed exactly 1 unit apart, similar to 4 unit cubes that represent the scene. These primitives can be represented as $(minX, maxX, minY, maxY, z)$, where the sample

Primitive	p_1	p_0	b_2	b_1	b_0	Decimal min_X
P_0	0	0	0	0	1	1
P_1	0	1	0	1	0	2
P_2	1	0	0	1	1	3
P_3	1	1	1	0	0	4

TABLE I: Binary encoding of the min_X coordinate for 4 primitives where only the X value matters.

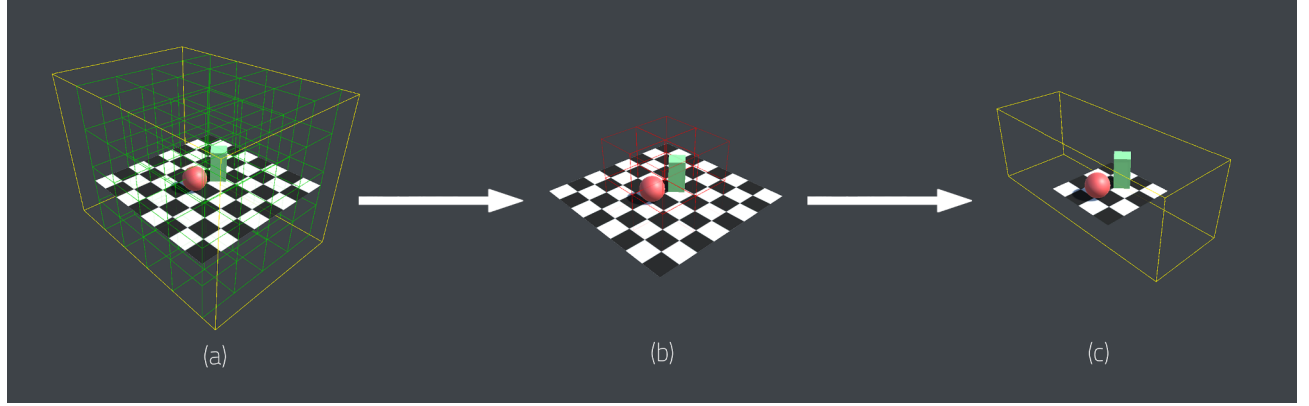


Fig. 2: Uniform spatial division rendering pipeline. The scene is first partitioned into $n \times n$ equally sized cubes within the view frustum (a) classically, with each cube assigned a unique ID, outlined with green. Quantum intersection detection is then performed on the full set of uniformly divided cubes using Alg. 1 (b), creating subset S outlined in red. Using the reduced subset S , classical intersection detection is used to calculate intersection point and rendered using classical methods (c).

Output Bit	Boolean Expression
b_2	$p_1 \wedge p_0$
b_1	$p_1 \oplus p_0$
b_0	$\neg p_0$

TABLE II: Boolean expressions for each bit of the $\min_X$ coordinate.

primitives look like $(0, 1, 0, 1, z)$, $(1, 2, 0, 1, z)$, $(2, 3, 0, 1, z)$, $(3, 4, 0, 1, z)$, where the z coordinate is omitted in the 2D case. The scene can be represented using the binary table Tab. I, where P_0 is the primitive ID, p_i is the binary encoding, and b_i represents the $\min X$ boundary. The x coordinates of each primitives is used in this case, represented by b_i .

The conversion uses boolean operations, summarized in Tab. II. Each bit of the coordinate is defined by a boolean function of the index bits:

- $b_2 = p_1 \wedge p_0$, true only when both p_0 and p_1 are 1.
- $b_1 = p_1 \oplus p_0$, the XOR operation of index bits, only becoming 1 when bits differ.
- $b_0 = \neg p_0$, negating the least significant index bit.

These operations are then encoded into a quantum circuit using CNOT, Toffoli, and NOT gates, similar to the original implementation. The example circuit for the $\min X$ case can be seen in Fig. 3.

For other configuration such as $\max X$ or $\max Y$, including the z axis, can be encoded using the same method, before being appended into Alg. 1 for the full oracle, as demonstrated by the original work [10].

