Towards Quantum Ray Tracing

Luís Paulo Santos, Thomas Bashford-Rogers, João Barbosa, and Paul Navrátil

***Abstract*—Rendering on conventional computers is capable of generating realistic imagery, but the computational complexity of these light transport algorithms is a limiting factor of image synthesis. Quantum computers have the potential to significantly improve rendering performance through reducing the underlying complexity of the algorithms behind light transport. This article investigates hybrid quantum-classical algorithms for ray tracing, a core component of most rendering techniques. Through a practical implementation of quantum ray tracing in a 3D environment, we show quantum approaches provide a quadratic improvement in query complexity compared to the equivalent classical approach. Based on domain specific knowledge, we then propose algorithms to significantly reduce the computation required for quantum ray tracing through exploiting image space coherence and a principled termination criteria for quantum searching. We show results obtained using a simulator for both Whitted style ray tracing, and for accelerating ray tracing operations when performing classical Monte Carlo integration for area lights and indirect illumination.**

***Index Terms*—Quadratic query complexity quantum advantage, quantum computing, ray tracing.**

I. Introduction

QUANTUM computing, originally proposed by Benioff [1] and Feynman [2], has witnessed huge developments since its inception in the 80's. Quantum algorithms exhibiting a complexity advantage over classical ones have been developed. These include Shor's algorithm for integer factorisation [3] and Grover's algorithm for searching on an unstructured data set [4]. The former exhibits an exponential complexity gain (i.e. from $\mathcal{O}(N)$ to $\mathcal{O}(log(N))$) and the latter a quadratic advantage (from $\mathcal{O}(N)$ to $\mathcal{O}(\sqrt{N})$). Ray tracing is a visibility evaluation algorithm that determines which, if any, geometric primitive is intersected by a ray nearest to its origin. In the absence of a partial spatial ordering ray tracing has a worst-case complexity of $\mathcal{O}(N)$, where N is the number of geometric primitives in S, the set describing the 3D virtual scene. This paper investigates the use of quantum computing to perform ray tracing over an unordered set of geometric primitives, whilst achieving a query complexity gain with respect to a classical approach.

Ray tracing is a search problem. For any ray r the algorithm has to find the geometric primitive $p \in S$ which complies with 2 conditions: i) r intersects p, as denoted by $f(r,p) = 1$ and ii) the depth of the current intersection along the ray's direction is the minimum among all intersections occurring in S, as denoted by $g(r,p) \leq g(r,p'), \forall (p' \in S | f(r,p') == 1)$. These 2 conditions can be captured on a single intersection function, denoted by $f_{int}()$:

$$f_{int}(r,p,\min) = \begin{cases} 1 & \text{if } f(r,p) = 1 \text{ and } g(r,p) < \min \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The algorithm in Fig. 2 describes ray tracing over an unordered set S of geometric primitives; its complexity is $\mathcal{O}(N)$. Classical spatial sorting allows for a logarithmic improvement in complexity [5], however, the quadratic quantum advantage claimed in this paper refers to the non ordered classical algorithm. There are quantum exact search algorithms over ordered lists with theoretical proofs of $\mathcal{O}(log(N))$ query complexity [6]. However, with current technology, these algorithms are not practical for either execution or simulation, which prevents their usage in this paper; future developments may allow the integration of our proposed method with this type of spatial ordering and ordered search.

Quantum searching, also referred to as Grover's algorithm [4], or in its more general form as amplitude amplification [7], allows for a quadratic query complexity gain over the classical case for unstructured, non-ordered, data sets. The search key evaluation function – $f_{int}()$ – is evaluated (queried) $\mathcal{O}(\sqrt{N})$ times, rather than $\mathcal{O}(N)$. In this paper we focus on the development and analysis of quantum searching algorithms for ray tracing. Based on certain assumptions, we demonstrate a quadratic query complexity advantage when compared to a classical approach without spatial data structures. These results are demonstrated for primary, shadow and indirect rays in a 3D environment.

Our work builds towards demonstrating that quantum ray tracing is possible. However, rendering a real 3D environment in a quantum computer is currently not possible due to the quantum hardware's limited robustness to noise and coherence time. Simulation of quantum hardware on classical computers is therefore used, allowing both the development of the quantum algorithms and gaining an understanding of their performance without the constrains of a real machine. Performance is hereby used to denote query complexity, i.e. the number of intersection evaluations required by the algorithm as a function of the number of geometric primitives. No claim is presented w.r.t. rendering time, since execution on real machines is currently not possible.

Manuscript received 11 July 2023; revised 21 March 2024; accepted 2 April 2024. Date of publication 8 April 2024; date of current version 28 February 2025. This work was supported by the National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within project LA/P/0063/2020. [DOI: 10.54499/LA/P/0063/2020]. Recommended for acceptance by Y. Zhao. *(Corresponding author: Luís Paulo Santos.)*

Luís Paulo Santos is with Universidade do Minho & INESC-TEC, 4710-057 Braga, Portugal (e-mail: psantos@di.uminho.pt).

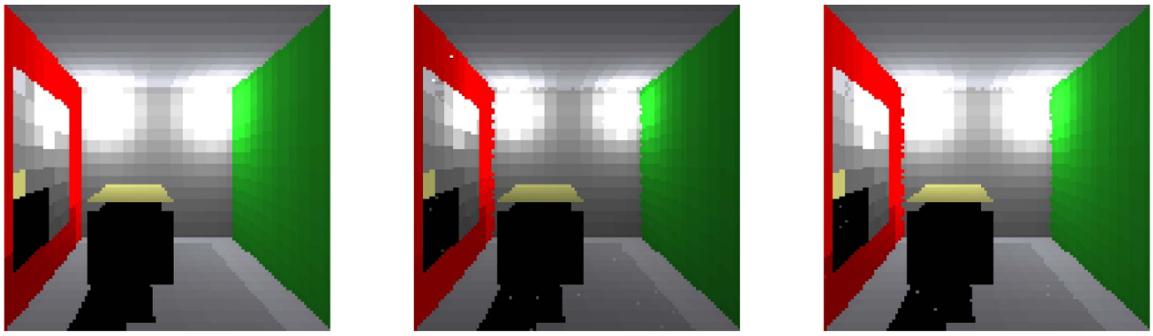
Thomas Bashford-Rogers is with the University of Warwick, CV4 7AL Coventry, U.K..

João Barbosa is with INESC-TEC, 4710-057 Braga, Portugal.

Paul Navrátil is with the Texas Advanced Computing Center, University of Texas at Austin, Austin, TX 78712 USA.

This article has supplementary downloadable material available at https://doi.org/10.1109/TVCG.2024.3386103, provided by the authors.

Digital Object Identifier 10.1109/TVCG.2024.3386103



(a) Reference classical rendering: $\approx 2678K$ intersections evaluated for 41842 rays (64 per ray).

(b) Non-optimized quantum rendering: $\approx 1366K$ intersection evaluations (33.6 per ray).

(c) Optimized quantum rendering: $\approx 896K$ intersection evaluations (22.1 per ray).

Fig. 1. 64 geometric primitives Qornell Box rendered at 128x128 resolution.

procedure RT(r, S)
  $MinDepth \leftarrow +\infty$; $minP \leftarrow -1$
  for $p \in S$ **do**
    if $f_{int}(r, p, MinDepth) == 1$ **then**
      $MinDepth \leftarrow g(r, p)$; $minP \leftarrow p$
    end if
  end for
end procedure
return ($minP$, $MinDepth$)

Fig. 2. Ray tracing over an unordered set of geometric primitives.

procedure QSEARCH($\widehat{\mathcal{A}}, f(\cdot)$)
  $i \leftarrow$ measure ($\widehat{\mathcal{H}}\,|0\rangle$) ▷ sample the superposition
  if $f(i) == 1$ **then**
    found $\leftarrow True$
  else
    $M_0 \leftarrow 0$; $l \leftarrow 0$; found$\leftarrow False$
    c constant, such that $c \in (1, 2)$
    while not found and $M_l < \left\lceil \sqrt{N} \right\rceil$ **do**
      $l \leftarrow l + 1$
      $M_l \leftarrow \min(\lceil c^l \rceil, \left\lceil \sqrt{N} \right\rceil)$
      $r_l \leftarrow$ randInt$(1 \ldots M_l)$
      $|\Psi\rangle \leftarrow \widehat{\mathcal{Q}}^{r_l} \widehat{\mathcal{A}}\,|0\rangle$
      $i \leftarrow$ measure ($|\Psi\rangle$)
      if $f(i) == 1$ **then**
        found $\leftarrow True$
      end if
    end while
  end if
  return (i, found)
end procedure

Fig. 3. Adaptive Exponential Search algorithm.

