Quantum Ray Marching for Reformulating Light Transport Simulation

Logan Mosier
University of Waterloo
Waterloo, Ontario, Canada
lmmosier@uwaterloo.ca

Morgan McGuire
Roblox, University of Waterloo
Waterloo, Ontario, Canada
morgan@casual-effects.com

Toshiya Hachisuka
University of Waterloo
Waterloo, Ontario, Canada
toshiya.hachisuka@uwaterloo.ca

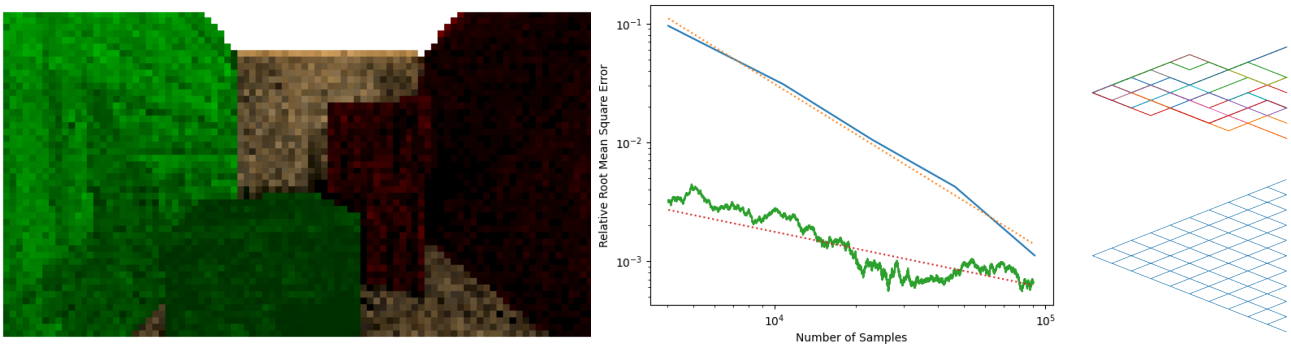

Figure 1: Left: Modified Cornell box rendered with our method using 32 paths per pixel (structural noise is due to limitations of simulation of quantum computing). Center: Error convergence plot. Our quantum light transport simulation with quantum ray marching (blue) converges asymptotically faster than classical Monte Carlo (MC) rendering (green). Right: Diagram of how light transport paths are sampled between when each bounce branches into two. Classical MC (top) will visit one random light transport path at a time and needs several samples (shown in different colors) to cover all the possible light transport paths faithfully. Thanks to the exponential nature of quantum computing, the quantum state in our quantum method captures all the exponential numbers of light transport paths in one quantum estimate (bottom).

ABSTRACT

The use of quantum computers in computer graphics has gained interest in recent years, especially for the application to rendering. The current state of the art in quantum rendering relies on Grover's search for finding ray intersections in $O(\sqrt{M})$ for M primitives. This quantum approach is faster than the naive approach of $O(M)$ but slower than $O(\log M)$ of modern ray tracing with an acceleration data structure. Furthermore, this quantum ray tracing method is fundamentally limited to casting one ray at a time, leaving quantum rendering scales for the number of rays the same as non-quantum algorithms. We present a new quantum rendering method, quantum ray marching, based on the reformulation of ray marching as a quantum random walk. Our work is the first complete quantum rendering pipeline capable of light transport simulation and remains asymptotically faster than non-quantum counterparts. Our quantum ray marching can trace an exponential number of paths with polynomial cost, and it leverages quantum numerical integration to converge in $O(1/N)$ for N estimates as opposed to non-quantum $O(1/\sqrt{N})$. These properties led to first quantum rendering that is asymptotically faster than non-quantum Monte Carlo rendering. We numerically tested our algorithm by rendering 2D and 3D scenes.

CCS CONCEPTS

• **Hardware** → **Quantum computation**; • **Computing methodologies** → **Ray tracing**; **Rendering**.