D. Hybrid Processing

It's important to note that, in our approach, the quantum ray-casting algorithm is only used to determine the first intersected cell. The Grover's oracle encodes the grid cell IDs and amplifies the intersected state, similar to the original ray-casting algorithm, as shown in Alg. 2. The Oracle is designed

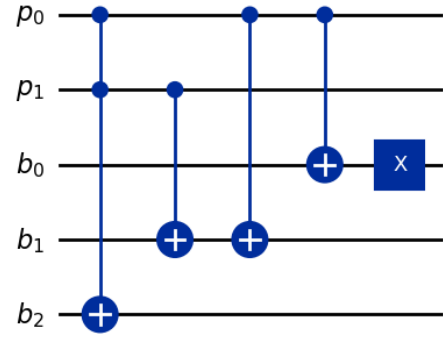


Fig. 3: Quantum circuit encoding a primitive's $\min X$ coordinate from a 2 bit index into a 3 qubit binary representation using CNOT, CCX, and X gates for the boolean functions in Tab. II.

to mark the cell if the ray's traversal in the 3D-DDA structure results in an intersection.

After the quantum algorithm identifies the first point of intersection, the classical 3D-DDA algorithm is applied to the reduced subset of cells, demonstrated in Alg. 3. The classical algorithm performs a detailed intersection test within the cell, verifying the precise point of intersection and shading computation using the PBR model. Additional rays created by ray tracing, such as reflection rays, are traced using the classical approach, as the reflected rays are unlikely to be axis-aligned.

E. Performance and Trade-Offs

The uniform spatial division approach, combined with 3D-DDA, offers theoretical speedup for reducing search space for ray-casting-based rendering. However, the efficiency of

Algorithm 2 Quantum Intersection Detection in Uniform Grid

```
1: Input: Scene grid of size  $n \times n \times n$  with unique cell IDs; ray parameters.
2: Output: Subset  $S \subseteq \{\text{cell IDs}\}$  of cells with an intersection.
3: Initialize a uniform superposition over all cell IDs.
4: for  $i = 1$  to  $r$  do  $\triangleright r$  Grover iterations
5:   Apply the Quantum Oracle to mark cells that intersect the ray.  $\triangleright$  Algorithm 1
6:   Apply the Grover Diffusion Operator.
7: end for
8: Measure the quantum register.
9: Extract cell IDs corresponding to the marked states.
10: return  $S$ 
```

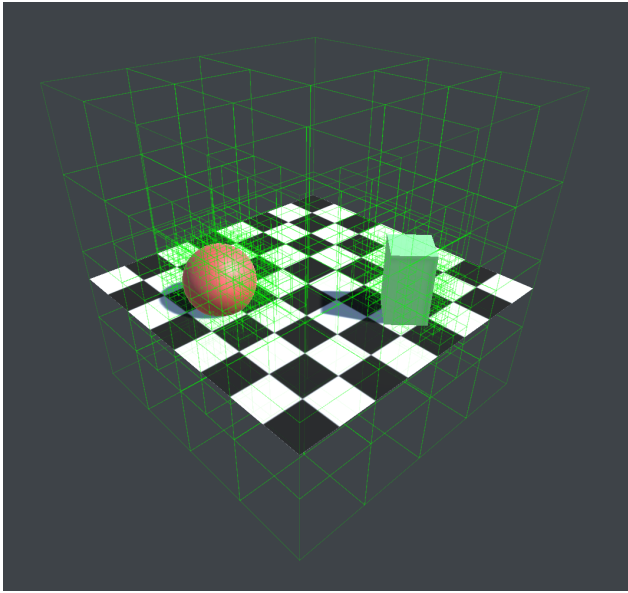


Fig. 4: Octree structure with each node outlined with green. The tree is initialized with maximum depth of 5. The tree continues to subdivide into smaller bounding volumes around the meshes until it reaches the maximum depth. The checkerboard floor is accounted for during subdivision.

the quantum search depends on multiple factors, such as grid resolution and the density of intersected cells.

While a grid with larger n provides a more precise localization, the increase in gate-depth and Grover’s iteration can grow exponentially. Conversely, a small n can lead to ambiguous intersections, requiring additional classical processing that can lead to overhead, degrading the quantum advantage provided by reducing search space. Selecting an optimal n in this case will remain important, and should be tailored to existing quantum hardware platforms.

V. ITERATIVE SUBDIVISION WITH OCTREE

Unlike uniform spatial division, octrees are a recursive hierarchical spatial data structure that efficiently partitions

the 3D scene by subdividing it into eight smaller cubical regions or octants [23]. It follows the same idea of space subdivision as uniform spatial division but with additional recursive subdivision. This structure adapts well to scenes with non-uniform object distributions and varying levels of detail (LODs) [45], [46]. Objects such as a sphere can be divided into voxelized structures, with each level of the tree covering a higher level of detail.