While the proposed quantum algorithm leads to improvement over a classical approach, we observe that there is scope for many optimisations for the specific case of rendering. We propose two such optimisations leading to a hybrid quantum-classical algorithm. The first is to exploit image space coherence to reduce noise resulting from the non-deterministic nature of measurements in quantum systems. The second is a novel principled termination criteria for the quantum searching algorithm, which substantially decreases the amount of computation required to generate images.

To summarize, the contributions of this paper are:

- A hybrid quantum-classical ray tracer using quantum searching algorithms,
- A thorough analysis of the performance of quantum algorithms for ray tracing compared to classical ray tracing,
- A hybrid quantum-classical algorithm for optimizing performance through exploiting image space coherence and principled stopping criteria for quantum searching,
- Further demonstrations of the use of quantum ray tracing such as evaluating visibility for Monte Carlo integration of area light sources and indirect illumination.

This is, to the extent of the authors' knowledge, the first hybrid quantum-classical ray tracing system for 3D environments to be demonstrated, confirming a quadratic query complexity advantage in a simulated environment, as predicted by the theory.

II. Related Work

The idea of utilizing quantum mechanics for computation extends back to work by Benioff [1], Feynmann [2] and Deutsch [8]. Building on these concepts, well known algorithms such as factoring primes [3] and quantum searching [4], [7] were developed over the subsequent decades.

Quantum searching is fundamental for the current work. Consider the function $f : X \rightarrow \{0, 1\}, X = \{0, \ldots N-1\}, N = 2^n$. Searching is described as the problem of finding a value $x \in X$ such that $f(x) = 1$. If there are t solutions, i.e. t different values of x satisfying $f(x) = 1$, then the complexity of quantum searching is $\mathcal{O}(\sqrt{\frac{N}{t}})$, independently on whether t is previously known or unknown. For the classical case, and if $f : X \rightarrow \{0, 1\}$ does not exhibit any known structure, the worst case complexity is $\mathcal{O}(N)$; this is the quadratic quantum advantage being leveraged on this paper for ray tracing. Fig. 3 depicts `QSearch()`, the hybrid searching algorithm used when t is unknown and inspired on classical adaptive exponential search.

A quantum minimum finding algorithm using quantum searching has also been demonstrated to find the minimum,

with high probability, with complexity $\mathcal{O}(\sqrt{N})$, therefore maintaining the quadratic advantage with respect to a classical approach [9], [10]. A detailed description of quantum searching and minimum finding is included in the supplementary material associated with this manuscript.

Closer to the graphics field, a theoretical analysis of minimum finding utilising Grover's and a related method [9] was performed by [11]. This investigated several geometric applications, such as intersection, nearest neighbour queries, convex hull computation, separation and optimisation problems.

The idea of applying concepts from quantum computing to computer graphics was discussed by Glassner [12]. Two suggested applications were to accelerate the performance of the Z-Buffer algorithm and ray casting. For both of these problems a similar set up was proposed: a superposition over the primitives in the scene could be created, simultaneous intersections computed, and a minimum could be searched for through the application of Grover's algorithm. This would lead to a quadratic advantage in complexity versus a classical linear search of the primitives.

[13] also proposed the application of quantum computers to accelerate visibility evaluation problems, nearest neighbour queries, object-object intersection, radiosity and level of detail. These concepts were further explored by [14] which also proposed using quantum computers to accelerate the Z-Buffer algorithm, ray casting and nearest neighbour queries for photon mapping [15].

These methods make the argument that given certain conditions, such as a large number of primitives N and sufficiently high dimension of the domain $d \geq 3$, then quantum search methods are asymptotically advantageous over a classical search. In the case of unsorted search, classical computers require $\mathcal{O}(N)$ query and space complexity, whereas quantum algorithms require $\mathcal{O}(\sqrt{N})$ query complexity and $\mathcal{O}(\log(N))$ space complexity. Spatial data structures can gain an improved query complexity $\mathcal{O}(\log_d N)$, yet require $\mathcal{O}(N \log_d N)$ time to build. As N or d become very large, the building cost can be significant which can decrease the gains from using these structures.

A practical implementation of raycasting for occlusion and visibility which considered primary rays only was presented in [16]. This used an orthographic camera and rectangular primitives parallel to the image plane. These algorithms were implemented on a simulator and a simple case was run on a IBM 20 qubit quantum computer. This showed improvements compared to a classical approach, and also illustrated the probabilistic nature of quantum computing in that some visibility and occlusion measurements were incorrect.

While the previous references suggest possible applications of accelerating rendering using quantum computers, none of these, with the exception of [16], present any practical algorithms or implementations. Our paper is the first to propose a concrete approach and implementation to quantum ray tracing and to present results including direct and indirect lighting.

Other work as focused on quantum numerical integration within the context of rendering. This is a different problem than visibility evaluation through ray tracing, as addressed in this paper. These approaches are based on preparing a superposition over multiple, eventually many, rays or sub pixel locations. Then some form of quantum amplitude estimation, or quantum counting, is used to integrate the contributions of the various rays or sub pixel locations.

Quantum supersampling was investigated by [17] who proposed a hybrid quantum-classical method for integrating subpixel samples to a final pixel value. This used the canonical quantum amplitude estimation algorithm [7] combined with a classical lookup table to show improvements compared to classical Monte Carlo integration. This was validated on a simulator, a IBM five qubit machine, and a photonic quantum computer. This work is further elaborated in [18, Chapter 11], which also serves as an introductory text for quantum computing. A different integration approach was applied to the quantum supersampling problem by [19] who implemented QCoin [20], another approach for computing integrals, and gained improved results on both a simulator and a real machine. The authors also proposed that the QCoin algorithm potentially could be applied to the simulation of light transport. These results are further developed by Lu et al. [21], [22]; the authors improve the numerical integration results by proposing a new algorithm referred to as QFT-based adaptive Bayesian phase estimation.

Quantum numerical integration is a different avenue to the approach proposed in this paper and is likely to require significantly more computational resources as existing algorithms require the entire rendering context, including all scene geometry and material data, to reside on the quantum device. Additionally, the ray or path tracing operations are abstracted away behind some oracle operator, whose details are never disclosed. In some cases the authors explicitly state that ray tracing would be too demanding for current quantum hardware and simulators and substitute it by some statistical process (e.g. [21]).

III. Quantum Ray Tracing: The Algorithm

This section presents the general structure of the hybrid classical–quantum algorithm proposed for tracing a ray. Novel domain-specific optimizations made possible by the fact that rendering an image entails tracing many coherent rays are then discussed in Section III-D. Practical considerations stemming from constraints due to the current state of development of real quantum systems will be discussed in Section IV.

The quantum circuit model [23] is used to present the quantum components of the hybrid ray tracing algorithm. The stepping stone of quantum ray tracing is the adaptive exponential search algorithm presented in the supplementary material and referred to as `QSearch()` (see Algorithm 3). This is an hybrid classical plus quantum algorithm; at its core lies a quantum circuit implementing Grover's quantum search algorithm, as presented in the supplementary material. This circuit uses two operators, $\widehat{\mathcal{A}}$ and $\widehat{\mathcal{Q}}$, to prepare the state $|\Psi\rangle = \widehat{\mathcal{Q}}^r \widehat{\mathcal{A}}|0\rangle$, which maximizes the probability of measuring the index of a primitive actually intersected by the ray being traced.

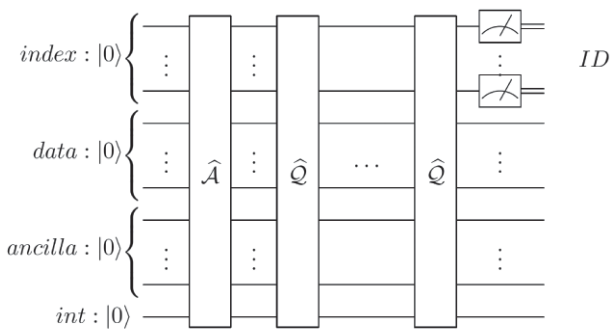


Fig. 4. Grover circuit for tracing one ray.