ACM Reference Format:
Logan Mosier, Morgan McGuire, and Toshiya Hachisuka. 2023. Quantum Ray Marching for Reformulating Light Transport Simulation. In *SIGGRAPH Asia 2023 Conference Papers (SA Conference Papers '23), December 12–15, 2023, Sydney, NSW, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3610548.3618151

1 INTRODUCTION

Recent advances in quantum computing have opened up opportunities to fundamentally accelerate certain numerical computation methods. One example is quantum numerical integration which was introduced to computer graphics by Johnston [2016]. Quantum numerical integration can be seen a fundamental improvement over Monte Carlo (MC) integration due to its faster $O(1/N)$ convergence than the classical $O(1/\sqrt{N})$ convergence for N estimates. It also has another unique property that its error is *independent* from the variance of the integrand unlike MC integration [Shimada and

SA Conference Papers '23, December 12–15, 2023, Sydney, NSW, Australia

ACM ISBN 979-8-4007-0315-7/23/12…$15.00
https://doi.org/10.1145/3610548.3618151

Hachisuka 2020]. Due to the popularity of MC integration in light transport simulation, it would be interesting to study how to apply quantum numerical integration to light transport simulation.

To fully utilize quantum numerical integration in light transport simulation, one would need to implement ray tracing on quantum computers because ray tracing is a key component in MC light transport simulation. Currently, ray tracing on quantum computers (which we call *quantum ray tracing* in this paper) is said to be realizable based on Grover's search algorithm [Alves et al. 2019; Lanzagorta and Uhlmann 2005; Santos et al. 2022]. Recently, Lu and Lin [2022a] speculated that a potential strength of light transport simulation on quantum computers (which we call *quantum light transport simulation* in this paper) is that it can branch a light transport path *exponentially* with a *polynomial cost* by utilizing the exponential nature of qubits (quantum bits). It is in contrast to classical MC light transport simulation where it *stochastically* selects a single path to avoid this exponential branching [Kajiya 1986]. Quantum numerical integration uses this property where the integrand is evaluated against *all the possible inputs* (e.g., an exponential number of branching paths) with polynomial cost to prepare a quantum state for numerical integration.

While all those properties sound attractive, we identified two fundamental issues in this current formalism of quantum light transport simulation via quantum ray tracing. The first issue is that each quantum ray tracing operation based on Grover's search costs $O(\sqrt{M})$ for M primitives [Lanzagorta and Uhlmann 2005]. While this is faster than a naive classical approach of $O(M)$, it is asymptotically slower than the $O(\log M)$ of ray tracing with a tree data structure on classical (i.e., non-quantum) computers [Wald and Havran 2006]. With this asymptotic performance gap, even when quantum computers become stable and fast in the future, quantum ray tracing is not very attractive over classical ray tracing.

The second issue is that the speculation that an exponential number of light transport paths can be computed at a polynomial cost is in fact *false* for ray tracing with Grover's search. While qubits might be able to store an exponential number of search results in a polynomial *storage cost*, finding an exponential number of search results with the Grover's algorithm will take an exponential *computation cost* because each search result needs to be read out into a classical bit first. Quantum light transport simulation with Grover's search thus scales the same way as the classical counterpart for the number of sampled paths, that is, exponential cost for an exponential number of paths. If we were to use MC sampling to avoid this exponential branching (like path tracing), the process becomes incompatible with quantum numerical integration.

We propose a new formulation of quantum light transport simulation with a new *quantum ray marching* algorithm to address these issues. Unlike quantum ray tracing with Grover's search, we employ ray marching as its core method where the scene geometry is represented by a voxel data as a promising alternative to ray tracing [Xie et al. 2022]. On the first issue of scaling against classical algorithms, our quantum ray marching scales equivalently to classical ray marching for the number of voxels and the number of steps. While this property alone is still neither good nor bad, quantum ray marching simultaneously solves the second issue and is capable of computing an exponential number of light transport paths in a polynomial time. We achieve this property by utilizing quantum random walk [Aharonov et al. 1993] to prepare a superposition of an exponential number of paths as qubits in a polynomial time [Joshi et al. 2018]. Being a fully quantum approach, our approach can benefit from the faster convergence $O(1/N)$ of quantum numerical integration. Our work is the first to provide a full picture of quantum light transport simulation without reading out qubits until the very end, with its fundamental advantages maintained over classical approaches. We also tested these properties numerically as a proof of concept. To summarize, our contributions are

- Formulation of ray marching on quantum computers.
- Quantum light transport simulation algorithm based on quantum ray marching with exponential branching.
- First full pipeline of quantum light transport simulation that is asymptotically faster than MC light transport simulation.