Algorithm 3 3D-DDA Intersection and Rendering

```
1: Input:
   • Scene grid of size  $n \times n \times n$ 
   • Ray parameters and camera settings (using orthographic projection)
   • Light source parameters and material properties for PBR shading
2: Output: RGB color values for every pixel.
3:  $S \leftarrow \text{QUANTUMINTERSECTIONDETECTION}(\text{scene grid, ray})$ 
4: for each pixel  $(i, j)$  in the image do
5:   Generate the ray corresponding to pixel  $(i, j)$ .
6:   Initialize  $t \leftarrow 0$ , set current cell  $c$  from the ray origin.
7:   while cell  $c$  exists and no intersection is confirmed do
8:     Compute the next cell using 3D-DDA.
9:     if  $c \in S$  then
10:      Perform classical intersection test in cell  $c$ .
11:      if Point  $P$  intersect with normal  $N$  then
12:        Render PBR shading
13:        Assign pixel  $(i, j)$  an RGB color.
14:        break from the while loop.
15:      end if
16:    end if
17:  end while
18:  if no intersection is detected after grid traversal then
19:    Assign a background color to pixel  $(i, j)$ .
20:  end if
21: end for
22: return the set of RGB color values for all pixels.
```

In our approach, the octree serves as a multi-resolution framework that uses recursive quantum ray-casting at different scales. The idea is to use octrees to quickly eliminate large portions of the scene that do not intersect with the ray, similar to uniform spatial subdivision, and then apply classical ray-casting when the quantum model starts to lose precision, ultimately refining intersection test down to levels of individual primitives.

A. Methods

The octree is traditionally constructed by recursively subdividing the scene’s bounding volume until a predetermined depth is reached or until each leaf node contains a sufficiently small number of primitives. During construction, each node of the octree represents an axis-aligned cubic volume and is assigned a unique ID. While not a strict requirement for the octree structure, all nodes are assumed to be axis-aligned

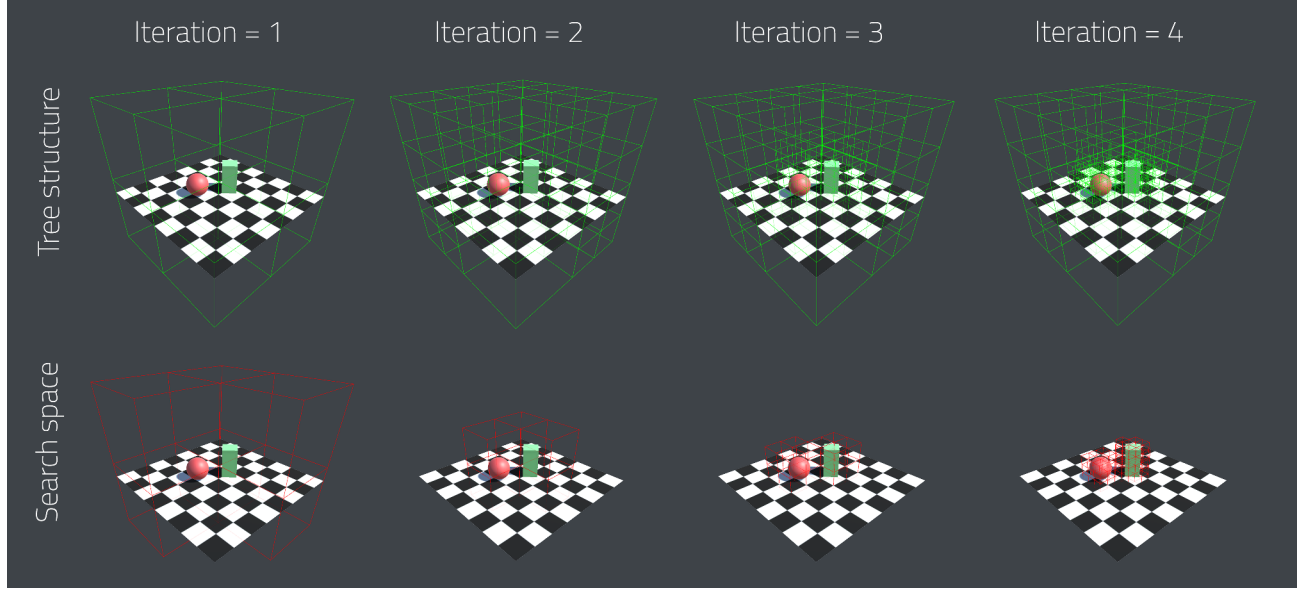


Fig. 5: The octree tree structure, outlined in green, and each subsequent search space, outlined in red, with 4 iterations of the recursive subdivision. If a node (green cube) contains meshes, branches are added to the node to further subdivide into additional leaf nodes (red cube).

parallel to the camera’s view frustum, demonstrated in Fig. 4. For nodes that are further subdivided, the union of their children’s volume represents the parent’s volume.