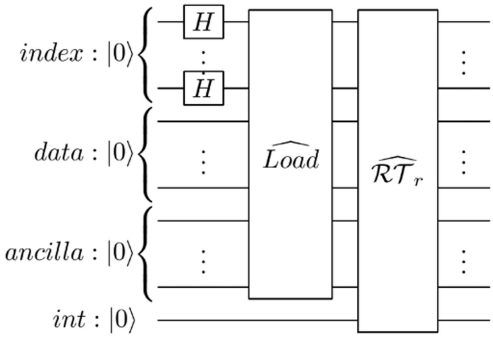


Fig. 5. Oracle $\widehat{\mathcal{A}}$ for tracing ray r.

The $\widehat{\mathcal{A}}$ operator, referred to as the oracle, encodes all problem dependent logic. $\widehat{\mathcal{A}}$ evaluates the intersections of one particular ray against all geometric primitives in the scene. Grover's operator $\widehat{\mathcal{Q}}$ is problem independent, except in the sense that it instantiates $\widehat{\mathcal{A}}$, since $\widehat{\mathcal{Q}} = -\widehat{\mathcal{A}}\widehat{\mathcal{S}_{\mathcal{O}}}\widehat{\mathcal{A}}^{-1}\widehat{\mathcal{S}_{\chi}}$.

Fig. 4 presents an high level view of Grover's circuit, with the horizontal dots between $\widehat{\mathcal{Q}}$ operators representing that any number r of such operators is being applied. The initial quantum state is $|0\rangle$, with qubits organized into four subsets:

index - n qubits indexing the scene's $N = 2^n$ geometric primitives; these qubits are measured in the end to retrieve, with high probability, the index of a primitive intersected by the ray being traced;
data - geometric data describing each primitive;
ancilla - ancillary[1] qubits, used to store intermediate data;
int - oracle's evaluation result; $|1\rangle$ if the ray intersects a primitive p, $|0\rangle$ if it does not, for all primitives p represented in the superposition.

A. The Ray Tracing Oracle

Fig. 5 presents a high level conceptual view of the oracle, $\widehat{\mathcal{A}}$, implementing, for a given ray r and all primitives $p \in S$, either $f(p, r)$ or $f_{int}(r, p, \min)$, depending on whether any intersection is accepted (shadow rays) or only those whose depth is below a given minimum threshold (other primary and secondary rays), respectively.

[1]ancillary is the term used by the quantum computing community to what would be otherwise referred to as auxiliary qubits

The circuit is structurally subdivided into 3 blocks. The set of Hadamard gates applied to the *index* register prepares an uniform superposition over all $N = 2^n$ indexes of geometric primitives. Let n, d and anc denote the number of qubits in registers *index*, *data* and *ancilla*, respectively. There is a total of $n + d + anc + 1$ qubits. The quantum state after the Hadamard gates is given by:

$$|\Psi_H\rangle = \mathcal{H}^{\otimes n} \otimes \mathcal{I}^{\otimes(d+anc+1)} \left(|0\rangle^{\otimes n}|0\rangle^{\otimes d}|0\rangle^{\otimes anc}|0\rangle\right)$$
$$= \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} \left(|i\rangle|0\rangle^{\otimes d}|0\rangle^{\otimes anc}|0\rangle\right)$$

The $\widehat{Load}$ operator performs classical data uploading by setting the *data* qubits, such that they binary encode the numerical parameters describing the N geometric primitives (e.g. their vertex positions) indexed by *index*. It is a logic circuit implementing the required Boolean functions using unitary reversible quantum logic gates. Each qubit in the *data* register is a Boolean function of the qubits in the *index* register. Since the state of the latter is an uniform superposition of N basis states, *data* also encodes the numerical parameters of all N geometric primitives.

Representing a geometric primitive requires an arbitrary number of qubits d depending on the primitive type and the representation resolution. $\widehat{Load}$ implements these d Boolean functions:

$$data_j = load_j(index_0, \ldots, index_{n-1}), j = 0 \ldots d-1,$$

where $data_j$ represents each of the qubits in the *data* register. The resulting state is

$$|\Psi_L\rangle = \widehat{Load} \otimes \mathcal{I}\left(|\Psi_H\rangle\right)$$
$$= \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} \left(|i\rangle|p_i\rangle|0\rangle^{\otimes anc}|0\rangle\right),$$

where $|p_i\rangle$ stands for the numerical parameters of the i^{th} geometric primitive. Encoding each of these functions might, in the worst case, require a circuit with depth $\mathcal{O}(N)$. This worst case scenario is seldom met, given that well established Boolean algebra rules for circuit simplification are aggressively applied [24]. Nevertheless, classical data upload is a major challenge, which may eventually compromise quantum computational complexity gains. Therefore, this paper claims a quantum query complexity advantage, rather than a computational advantage over a classical approach. Data uploading will be further discussed in Section VI.

The ray tracing intersection operator $\widehat{RT}_r$ evaluates which of the encoded primitives are intersected by ray $r : |int\rangle$ is set accordingly. Its implementation depends on various practical design decisions discussed in the supplementary material and Section IV. The resulting state is:

$$|\Psi_{\mathcal{A}}\rangle = \widehat{RT}_r\left(|\Psi_L\rangle\right)$$
$$= \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} \left(|i\rangle|p_i\rangle|0\rangle^{\otimes anc}|f(p_i, r)\rangle\right)$$

```
1: procedure QINTERSECT(ray, S, maxD)
2:     Â_min ← build_Amin (ray, S, maxD)
3:     p, found ← QSearch (Â_min, f_min())
4:     rt_info, depth = eval_rt_info (p, ray, S)
5:     return (p, found, rt_info, depth)
6: end procedure
```

Fig. 6. QIntersect(ray, S, maxD): Intersecting a ray.

```
1: procedure QTRACE(r, S, maxD, #IT)
2:     cDepth ← maxD
3:     solution ← −1; it ← 1
4:     while it ≤ #IT do        ▷ improve on current depth
5:         p, fnd, rt_info, d ← QIntersect (r, S, cDepth)
6:         if fnd then
7:             cDepth ← d; solution ← p
8:         end if
9:         it ← it + 1
10:    end while
11:    return solution
12: end procedure
```

Fig. 7. `QTrace()` : Tracing a ray r.

B. Grover's Operator

Grover's operator is defined as $\widehat{\mathcal{Q}} = -\widehat{\mathcal{A}}\widehat{\mathcal{S}_{\mathcal{O}}}\widehat{\mathcal{A}}^{-1}\widehat{\mathcal{S}_{\chi}}$. $\widehat{\mathcal{S}_{\chi}}$ flips the sign of the amplitudes of marked solutions, i.e, those with the *int* qubit equal to $|1\rangle$, and $\widehat{\mathcal{S}_{\mathcal{O}}}$ performs a reflection over the amplitudes' average. $\widehat{\mathcal{Q}}$ implementation, given $\widehat{\mathcal{A}}$, is straightforward (see the supplementary material for details).

C. Hybrid Ray Tracing Algorithm

`QIntersect()` (Fig. 6) is the hybrid method responsible for finding an intersection between a ray and the scene's primitives, resorting to `QSearch()` .

`QIntersect()` builds the oracle $\widehat{\mathcal{A}}_{min}$ based on the current ray, the scene and the maximum allowed depth (depth is hereby understood as the distance from the ray's origin to the intersection point). Subsequently, `QSearch()` searches for an intersection within the allowed depth range. Upon termination, `QIntersect()` returns whether or not a valid intersected primitive was found, together with classically evaluated information on this intersection, such as primitive ID, 3D point, normal and depth. It returns any intersection whose depth is less than `maxD`. This is enough for shadow rays, but not for other rays, where the minimum depth intersection is required. `QTrace()` (Fig. 7) is responsible for tracing a single ray. Inspired in the quantum minimum finding algorithm (see supplementary material), it calls `QIntersect()` a fixed number ($\#IT$) times, updating the current depth when an intersection is found.