Fig. 1 shows a result of our method via a simulator of quantum computation. The reference solution was computer classically via MC integration using a larger number of samples. Due to the limitations of current (both actual and simulated) quantum computers we have access to, it is expected that the visual quality is not as good as what we typically see with MC rendering. This result mainly serves as a proof of concept that our method is implementable and has a faster convergence as expected. The amount of error is also worse in our method, which is expected for simple scenes with smooth illumination where MC integration is expected to work well. Nevertheless, unlike simulated results in prior work [Lu and Lin 2022b; Santos et al. 2022] that uses classical ray tracing in its numerical results, our results are from a fully quantum light transport simulation algorithm and are some of the most complicated scenes ever numerically tested on fully quantum rendering algorithms [Alves et al. 2019].

2 BACKGROUND

Since quantum computing is relatively new to typical readers in computer graphics, we first summarize the current approaches of quantum numerical integration and quantum ray tracing. Readers familiar with those topics may skip this section. Interested readers can refer to the book by Johnston et al. [2019] for a general review.

2.1 Basics of Quantum Computing

Qubits. A qubit is the quantum equivalent of a bit in classical computing. Unlike a bit, which can only be a 0 or 1, a qubit is represented by a complex vector, known as a state vector. Each entry in this vector represents the amplitude of the corresponding state. The equivalent of 0 and 1 states for a qubit are

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{1}$$

for $|0\rangle$ and $|1\rangle$. A ket, $|\rangle$, represents a column vector and a bra, $\langle|$, is a row vector (as opposed to complex conjugate). We will stick with this notation for the sake of brevity while admitting a lack of mathematical rigor, which is sufficient to explain our method.

We can represent a mixture of states known as a *superposition* in qubits. A superposition of the $|0\rangle$ and $|1\rangle$ states can be written as

$$|\psi\rangle = \sqrt{1-\alpha^2}\,|0\rangle + \alpha\,|1\rangle \quad (\alpha \in [0,1]) \tag{2}$$

where α is the amplitude of the $|1\rangle$ state. The amplitude α is a complex number in general, but for our purposes, α can be simply thought of as the square root of the probability that a given state will be measured. As such, the norm of a state vector must be 1. Measuring a qubit is a destructive operation and collapses the superposition into a single state.

A single qubit is not sufficient for doing much useful work. When working with multiple qubits, the state can be represented as a tensor product of qubit states as $|0\rangle \otimes |1\rangle$. We often drop the operator $\otimes$ and write as $|0\rangle\,|1\rangle$ for brevity when it is not misleading. The state vector of a system with n qubits can be described by 2^n amplitudes.

We will utilize two approaches for representing values using qubits [Weigold et al. 2022]. The first, known as basis encoding, stores data as binary data similar to non-quantum computers. The other, amplitude encoding, uses the amplitude of the different states to store data. Since the data is stored in the amplitudes, the data must be normalized first (i.e., the l^2 norm is one).

Gates. There are several models of quantum computing and the most prevalent one for our work is the quantum circuit model [Deutsch 1989]. In this model, the state of the system is evolved by the application of quantum gates, similarly to how logic gates are applied to bits in non-quantum computers. These gates can be thought of similarly to classical logic gates but operate on qubits.

Quantum gates are often defined as matrices. Two gates that will be relevant to our work are the X and R_y gates. The X gate is the quantum version of a bit flip, changing $|0\rangle$ to $|1\rangle$ and vice versa. The R_y gate, on the other hand, does not have a classical counterpart. This gate applies a rotation around the Y-axis to a qubit. Gates are not limited to acting on a single qubit and can be applied to multiple qubits at once. Controlled gates [Barenco et al. 1995] use such operations. These gates can be thought of as a version of the non-controlled gate but only act on the target qubit when the control qubit is in a desired state.