The hierarchical organization allows for quick navigation through regions of interest. It is important to ensure that the octree stays balanced during population, where densely populated areas receive more subdivisions, while sparse regions remain as large nodes. In our implementation, we assume a $O(1)$ construction of the octree for the scene, as octree construction is not a part of this work.

Once the octree is constructed, the ray traversal process begins at the root of the node. As the ray enters a node, a preliminary intersection test is performed to determine if the ray intersects that node’s bounding volume, using the same Alg. 2 demonstrated in the previous section. If an intersection is detected, the algorithm recursively descends into the child nodes, where the quantum-ray casting algorithm continues until a leaf node, set depth, or a loss in precision. Often, the algorithm will terminate after a set depth, as it is still constrained to the positive integer search space.

After a potential intersection is detected at a set depth or a leaf node, classical intersection tests confirm the exact point of intersection. Once confirmed, the shading model is then applied to compute the pixel color. For this work, we iteratively searched until depth = 4, demonstrated in Fig. 5.

B. Performance and Trade-Offs

The octree traversal method serves as an extension of the uniform spatial subdivision method but with recursive subdivision of a given bounding volume. It offers a significant advantage for complex scenes, particularly when the scene

consists of non-uniform geometry. By localizing Grover’s search space to smaller branch nodes, the primitive target subset is significantly reduced, allowing for practicality without exponentially scaling gate depth and Grover’s iteration.

It is important to point out, however, that the efficiency of this approach greatly depends on the octree’s depth and distribution of primitives. Over-subdivision may lead to an increase in overhead in managing hierarchy, whereas under-subdivision can result in a larger search space for the classical component, which diminishes the theoretical quantum advantage.

C. Tree Traversal Approximation

One of the biggest limitations to the octree traversal approach, as pointed out in previous sections, is imposed by the data structure used to represent the primitive subdivisions [10], [39]. Since the subdivisions are set up as uniformly distributed axis-aligned cubes incremented as integers, the quantum subdivision search can only proceed until each cube are at least 1 unit apart.

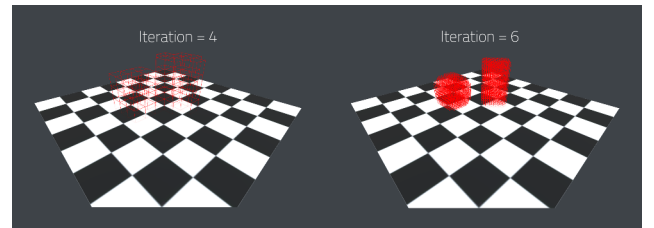


Fig. 6: Voxel approximation from the octree traversal, with depth = 4 (left), and depth = 6 (right).

While it is possible to normalize each level of the tree such that each of the eight possible cubes are exactly 1 unit apart, this would nullify any quantum advantage provided by Grover’s algorithm unless this normalization is performed during the construction of the tree.

Depending on practicality, the construction of a normalized octree is still possible. Assume such an octree structure has been provided, or alongside some practical implementation that allows for floating point representation using qubits, it is possible in theory to continue the subdivision past unit cubes. Using said continued subdivision of the octree, the algorithm is able to provide a close approximation of the mesh enclosed similar to a voxel structure, without any classical intersection tests, demonstrated by Fig. 6.

It should be noted however, this implementation of an octree requires additional calculation during construction, which remains unconventional and impractical unless used to bypass the data structure constraint imposed by qubits. The authors believe that, while it is possible, to be truly practical, future work should rely on better data representation of primitives rather than altering the construction of the octree.

VI. DISCUSSIONS AND LIMITATIONS

While our work demonstrates a potentially practical usage of quantum ray casting for orthographic cameras, several limitations must be addressed. First, similar to the majority of prior work, our analysis assumes $O(1)$ input initialization and retrieval for the quantum components in the hybrid pipeline. In practice, encoding a large amount of primitive geometric data into quantum registers can be computationally expensive, and the overhead associated with state preparation should not be ignored. This assumption is the foundation for the quadratic speedup provided by Grover’s algorithm, and any additional runtime can significantly impact the overall performance and potentially nullify any theoretical speedup provided.

Furthermore, the current state of quantum hardware, while improving noticeably in recent years, is still in its early stages in terms of practicality [47]. The number of qubits available, along with issues related to qubit coherence, error rates, and gate fidelity, pose significant challenges to implementing our proposed methods on hardware at a scale that would be competitive with classical algorithms [48]. At this stage, our work remains primarily theoretical, with similar results only achievable through simulation.