The hybrid quantum-classical algorithm used by `QTrace()` for each ray is probabilistic. At each iteration `QIntersect()` (line 5) will return either:

- *fnd* $== False$ – no primitive was found which intersects this ray at a depth less than the current depth. Such primitive might exist, but `QIntersect()` has returned a false negative, which happens with probability $\overline{p}_{QS}$. `QTrace()` performs a fixed, user supplied, number of iterations ($\#IT$) to avoid stopping on false negatives. However, if this was a true negative (i.e. there are no further primitives intersected at a smaller depth), then the additional iterations are useless computational overhead.
- *fnd* $== True$ – an intersection was found at a depth less than the current depth; current depth is updated to the new value. The newly found primitive might not be the one nearest to the ray's origin, i.e., the one with minimum depth. In fact, all primitives intersected by the ray with a depth below the current limit have the same probability of being returned by `QIntersect()` . It is not possible to determine whether the last found primitive is the true minimum without verifying all primitives, which would be prohibitive. If a solution is found, then `QTrace()` continues iterating until falling to the previously described case.

D. Optimizations

This section proposes two optimizations to the hybrid quantum ray tracing algorithm, by exploiting two sources of information arising from the data. First, image plane coherence is exploited to improve convergence onto the minimum depth solution, i.e., the visible geometric primitive at each pixel. Second, a principled termination criterion based on successive non intersections for the same ray is proposed, enabling autonomously stopping iterating `QTrace()` on a per ray basis.

Rendering an image entails tracing various types of rays, such as primary, specular and shadow rays. Tracing rays of the same type in a single pass, i.e. in a breadth-first manner, allows exploiting optimizations due to coherence among neighbouring rays. The actual algorithm used to render many rays of the same type is similar to `QTrace()` , but receives as input a list of rays (and respective maximum depths), rather than a single ray. Looping over all rays in the input list takes place within the `while` loop (line 4 of Fig. 7), thus performing a breadth-first minimum searching.

1) Gathering Neighboring Data: Natural images often exhibit image plane coherence, meaning that some measure is locally constant, i.e., varies smoothly within each pixel's neighborhood. Examples of such measures are radiance and the geometric primitives that project onto each pixel.

We exploit the latter: it is highly probable that the geometric primitive projecting onto any pixel (x,y) also projects onto at least one of its 4 von Neumann neighbours. After each evaluation of `QIntersect()` (line 5) the ray is classically intersected against the primitives currently projected onto each of its 4 neighbours in the image plane. If any such primitives are intersected and if depths are less than the current depth, then the one at minimum depth is taken as the one intersecting this ray, updating current minimum depth accordingly. Local information is shared across the image plane, accelerating convergence

towards minimum depth, therefore reducing the number of `QTrace()` iterations on a per ray basis. Furthermore, since the update is performed '*in place*' a diffusion process occurs, where previously found true minimum depth primitives can spill to contiguous regions.

2) Termination Criterion: `QTrace()` performs a fixed, user-supplied, number of iterations ($\#IT$) in order to both continue searching for a true minimum depth and avoid stopping at false negatives. The proposed optimization stochastically decides whether to terminate iterating based on an estimate of $\overline{p}_{QS}$, the probability that `QSearch()` , and consequently `QIntersect()` , return a false negative. This estimate is derived next.

For any given iteration l of `QSearch()` the number of evaluations of Grover's operator $\widehat{\mathcal{Q}}$ is randomly selected with uniform distribution, such that $r_l \in \{1 \dots M_l\}$. The expected value for r_l is therefore $M_l/2$. The probability that iteration l returns a solution for any particular (unknown) number of solutions t is $p_l(t) = \sin^2((M_l + 1)\theta_a)$. The probability of being unsuccessful, i.e., of iteration l reporting a false negative when in fact there are t solutions, is

$$\overline{p}_l(t) = 1 - p_l(t) = \cos^2((M_l + 1)\theta_a) \tag{2}$$

t is unknown, but let $p_N(i)$ be the probability mass function, with $p_N(i) = P(t = i), i = 1 \dots N$. The probability of a false negative for any possible t is then given by

$$\overline{p}_l = \sum_{t=1}^{N} \left(p_N(t) \cos^2((M_l + 1)\theta_a)\right) \tag{3}$$

In order to return a false negative, `QSearch()` must iterate L times, with the iteration index l taking all values $l \in \{1 \dots L\}$ and M_l taking all corresponding values from set $\mathcal{M} = \{\lceil c^1 \rceil, \lceil c^2 \rceil, \dots, \min(\lceil c^L \rceil, \lceil \sqrt{N} \rceil)\}$. L and $\mathcal{M}$ can be easily computed given c and N. The probability that the algorithm returns a false negative after performing all L iterations is therefore given by a variant of the Poisson binomial distribution:

$$\begin{aligned} \overline{p}_{QS} &= \prod_{M_l \in \mathcal{M}} \overline{p}_l \\ &= \prod_{M_l \in \mathcal{M}} \left[\sum_{t=1}^{N} \left(p_N(t) \cos^2((M_l + 1)\theta_a)\right) \right] \end{aligned} \tag{4}$$

Let $p_N(t) = 1/N$, i.e., the number of primitives potentially intersected by any ray is assumed to be uniformly distributed between 1 and N ($t = 0$ would not be a false negative). Equation 4 becomes

$$\overline{p}_{QS} = \prod_{M_l \in \mathcal{M}} \left[\frac{1}{N} \sum_{t=1}^{N} \cos^2((M_l + 1)\theta_a) \right] \tag{5}$$

Given c and N an estimate of $\overline{p}_{QS}$ can be precomputed and used in `QTrace()` termination criterion as follows:

- if an intersection is found, update the visible primitive ID and the minimum depth threshold and trace the ray again one further iteration, to ensure that eventual intersections occurring at shorter depths can still be found;
- if no intersection is found, then add 1 to the number of consecutive observed non-intersections for this ray (ni). Then draw a random number, ξ, uniformly distributed in $[0..1)$; if $\xi \leq (\overline{p}_{QS})^{ni}$ then keep iterating, otherwise terminate.

The new hybrid ray tracing algorithm dispenses with parameter $\#IT$ and will continue iterating for non terminated rays until there is none. Empirical results (Section IV-F2) clearly show that $\mathcal{O}(\sqrt{N})$ query complexity is maintained with much better constants than the previous non-optimized algorithm.

IV. Quantum Ray Tracing Analysis

This section experimentally verifies whether the proposed hybrid algorithm complies with the theoretical expectation of providing a quadratic query complexity gain with respect to a classical approach. Performance scalability with the number of geometric primitives and rendering error with the number of minimum searching iterations are analysed. Results presented in Sections IV-D and IV-E do not include the optimizations discussed in Section III-D; their impact on performance and error is analysed in Section IV-F. Experiments are performed in the context of a Whitted-style renderer, which shoots primary and specular rays, as well as shadow rays that assess the visibility of point light sources. This is, to the best of the authors' knowledge, the first image renderer based on a hybrid quantum-classical ray tracing routine.

A. Practical Quantum Ray Tracing

The current stage of development of quantum hardware imposes severe constraints on the depth and number of gates of quantum circuits [25]. The proposed ray tracing algorithm is too deep to allow robust execution on current quantum hardware. Even validating it on classical simulators of quantum circuits in useful time requires constraining the problem in order to make the algorithm practical.

The supplementary material associated with this paper presents the oracle's quantum circuit in detail and gives a detailed description and justification of the restrictions and design options imposed in order to develop a practical quantum ray tracing algorithm. Here we enumerate the main points:

- all numerical values (e.g. coordinates) are represented using fixed point representation;
- axis aligned rectangles (AAR) are the only supported geometric primitives;
- the hybrid quantum classical renderer used for these experiments classically computes the ray intersection coordinates with the plane embedding each AAR. Whereas this option seems to compromise this paper's goal of presenting a quantum ray tracer algorithm, we argue that:
  - the intersection evaluation is still performed using the quantum circuit (see supplementary material);
  - there is no fundamental reason why in the near future these computations can not be performed by the quantum system. As the technology of quantum computers

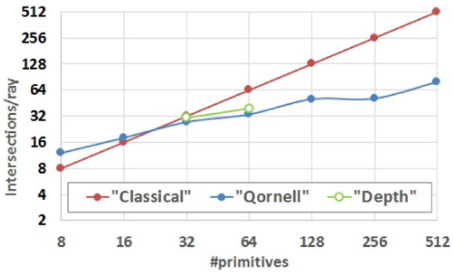


Fig. 8. Qornell Box: number of oracle evaluations per ray with varying numbers of geometric primitives.

evolves so will the supported circuit depth and width, allowing the migration of these operations to the quantum machine;

- this allows us to practically evaluate and optimize the performance of the core parts of the quantum ray tracing algorithm before sufficiently powerful quantum computers are available.