Attempts at implementing this pipeline on existing quantum hardware have been made, specifically on Rigetti Ankaa-3, but with little to no success. Even a small scene with two primitives, uniformly subdivided into $4 \times 4 \times 4$ cubes, returned amplitudes that are unusable for determining intersection, as demonstrated by Fig. 7.

For this reason, our work at the current stage heavily relies on a simulator as a proof of concept, and will remain so until hardware improvements. Reliable intersection tests on hardware are only achievable with a sliding-window implementation that further cuts up the grid into smaller subunits, demonstrated by Fig. 8 and Fig. 9.

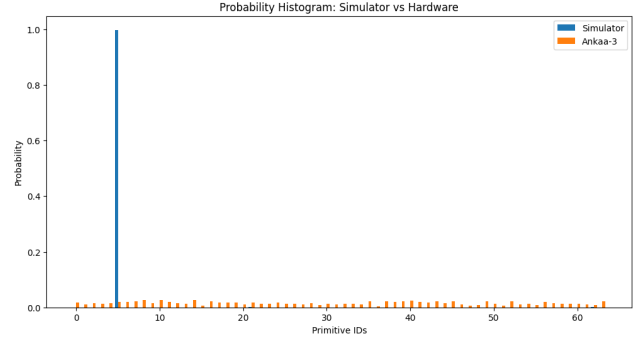


Fig. 7: Probability amplitude histogram that compares results from running Grover’s search on $4 \times 4 \times 4$ uniformly subdivided cubes. Blue bar indicates simulator results and orange bar indicates hardware result returned by Rigetti Ankaa-3.

Limitations also exist due to the nature of Grover’s algorithm. While it provides a theoretical quadratic speedup for unstructured search, it does not necessarily hold true when multiple marked states exist, such as an overlap in primitives within a single cell. For the purpose of this work, we assume a cell is either occupied or unoccupied. Additionally, the oracle construction in our pipeline, proposed by Carolina Alves et al. [10], introduces overhead that may offset some of the benefits provided by the quantum search. Our work relies heavily on complex oracle constructions that perform geometric computations for a static scene. In practice, designing and optimizing such oracles requires significant circuit depth and ancillary qubits, adding additional challenges posed by existing hardware.

For any real-time usage of ray-based photorealistic rendering, classical methods should still be the primary option, as hybrid models can potentially introduce latency required in real-time rendering tasks. Our work assumes that hard-

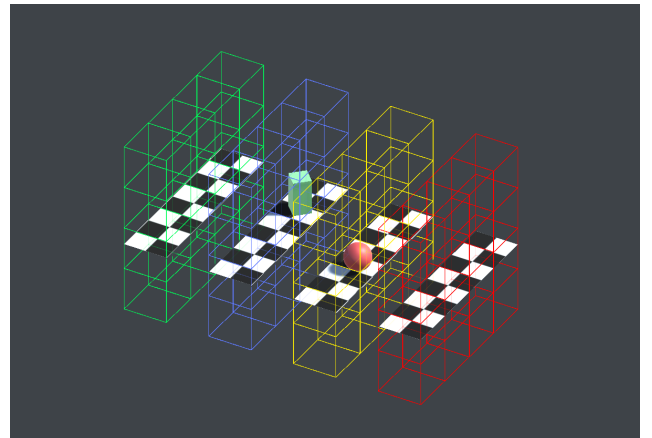


Fig. 8: Uniformly subdivided scene that uses a sliding window subdivision which splits each set of cubes at different depth (z-axis) into subsets. Each subset indicated by different colors.

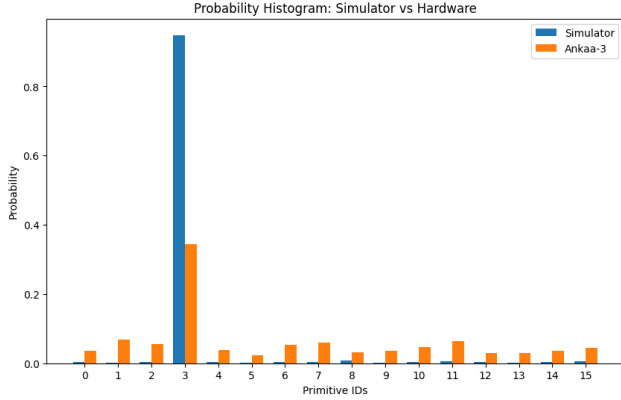


Fig. 9: Probability amplitude histogram that compares results from running Grover’s search on 4×4 uniformly subdivided cubes. Blue bar indicates simulator results and orange bar indicates hardware result returned by Rigetti Ankaa-3.

ware communication latency can be disregarded within the pipeline, where scenes do not need to be updated in real-time. Furthermore, we assume the scene contains no dynamic primitives, except primitives that do not collide with any rays from the camera. Any changes to the camera location or angle will inevitably lead to a change in ray origin, leading to the reconstruction of the oracle.