B. Experimental Setup

All experiments were performed using Qiskit (ver. 0.18.1), an open source SDK for quantum computing at the level of algorithms and circuits. Qiskit allows executing the quantum circuits both on a real quantum backend or on a simulator. The latter has been used in these experiments due to the limitations of current quantum computers. In order to maintain the number of required qubits within the simulator supported range (≤ 25) the world space is restricted to a $16 \times 16 \times 16$ integer coordinate volume; this quantization of the 3D world coordinates is clearly perceivable in all rendered images and severely constrains the complexity of the 3D scenes in terms of the number and relative positioning of geometric primitives. Additionally, simulation times are long and directly proportional to the number of rays; all images were rendered at 128×128 resolution to guarantee acceptable (a few hours) rendering times. The Whitted-style renderer processes rays in a breadth first manner, by processing all rays of the same type before proceeding to the next level of the rays' trees.

The query complexity of Grover's algorithm (i.e., the number of evaluation of the oracle $\widehat{\mathcal{A}}$) depends on the cardinality of the search domain and the number of actual solutions, denoted by N and t, respectively. In the context of ray tracing N is the number of geometric primitives present in the scene and t is the number of geometric primitives actually intersected by the ray below the maximum acceptable depth; t is also referred to as the depth complexity.

We perform comparisons against a classical deterministic renderer which implements the same ray primitive intersections with the same geometrical limitations, but processes all N primitives for each ray.

We use two scenes to compare the proposed approach: the Qornell Box (Fig. 1(a)) and the Depth Complexity scene (Fig. 12). Detailed descriptions are provided in the supplementary material.

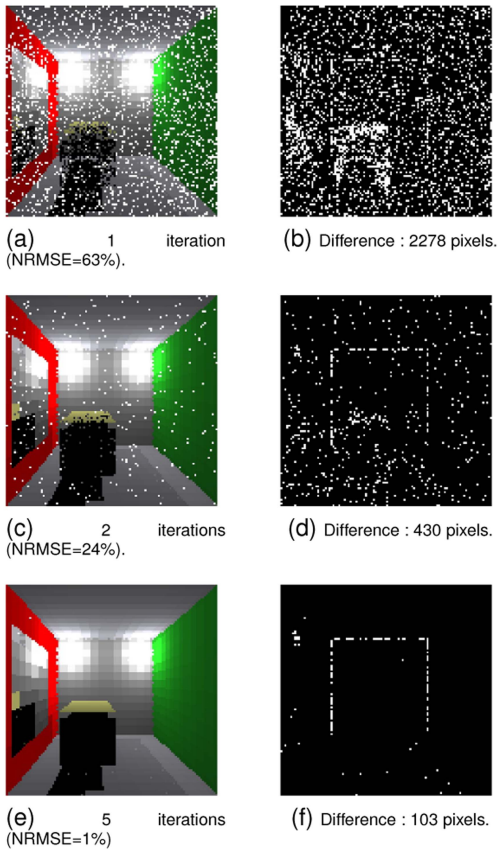


Fig. 9. `QTrace()` number of iterations - Qornell Box with 64 primitives.

TABLE I
PERFORMANCE AND ERROR METRICS

Performance metrics	
#Rays	Total number of rays traced
#C_Int	Classical: number of calls to the ray/geometry intersection method; equals the number of rays times the number of geometric primitives,since no acceleration structures are used to impose a spatial ordering
#Eval	Quantum: number of evaluations of the oracle $\widehat{\mathcal{A}}$
#Int	#C_Int + #Eval: total number of intersection evaluations (classical + quantum); this is required because `QSearch()` classically verifies the results measured from the quantum circuit (lines 3 and 14, algorithm 3)
Int/Ray	average number of intersections per ray (#Int / #Rays)
#It	number of iterations of the minimum finding algorithm
#Cpix	number of pixels updated by exploiting image space coherence (sec. III-D1)
Error metrics Quantum rendered image w.r.t. classical	
#NRMSE	Normalized Root Mean Squared Error
#Dpix	number of pixels with different RGB values
%Dpix	percentage of pixels with different RGB values

C. Performance and Error Metrics

The expected advantage of hybrid quantum/classical ray tracing over classical ray tracing is a quadratic improvement on the asymptotic number of intersection evaluations as the number of the scene's geometric primitives increases. Table I describes

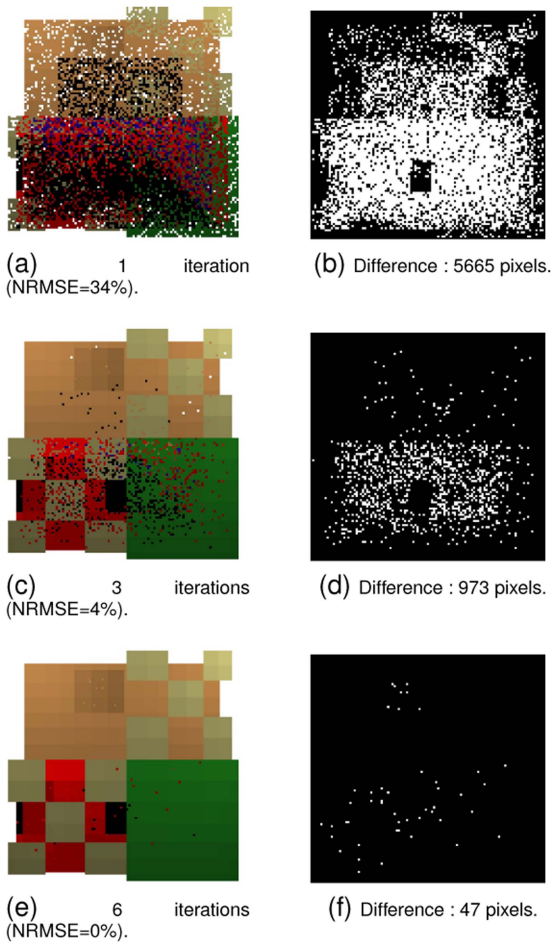


Fig. 10. `QTrace()` number of iterations - Depth Complexity Scene with 64 primitives.

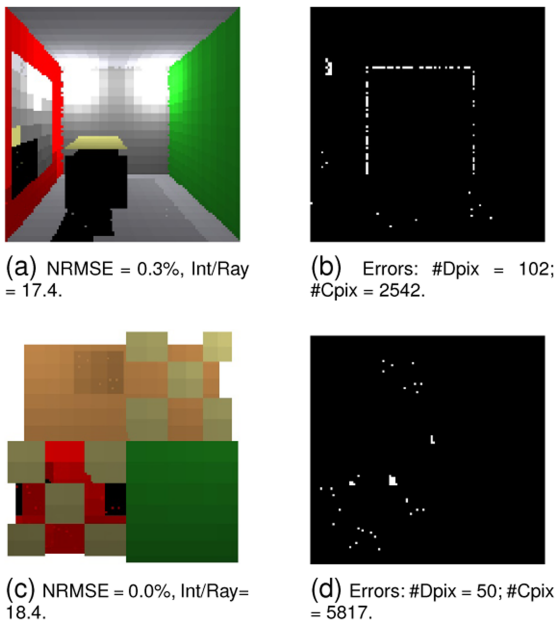


Fig. 11. Gathering neighbouring data: single iteration.

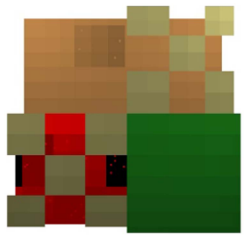


Fig. 12. Depth Complexity scene: optimized ray tracer ($N = 64$).

TABLE II
SCALABILITY WITH THE NUMBER OF PRIMITIVES

Scene	N	Classical #Int	Classical $\frac{\text{Int}}{\text{Ray}}$	Quantum #Int	Quantum $\frac{\text{Int}}{\text{Ray}}$
Qornell	8	341 K	8	500 K	12.0
	16	669 K	16	731 K	18.0
	32	1339 K	32	1112 K	27.4
	64	2678 K	64	1366 K	33.6
	128	5356 K	128	2045 K	50.4
	256	10712 K	256	2075 K	51.3
	512	21423 K	512	3240 K	79.7
Depth	32	1207 K	32	1132 K	30.7
	64	2413 K	64	1455 K	39.4

the metrics used to quantify and compare the classical and the quantum renderers' performance.

Since the classical renderer is fully deterministic, the classically rendered images are used as the reference result for the images obtained with the quantum renderer under the same rendering conditions. Table I also describes the three metrics used to quantify the error of the quantum rendered images with respect to reference.

D. Scalability With the Number of Primitives

Table II presents the average number of intersection evaluations per ray with varying number of geometric primitives. While this figure increases linearly for the classical renderer, it grows with the square root of the number of geometric primitives in the quantum case. It is clearly shown that the quantum renderer scales quadratically better than the classical renderer; Fig. 8 visually illustrates this trend.

Fig. 1(b) presents the rendered image for the 64 primitives case. This is mostly identical to the reference case (Fig. 1(a)) except for:

1) a very reduced number of pixels scattered across the image plane — due to the stochastic nature of the quantum minimum finding algorithm: a false positive (non intersecting primitive) is occasionally measured, despite Grover's amplitude amplification. The ensuing classical verification transforms this measurement into a non-intersection by verifying that the ray does not intersect that primitive – the resulting false negative appears as an erroneous pixel.
2) a significant number of pixels in the intersection between the back wall and the adjacent walls — this is due to Z fighting: the two intersecting walls are intersected by the

TABLE III
STATISTICS AND ERROR MEASURES FOR DIFFERENT NUMBER OF ITERATIONS (BOTH SCENES WITH 64 GEOMETRIC PRIMITIVES)

Scene	It	#Int	$\frac{\text{Int}}{\text{Ray}}$	Error Metrics NRMSE	#Dpix	%Dpix
Qornell	1	631 K	17.0	63%	2278	14%
	2	905 K	22.6	24%	430	3%
	3	1139 K	28.1	10%	146	1%
	4	1366 K	33.6	4%	107	1%
	5	1593 K	39.2	1%	103	1%
Depth	1	552 K	15.9	34%	5665	35%
	2	789 K	21.7	13%	2556	16%
	3	1017 K	27.6	4%	973	6%
	4	1240 K	33.6	1%	304	2%
	5	1455 K	39.4	0%	95	1%
	6	1668 K	45.2	0%	47	0%

primary ray at exactly the same (integer) ray length. The measurement in quantum searching stochastically selects one of the geometric primitives with equal probabilities, resulting in the perceived visual noise.

These experimental results corroborate the theoretical claim that the number of intersection evaluations is $\mathcal{O}(\sqrt{N})$, a clear advantage over the $\mathcal{O}(N)$ complexity of the classical approach.

E. Minimum Finding: Number of Iterations

Table III presents a range of performance and error metrics for both scenes with 64 geometric primitives. Results are presented for different numbers of iterations of the minimum finding algorithm (algorithm 7).

Table III presents how many intersection evaluations (quantum plus classical) are required, on average, per ray to render the respective image within a given error bound. By comparing #Dpix for both scenes it becomes clear that the number of intersection evaluations per ray is dependent on the depth complexity. #Dpix is much higher for the Depth Complexity scene than for the Qornell Box scene, for the same number of iterations, because the former exhibits higher depth complexity. Since the number of solutions, t, satisfying the intersection oracle $\widehat{\mathcal{A}}$ is larger for the initial iterations, more iterations are required to converge towards the minimum depth solution.

NRMSE is larger for the Qornell Box scene than the Depth Complexity scene, because the former includes a mirror spawning an additional layer of specular rays. Additionally, #Dpix does not go below 100 for the Qornell Box (at 128x128 resolution) due to Z fighting (see Section IV-D).

Figs. 9 and 10 further illustrate that errors concentrate in image regions with larger depth complexity. As more iterations are executed the minimum threshold progresses towards the correct value.

Table IV presents statistics for primary rays for each individual iteration out of a total of 6 iterations for the Depth Complexity scene. %Dpix decreases with the iteration count, as expected. However, the number of intersections per ray at each iteration increases with the iteration count. The number of primitives satisfying the oracle criterion diminishes as more iterations are performed and the minimum threshold gets smaller, reaching zero when the visible primitive is found. Grover's adaptive search algorithm, `QSearch()` , will keep searching for a non-existent solution, reaching the maximum number of allowed oracle evaluations. Table IV shows that after the first iteration the correct primitive has not been found for only 5853 pixels; however all 16384 pixels are processed with `QSearch()` , which will maximize computations for 10531 of these pixels for which a better solution that does not exist. This is a major source of wasted computation.

TABLE IV
INDIVIDUAL ITERATION STATISTICS FOR THE DEPTH COMPLEXITY SCENE WITH 6 ITERATIONS – PRIMARY RAYS ONLY

#Rays	It	#Int	$\frac{\text{Int}}{\text{Ray}}$	%Dpix	Dpix
16384	1	139 K	8.5	35.7%	5853
	2	178 K	10.8	16.1%	2630
	3	199 K	12.2	6.4%	1047
	4	208 K	12.7	1.9%	306
	5	212 K	12.9	0.5%	83
	6	213 K	13.1	0.3%	47

TABLE V
STATISTICS AND ERROR MEASURES FOR DIFFERENT NUMBER OF ITERATIONS WHILE GATHERING NEIGHBOURING DATA (BOTH SCENES WITH 64 GEOMETRIC PRIMITIVES)

Scene	It	$\frac{\text{Int}}{\text{Ray}}$	NRMSE	#Dpix	#Cpix
Qornell	1	17.4	0.3%	102	2542
	2	23.0	0,3%	96	0
	3	28.7	0.3%	97	0
Depth Complexity	1	18.4	0.0%	50	5817
	2	24.2	0.0%	29	4
	3	29.9	0.0%	17	0

F. Optimized Hybrid Ray Tracing Analysis

This section presents results which exploit the two optimizations proposed in Section III-D, and improve performance compared to the previous section.

1) Gathering Neighboring Data: Table V presents the average ratio of intersections per ray for different numbers of optimized `QTrace()` iterations, together with NRMSE, #Dpix and #Cpix. By comparing with Table III it is clear that the optimization process requires a single iteration to update most pixels to the correct visible geometric primitive. This substantial improvement in the convergence rate is further illustrated in Fig. 11, where rendered images for both scenes are presented for a single optimized iteration; visual comparison with the first row of Figs. 9 and 10 clearly demonstrate the achieved gain.

However, there is a subtle performance penalty with this approach: each optimized iteration requires evaluating more classical intersections than in the non-optimized case. For each pixel, the primitives projecting onto its four immediate neighbours have to be classically checked for intersection against the current ray. This increase in the rate of intersections per ray can be observed by comparing columns $\frac{\text{Int}}{\text{Ray}}$ of Tables III and V. In practice, however, the optimized process requires far fewer iterations to converge to a similar quality result.

TABLE VI
SCALABILITY WITH THE NUMBER OF PRIMITIVES

Scene	N	Classical	Quantum				
			No-Opt		Opt		
		$\frac{\text{Int}}{\text{Ray}}$	$\frac{\text{Int}}{\text{Ray}}$	#Dpix	$\frac{\text{Int}}{\text{Ray}}$	#Dpix	#Cpix
Qornell	8	8	12.0	18	9.8	11	912
	16	16	18.0	94	14.4	89	1077
	32	32	27.4	93	21.3	92	1316
	64	64	33.6	107	22.1	103	2438
	128	128	50.4	108	32.6	108	2197
	256	256	51.3	295	33.7	144	4448
	512	512	79.7	157	51.8	124	3445
Depth	32	32	30.7	201	22.0	2	5456
	64	64	39.4	95	22.5	21	5781

2) Termination Criterion: Table VI presents performance and error metrics for the proposed termination criterion combined with gathering data from neighboring pixels. For the sake of clarity the number of intersections per ray is also shown for the quantum non-optimized case (taken from Table II) and for the classical approach.