VII. CONCLUSION AND FUTURE WORK

In this work, we introduced two practical and novel applications of hybrid methods for quantum ray casting in computer graphics: using uniform spatial division with 3D-DDA to identify the first cell intersection, and octree traversal that uses hierarchical spatial partitioning with hybrid leaf search. Both methods integrate Grover’s algorithm into the hybrid orthographic ray-casting pipeline, providing theoretical quadratic speedup in intersection detection compared to classical search methods.

It is important to note that while our work demonstrates promising results, it remains heavily theoretical and simulation-based at its current stage, relying on several assumptions regarding input output encoding and instant oracle construction. Addressing the challenges bypassed by these assumptions is required, as they may challenge the immediate practical implementation of this work. Furthermore, while this work mainly focuses on orthographic projection, most real-world applications of ray-based rendering use the more complex perspective projection model.

Future work should not only aim to refine the quantum oracle and hybrid pipeline but also look into additional spatial data structures beyond ones that use uniform subdivisions. Structures such as kd-trees [49], BSP trees [50], or adaptive partitioning schemes [44], should also be possible in theory, given that the search space and target can be encoded into the oracle efficiently. Additionally, extending the framework to handle parallel rays, such as in the context of anti-aliasing,

reflections, path tracing, and photon mapping, could further enhance the visual realism and applicability of the hybrid rendering model [17], [27], [51].

With this work, the authors hope that it inspires further research that uses hybrid quantum models for computer graphics and that it serves as a starting point for practical, scalable quantum algorithms capable of addressing the computational challenges of rendering techniques in computer graphics.

ACKNOWLEDGMENT

This work was supported by the National Defense Education Program (NDEP) for Science, Technology, Engineering, and Mathematics (STEM) Education, Outreach, and Workforce Initiative Programs under Grant No. HQ0034-21-1-0014. We would also like to thank Professor Mahdi Hosseini for supporting the Quantum Game Club.

REFERENCES

- [1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.
- [2] D. Ristè, M. P. Da Silva, C. A. Ryan, A. W. Cross, A. D. Córcoles, J. A. Smolin, J. M. Gambetta, J. M. Chow, and B. R. Johnson, “Demonstration of quantum advantage in machine learning,” *npj Quantum Information*, vol. 3, no. 1, p. 16, Apr. 2017. [Online]. Available: <https://www.nature.com/articles/s41534-017-0017-3>
- [3] H.-Y. Huang, M. Broughton, J. Cotler, S. Chen, J. Li, M. Mohseni, H. Neven, R. Babbush, R. Kueng, J. Preskill, and J. R. McClean, “Quantum advantage in learning from experiments,” *Science*, vol. 376, no. 6598, pp. 1182–1186, Jun. 2022. [Online]. Available: <https://www.science.org/doi/10.1126/science.abn7293>
- [4] A. J. Daley, I. Bloch, C. Kokail, S. Flannigan, N. Pearson, M. Troyer, and P. Zoller, “Practical quantum advantage in quantum simulation,” *Nature*, vol. 607, no. 7920, pp. 667–676, Jul. 2022. [Online]. Available: <https://www.nature.com/articles/s41586-022-04940-6>
- [5] N. Moll, P. Barkoutsos, L. S. Bishop, J. M. Chow, A. Cross, D. J. Egger, S. Filipp, A. Fuhrer, J. M. Gambetta, M. Ganzhorn, A. Kandala, A. Mezzacapo, P. Müller, W. Riess, G. Salis, J. Smolin, I. Tavernelli, and K. Temme, “Quantum optimization using variational algorithms on near-term quantum devices,” *Quantum Science and Technology*, vol. 3, no. 3, p. 030503, Jul. 2018. [Online]. Available: <https://iopscience.iop.org/article/10.1088/2058-9565/aab822>
- [6] A. Abbas, A. Ambainis, B. Augustino, A. Bäertschi, H. Buhrman, C. Coffrin, G. Cortiana, V. Dunjko, D. J. Egger, B. G. Elmegreen, N. Franco, F. Fratini, B. Fuller, J. Gacon, C. Goniculea, S. Gribling, S. Gupta, S. Hadfield, R. Heese, G. Kircher, T. Kleinert, T. Koch, G. Korpas, S. Lenk, J. Marecek, V. Markov, G. Mazzola, S. Mensa, N. Mohseni, G. Nannicini, C. O’Meara, E. P. Tapia, S. Pokutta, M. Proissl, P. Rebentrost, E. Sahin, B. C. B. Symons, S. Tornow, V. Valls, S. Woerner, M. L. Wolf-Bauwens, J. Yard, S. Yarkoni, D. Zechiel, S. Zhuk, and C. Zoufal, “Challenges and opportunities in quantum optimization,” *Nature Reviews Physics*, vol. 6, no. 12, pp. 718–735, Oct. 2024. [Online]. Available: <https://www.nature.com/articles/s42254-024-00770-9>
- [7] P. R. Giri and V. E. Korepin, “A review on quantum search algorithms,” *Quantum Information Processing*, vol. 16, no. 12, p. 315, Dec. 2017. [Online]. Available: <http://link.springer.com/10.1007/s11128-017-1768-7>
- [8] P.-E. Emeriau, M. Howard, and S. Mansfield, “Quantum Advantage in Information Retrieval,” *PRX Quantum*, vol. 3, no. 2, p. 020307, Apr. 2022. [Online]. Available: <https://link.aps.org/doi/10.1103/PRXQuantum.3.020307>
- [9] L. K. Grover, “A fast quantum mechanical algorithm for database search,” in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, ser. STOC ’96. New York, NY, USA: Association for Computing Machinery, 1996, p. 212–219. [Online]. Available: <https://doi.org/10.1145/237814.237866>