The following conclusions can be drawn:

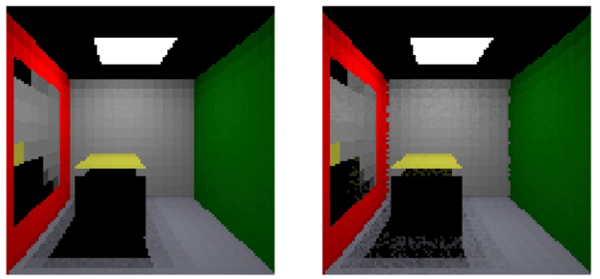
- the optimized self terminating approach is still $\mathcal{O}(\sqrt{N})$, but with significantly better constants than the non-optimized version. This improvement comes from the fact that rays are terminated on early iterations, thus reducing the overall workload;
- the improvement on the number of intersections per ray over the non-optimized quantum version increases significantly with N; this indicates that much more significant gains may be achievable as technology improvements allow for more complex scenes (larger N);
- there is also a gain in image quality (#Dpix), which is particularly evident for the Depth Complexity scene (Z-fighting in the back wall edges imposes a lower bound on #Dpix for the Qornell Box). This improvement in image quality is due to the gathering of neighbouring data among pixels.

Figs. 1(c) and 12 present the images rendered with the optimized hybrid ray tracer.

V. BEYOND WHITTED RAY TRACING

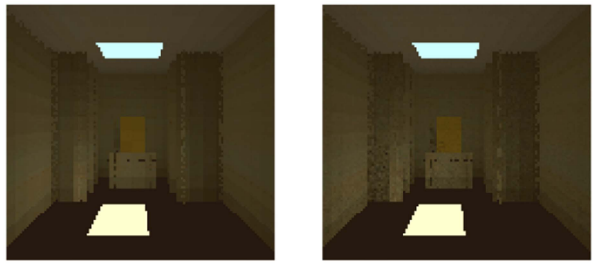
This section simulates light transport phenomena beyond the constraints associated with Whitted ray tracing, namely area light sources and global illumination. In both these cases classical Monte Carlo sampling is used to generate samples, while the proposed algorithm is used to compute visibility.

Fig. 13 shows results for the Qornell Box lit from an area light source in the ceiling, using 32 light samples (shadow rays) per pixel. A classically rendered image using 512 light samples is also displayed as a reference. The umbra and penumbra regions are clearly distinguishable, even though they are noisy. This noise has two causes. First, a smaller number of light samples per pixel is used since computing 512 shadow rays per pixel would require unacceptable quantum simulation times. Second, the reuse of neighboring data optimization is not performed for shadow rays, therefore false negatives occur more frequently than for primary and specular rays. An extension of this optimization to secondary and shadow rays based on the same algorithm is possible, and we leave this to future work. Despite the noise, these results clearly demonstrate that the hybrid approach handles area light sources.



(a) Classical reference: 512 light samples. (b) Quantum rendering: 32 light samples.

Fig. 13. Area light source lighting: Qornell Box with 64 primitives.



(a) Classical reference: 512 VPLs per pixel. (b) Quantum rendering: 4 VPLs per pixel.

Fig. 14. Indirect lighting using single bounce VPLs per pixel.

Fig. 14 presents results for indirect lighting in a temple scene consisting of 32 primitives. This scene is lit from directional light coming through a small hole in the ceiling. From this, single bounce indirect illumination is computed from a unique set of Virtual Point Lights (VPLs) per pixel. A single primary ray per pixel is shot, which leads to some noise due to Z fighting associated with fixed point quantization (see Section IV-B). However, the contribution of indirect lighting from the 4 VPLs per pixel using the hybrid quantum-classical ray tracer (Fig. 14(b)) is clearly converging towards the classical reference with 512 VPLs per pixel (Fig. 14(a)).

The number of intersections per ray using the optimized hybrid ray tracer for the area light source scene is 21.8 compared to 64 classically, and indirect lighting requires 22.4 compared to 32 classically. Performing operations classically has $\mathcal{O}(N)$ query complexity as expected, whereas the hybrid quantum-classical approach leads to the expected quadratic improvement in the number of intersections required.

VI. DISCUSSION

This article proposes a hybrid quantum-classical ray tracer which exhibits a quadratic query complexity advantage over classical ray tracing in the absence of spatial sorting data

structures. There are however limitations and challenges whose impact on the feasibility and quantum advantage of the proposed algorithm must be discussed.

Practical limitations – a prominent practical limitation is the classical evaluation of the intersection points between the ray and the planes embedding each geometric primitive – section IV-A. This stage of the intersection procedure becomes $\mathcal{O}(N)$, whereas it would be $\mathcal{O}(1)$ if performed by the quantum hardware. As previously discussed, there are no fundamental reasons why these computations cannot be migrated to the quantum circuit. There are practical reasons: the resulting circuit width and depth would prevent mapping it to either a real device or a simulator. Identical reasons constrained the representation of all data, including the description of the 3D world, to small fixed point values, represented using a small number of qubits. These constraints place a hard limit on the complexity of the scenes that can be processed and produce quantization artifacts perceivable in the rendered images. However, as technology advances and quantum volume [26] increases these constraints will be lifted once floating point representations become practical.

Data loading – a stronger argument against the proposed approach is the cost of loading the classical data describing the scene, performed by the oracle $\widehat{\mathcal{A}}$. The term "oracle" is used in complexity theory to refer to a data access operator whose application is $\mathcal{O}(1)$, or, equivalently, to mean that its implementation cost is not relevant to query complexity analysis [27]. Quantum query complexity relates to the number of queries to this $\mathcal{O}(1)$ oracle. Time complexity, on the other hand, is often measured as the number of gates or as the depth of the quantum circuit. Since quantum circuit depth for Grover's algorithm is directly proportional to the number of queries made to the oracle, query complexity is in fact a lower bound for time complexity – the two being the same if each query is indeed $\mathcal{O}(1)$ in time.

It has been shown in the previous sections that the proposed ray tracing algorithm is $\mathcal{O}(\sqrt{N})$ in the number of queries. Each query to $\widehat{\mathcal{A}}$ loads the data describing the scene, providing access to a set of N indexed numbers x_i as in:

$$\widehat{\mathcal{A}}\left(\frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}|i\rangle|b\rangle\right)=\frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}|i\rangle|b\oplus x_i\rangle.$$

Two different architectures have been proposed to implement such oracles: circuit-based or QRAM-based. The former has been used in the previous sections. Its depth is $\mathcal{O}(N)$, which, for a $\mathcal{O}(\sqrt{N})$ query complexity, results on a $\mathcal{O}(N\sqrt{N})$ time complexity for the overall algorithm, which is worst than classical linear searching. The QRAM is, currently, a hypothetical device that stores classical data, able to answer queries in quantum superposition in $\mathcal{O}(\log N)$ time [28], which would allow for a $\mathcal{O}(\sqrt{N}\log N)$ time complexity, quasi-quadratically better than classical linear search.

Probabilistic measurements – quantum algorithms have a probabilistic nature, therefore there is a non zero probability, $\overline{p}_{QS}$ (Section III-C), of finding no intersection when in fact the ray intersects some primitive(s). $\overline{p}_{QS}$ can be made arbitrarily low by increasing the number of iterations, which will not impact on the query complexity, but will increase execution time. Experimental results (Section IV-F) demonstrate that the proposed optimizations (Section III-D) dramatically reduce $\overline{p}_{QS}$ while maintaining the quadratic query complexity advantage. Constants associated with the algorithm's asymptotic behaviour are significantly improved w.r.t. the non optimized version, ensuring that the quantum advantage does not vanish due to the probabilistic nature of quantum measurements.

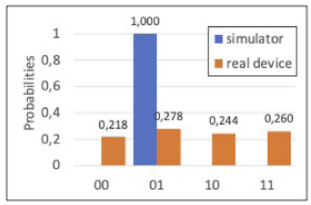


Fig. 15. Simulator vs. Real Device Measurement Probabilities: 1024 shots.

Noise – a quantum quadratic query complexity advantage is demonstrated using a simulator under ideal conditions, i.e. in the absence of noise. Noise levels of current quantum computers do not allow executing the proposed ray tracing algorithm: noise from decoherence would overwhelm the results.

To allow execution both on the simulator and on a real device with 27 qubits and quantum volume equal to 32, a minimal scene with 4 geometric primitives was designed together with a primary ray intersecting only one of primitive. The simulated circuit uses 19 qubits with a gate depth of 1185. After transpilation for execution on the real device, 27 qubits are used and the gate depth is 11096, at least 3 orders of magnitude larger than what can be reliably executed on current devices. Fig. 15 presents the measured state histogram for 1024 shots. Results on the real device approach a uniform distribution over all possible states, precluding its utilization with such deep circuits.

Grover's algorithm has been studied under noisy conditions [29]. The success probability decays exponentially with the number of qubits and circuit depth. Future fault tolerance and error correcting codes are expected to allow execution of deep quantum algorithms while preserving quantum advantage. The quantum threshold theorem asserts that computation errors can be made arbitrarily low at the cost of additional resources. If noise remains below a given threshold then the required additional resources grow with $\log^c(p(n))$, where $p(n)$ is the number of gates. Therefore the number of oracle evaluations (or, equivalently, $\#Int$) will grow as $\mathcal{O}(\log(p(n))\sqrt{N})$, below the $\mathcal{O}(N)$ classical asymptotical query complexity.

VII. Conclusion and Future Work

This paper presents the first hybrid quantum-classical algorithm for ray tracing, improving the query complexity of ray primitive intersection operations. A classical algorithm requires $\mathcal{O}(N)$ intersection evaluations per ray (without 3D spatial ordering), while the hybrid algorithm requires $\mathcal{O}(\sqrt{N})$. Our algorithms improve performance by exploiting spatial coherence to minimize the impact of quantum randomness and by a stopping criterion for the searching process. We present results

for Whitted-style renderer, and also demonstrate additional light transport phenomena.

We plan to extend this work in multiple directions. The first is to further investigate quantum numerical integration. This would be another step towards a fully quantum rendering system. However, this would require the quantum machine to store the entire rendering context which may require a prohibitively large quantum volume for medium-term quantum systems (Section II). Second, as the quantum volume available on real machines increases we intend to integrate floating point arithmetic and fully evaluate intersections using the quantum hardware. Finally, we intend to integrate new developments allowing efficient loading and access to unstructured classical databases. These include both developments on QRAM and circuit synthesis, which will facilitate closing the efficiency gap between query and time complexities.

REFERENCES

[1] P. Benioff, "The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by turing machines," *J. Stat. Phys.*, vol. 22, no. 5, pp. 563–591, 1980.

[2] R. P. Feynman, "Simulating physics with computers," *Int. J. Theor. Phys.*, vol. 21, no. 6, pp. 467–488, 1982.

[3] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, pp. 1484–1509, 1997.

[4] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. 28th Annu. ACM Symp. Theory Comput.*, 1996, pp. 212–219, doi: 10.1145/237814.237866.

[5] H. Samet, "Sorting in space: Multidimensional, spatial, and metric data structures for computer graphics applications," in *Proc. ACM SIGGRAPH Asia 2010 Courses*, 2010, pp. 1–52, doi: 10.1145/1900520.1900523.

[6] P. Hoyer, J. Neerbek, and Y. Shi, "Quantum complexities of ordered searching, sorting, and element distinctness," *Algorithmica*, vol. 34, pp. 429–448, 2002.

[7] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, "Quantum amplitude amplification and estimation," *Contemporary Math.*, vol. 305, pp. 53–74, 2002.

[8] D. Deutsch, "Quantum theory, the church–turing principle and the universal quantum computer," *Proc. Roy. Soc. London Math. Phys. Sci.*, vol. 400, 1985, Art. no. 1818.

[9] C. Dürr and P. Høyer, "A quantum algorithm for finding the minimum," 1996, *arXiv:quant-ph/9607014*.

[10] A. Ahuja and S. Kapoor, "A quantum algorithm for finding the maximum," 1999, *arXiv:quant-ph/9911082*.

[11] K. Sadakane, N. Sugawara, and T. Tokuyama, "Quantum computation in computational geometry," *Interdiscipl. Inf. Sci.*, vol. 8, no. 2, pp. 129–136, 2002.

[12] A. Glassner, "Quantum computing. I," *IEEE Comput. Graph. Appl.*, vol. 21, no. 4, pp. 84–92, 2001, doi: 10.1109/38.953464.

[13] M. Lanzagorta and J. Uhlmann, *Quantum Computer Science*. San Rafael, CA, USA: Morgan & Claypool, 2009.

[14] S. Caraiman, "Quantum computer graphics algorithms," *Buletinul Institutului Politehnic din Iasi Sectia Automatica si Calculatoare*, vol. 62, no. 4, pp. 21–38, 2012.

[15] H. W. Jensen, "Global illumination using photon maps," in *Rendering Techniques' 96*. Berlin, Germany: Springer, 1996, pp. 21–30.

[16] C. Alves, L. P. Santos, and T. Bashford-Rogers, "A quantum algorithm for ray casting using an orthographic camera," in *Proc. Int. Conf. Graph. Interaction*, 2019, pp. 56–63.

[17] E. R. Johnston, "Quantum supersampling," in *Proc. ACM SIGGRAPH 2016 Talks*, 2016, pp. 38:1–38:1, doi: 10.1145/2897839.2927422.

[18] E. R. Johnston, N. Harrigan, and M. Gimeno-Segovia, *Programming Quantum Computers*. Springfield, MO, USA: O'Reilly, 2019.

[19] N. H. Shimada and T. Hachisuka, "Quantum coin method for numerical integration," *Comput. Graph. Forum*, vol. 39, no. 6, pp. 243–257, 2020.

[20] D. S. Abrams and C. P. Williams, "Fast quantum algorithms for numerical integrals and stochastic processes," 1999, *arXiv:quant-ph/9908083*.

[21] X. Lu and H. Lin, "Improved quantum supersampling for quantum ray tracing," 2022, *arXiv:2208.05171*. [Online]. Available: https://arxiv.org/abs/2208.05171

[22] X. Lu and H. Lin, "Quantum ray tracing with simulation," *SPIN*, vol. 12, no. 4, 2022, Art. no. 2250030, doi: 10.1142/S2010324722500308.

[23] R. O'Donnell, "Lecture 1: Introduction to the quantum circuit model," 2015. [Online]. Available: https://www.cs.cmu.edu/odonnell/

[24] R. H. Katz, *Contemporary Logic Design*. San Francisco, USA: Benjamin-Cummings, 1994, p. 710.

[25] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, no. 79, pp. 79–98, 2018.

[26] A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta, "Validating quantum computers using randomized model circuits," *Phys. Rev. A*, vol. 100, no. 3, 2019, Art. no. 032328.

[27] A. Luongo, "Quantum algorithms for data analysis," 2022. [Online]. Available: http://quantumalgorithms.org

[28] V. Giovannetti, S. Lloyd, and L. Maccone, "Quantum random access memory," *Phys. Rev. Lett.*, vol. 100, no. 16, 2008, Art. no. 160501.

[29] S. M. Girvin, "Introduction to quantum error correction and fault tolerance," *SciPost Phys. Lect. Notes*, 2023, Art. no. 70. [Online]. Available: https://scipost.org/10.21468/SciPostPhysLectNotes.70

Luís Paulo Santos is an assistant professor with the University of Minho, Portugal. He lectures courses on Computer Architecture, Parallel Processing, Computers Graphics – Physically based Rendering and Quantum Computing. His research interests include focused on rendering, specifically Monte Carlo and MCMC methods for light transport simulation, and quantum computing, specifically Monte Carlo simulation of quantum circuits and near-term quantum algorithms.

Thomas Bashford-Rogers received the BSc degree in computer science from the University of Bristol, and the PhD in computer graphics, in 2011. He is an associate professor with the University of Warwick. His research interests include focused on rendering and high dynamic range imaging, specifically Monte Carlo and MCMC methods for light transport simulation, material models, natural phenomena such as sky illumination, quantum computing, and inverse tone mapping.

João Barbosa in March 2020 joined the Minho Advanced Computing Center (MACC) as a full-time Researcher in High-performance Computing, Scientific Visualization (SciVis). Previously, he was with the Texas Advanced Computing Center (TACC) Scalable Visualization team. He worked on several SciVis projects in partnership with leading hardware and software companies. His current research focuses on high-performance real-time in-situ photo-realistic ray tracing for SciVis.

Paul Navrátil is the director of strategic technologies with the Texas Advanced Computing Center (TACC), a research scientist with the University of Texas at Austin, and faculty for the Plan II interdisciplinary honors program. He leads a team tasked with imagining the scientific workflows of 2030 and beyond, encompassing hardware, software, data, analysis and UI/UX needs, developing TACC strategy to address those needs, and implementing transformative projects to achieve that vision..