- [10] C. Alves, L. P. Santos, and T. Bashford-Rogers, "A quantum algorithm for ray casting using an orthographic camera," in *2019 International Conference on Graphics and Interaction (ICGI)*. IEEE, 2019, pp. 56–63.
- [11] S. Lee, S. Jeong, and J. Seo, "Efficient quantum circuit encoding of object information in 2d ray casting," in *2024 International Technical Conference on Circuits/Systems, Computers, and Communications (ITC-CSCC)*. IEEE, 2024, pp. 1–6.
- [12] S. D. Roth, "Ray casting for modeling solids," *Computer graphics and image processing*, vol. 18, no. 2, pp. 109–144, 1982.
- [13] H. Ray, H. Pfister, D. Silver, and T. A. Cook, "Ray casting architectures for volume visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 3, pp. 210–223, 1999.
- [14] H. Gouraud, "Continuous shading of curved surfaces," *IEEE Trans. Comput.*, vol. 20, no. 6, p. 623–629, Jun. 1971. [Online]. Available: <https://doi.org/10.1109/T-C.1971.223313>
- [15] B. T. Phong, *Illumination for computer generated pictures*. New York, NY, USA: Association for Computing Machinery, 1998, p. 95–101. [Online]. Available: <https://doi.org/10.1145/280811.280980>
- [16] J. F. Blinn, "Models of light reflection for computer synthesized pictures," *SIGGRAPH Comput. Graph.*, vol. 11, no. 2, p. 192–198, Jul. 1977. [Online]. Available: <https://doi.org/10.1145/965141.563893>
- [17] J. T. Kajiya, "The rendering equation," *SIGGRAPH Comput. Graph.*, vol. 20, no. 4, p. 143–150, Aug. 1986. [Online]. Available: <https://doi.org/10.1145/15886.15902>
- [18] E. Walia and C. Singh, "An analysis of linear and non-linear interpolation techniques for three-dimensional rendering," in *Proceedings of the Conference on Geometric Modeling and Imaging: New Trends*, ser. GMAI '06. USA: IEEE Computer Society, 2006, p. 69–76. [Online]. Available: <https://doi.org/10.1109/GMAI.2006.5>
- [19] L. P. Santos, T. Bashford-Rogers, J. Barbosa, and P. Navrátil, "Towards quantum ray tracing," *IEEE Transactions on Visualization and Computer Graphics*, 2024.
- [20] X. Lu and H. Lin, "Improved quantum supersampling for quantum ray tracing," *Quantum Information Processing*, vol. 22, no. 10, p. 359, 2023.
- [21] —, "Quantum ray tracing with simulation," in *Spin*, vol. 12, no. 04. World Scientific, 2022, p. 2250030.
- [22] J. Amanatides, A. Woo *et al.*, "A fast voxel traversal algorithm for ray tracing," in *Eurographics*, vol. 87, no. 3, 1987, pp. 3–10.
- [23] D. Meagher, "Geometric modeling using octree encoding," *Computer graphics and image processing*, vol. 19, no. 2, pp. 129–147, 1982.
- [24] F. Cazals, G. Drettakis, and C. Puech, "Filtering, clustering and hierarchy construction: a new solution for ray-tracing complex scenes," in *Computer graphics forum*, vol. 14, no. 3. Wiley Online Library, 1995, pp. 371–382.
- [25] V. Havran, R. Herzog, and H.-P. Seidel, "On the fast construction of spatial hierarchies for ray tracing," in *2006 IEEE Symposium on Interactive Ray Tracing*. IEEE, 2006, pp. 71–80.
- [26] M. Pharr, W. Jakob, and G. Humphreys, *Physically based rendering: From theory to implementation*. MIT Press, 2023.
- [27] H. W. Jensen, *Realistic image synthesis using photon mapping*. AK Peters/crc Press, 2001.
- [28] H. Wann Jensen, S. R. Marschner, M. Levoy, and P. Hanrahan, "A practical model for subsurface light transport," in *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, 2023, pp. 319–326.
- [29] C. Zalka, "Simulating quantum systems on a quantum computer," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 454, no. 1969, pp. 313–322, 1998.
- [30] G. G. Guerreschi and A. Y. Matsuura, "Qaoa for max-cut requires hundreds of qubits for quantum speed-up," *Scientific reports*, vol. 9, no. 1, p. 6903, 2019.
- [31] N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden, "Quantum cryptography," *Reviews of modern physics*, vol. 74, no. 1, p. 145, 2002.
- [32] A. Glassner, "An introduction to quantum computing," in *ACM SIGGRAPH 2024 Courses*, 2024, pp. 1–65.
- [33] M. Lanzagorta and J. K. Uhlmann, "Hybrid quantum-classical computing with applications to computer graphics," in *ACM SIGGRAPH 2005 Courses*, ser. SIGGRAPH '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 2–es. [Online]. Available: <https://doi.org/10.1145/1198555.1198723>
- [34] S. Caraiman and V. Manta, "Image processing using quantum computing," 01 2012, pp. 1–6.
- [35] S. Caraiman and V. I. Manta, "New applications of quantum algorithms to computer graphics: the quantum random sample consensus algorithm," in *Proceedings of the 6th ACM Conference on Computing Frontiers*, ser. CF '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 81–88. [Online]. Available: <https://doi.org/10.1145/1531743.1531757>
- [36] S. Caraiman, "Quantum computer graphics algorithms," *Buletinul Institutului Politehnic din Iasi, Sectia Automatica si Calculatoare*, vol. 62, no. 4, pp. 21–38, 2012.
- [37] K. Sadakane, N. Sugawara, and T. Tokuyama, "Quantum computation in computational geometry," *Interdisciplinary information sciences*, vol. 8, no. 2, pp. 129–136, 2002.
- [38] A. Glassner, "Quantum computing. 3," *IEEE Computer Graphics and Applications*, vol. 21, no. 6, pp. 72–82, 2002.
- [39] L. P. Santos, T. Bashford-Rogers, J. Barbosa, and P. Navrátil, "Towards quantum ray tracing," *IEEE Transactions on Visualization and Computer Graphics*, vol. 31, no. 4, pp. 2223–2234, 2025.
- [40] —, "Towards quantum ray tracing," 2022. [Online]. Available: <https://arxiv.org/abs/2204.12797>
- [41] L. Mosier, M. McGuire, and T. Hachisuka, "Quantum ray marching: Reformulating light transport for quantum computers," in *SIGGRAPH Asia 2023 Conference Papers*, ser. SA '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: <https://doi.org/10.1145/3610548.3618151>
- [42] J. D. Foley, *Computer graphics: principles and practice*. Addison-Wesley Professional, 1996, vol. 12110.
- [43] P. Shirley, M. Ashikhmin, and S. Marschner, *Fundamentals of computer graphics*. AK Peters/CRC Press, 2009.
- [44] H. Samet, *The design and analysis of spatial data structures*. Addison-wesley Reading, MA, 1990, vol. 85.
- [45] H. Hoppe, "Progressive meshes," in *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, 2023, pp. 111–120.
- [46] D. Luebke, M. Reddy, J. D. Cohen, A. Varshney, B. Watson, and R. Huebner, *Level of detail for 3D graphics*. Elsevier, 2002.
- [47] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, p. 79, 2018.
- [48] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R.arends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.
- [49] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.
- [50] H. Fuchs, Z. M. Kedem, and B. F. Naylor, "On visible surface generation by a priori tree structures," in *Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, 1980, pp. 124–133.
- [51] D. P. Mitchell and A. N. Netravali, "Reconstruction filters in computer-graphics," *ACM Siggraph Computer Graphics*, vol. 22, no. 4, pp. 221–228, 1988.