There are physical restrictions on the types of gates that can be constructed in practice. In particular, all gates must be physically realizable and must be consistent with the laws of quantum mechanics, meaning that all gates take the form of unitary matrices [Barenco et al. 1995]. While it may seem like a severe restriction of quantum computers, it can be shown that the Toffoli gate, a X gate with two control qubits, is universal [Aharonov 2003] (i.e., anything we can do with non-quantum computers can be done on quantum computers). We assume that all the gates are available.

2.2 Quantum Ray Tracing with Grover's Search

Grover's search [Grover 1996] was by Lanzagorta and Uhlmann [2005] to computer graphics. Their quantum ray tracing approach formulates the computation of the intersection between a ray and M primitives as a *search problem* of finding a primitive that intersects with a ray. Recent work by Alves et al. [2019] showed a practical implementation of this approach for simple scenes with orthographic rays. In the absence of an acceleration data structure, non-quantum computers would require $O(M)$ to perform such a search, whereas Grover's search can accomplish this task in $O(\sqrt{M})$. The latest approach along this line by Santos et al. [2022] which supports recursive ray tracing [Whitted 1979].

Although several improvements have been proposed over the basic approach, even the latest approach is still $O(\sqrt{M})$, which is slower than the $O(\log M)$ of ray tracing with a tree data structure on non-quantum computers [Wald and Havran 2006]. Quantum ray tracing with Grover's search does not accelerate ray tracing of *multiple* rays either beyond what we can do with parallel computation. When tracing N rays, $O(N)$ measurements are required to find intersecting primitives, which is the same scaling as the non-quantum approach. Santos et al. [2022] thus claim that quantum ray tracing would be mainly useful for dynamic scenes where construction of an acceleration data structure would need at least $O(M)$ for each frame and thus $O(\sqrt{M})$ of Grover's search is better. We instead propose an alternative approach to perform ray tracing operations based on *ray marching* to step aside these limitations due to Grover's search.

2.3 Quantum Numerical Integration

Both MC integration and quantum numerical integration aim to estimate a definite integral in the form of

$$F = \int_{\mathcal{D}} f(x)dx, \tag{3}$$

where $\mathcal{D}$ is the domain of integration and $f(x)$ is a scalar function that does not have an analytical expression for its integral F. MC integration uses random samples to approximate F as the expected value over those random samples. Without loss of generality, we assume that $F \in [0, 1]$ and $f(x)$ is appropriately scaled if it is not.

In quantum numerical integration, one would need to have a quantum circuit that computes $f(x)$ for input qubits, x, representing *all* the possible values of x given the representation of numbers (e.g., 32-bit floating point numbers). Given such a circuit, it is easy to construct another qubit

$$|F\rangle = \sqrt{1 - F^2}\,|0\rangle + F\,|1\rangle\,, \tag{4}$$

where it encodes the *correct* integral F up to the limit of the precision of x, with the computation cost *equivalent* to evaluating $f(x)$ for a single sample in MC integration on a classical computer. Quantum numerical integration uses multiple instances of $|F\rangle$ to estimate the amplitude F since it is impossible to read F directly from $|F\rangle$. This process is called amplitude estimation in quantum computing and was introduced to computer graphics by Johnston [2016] followed by more recent developments [Shimada and Hachisuka 2020].

Quantum numerical integration has two unique properties. Firstly, it internally encodes F and thus its estimation of F is *independent* of the variance of $f(x)$. Its accuracy depends only on amplitude estimation of F as demonstrated by prior work [Johnston 2016; Shimada and Hachisuka 2020]. MC integration requires more samples to reach the same error when the integrand $f(x)$ has a larger variance, and variance reduction techniques need to be employed. Such variance reduction is likely unnecessary or will be in a different form in quantum numerical integration.

Secondly, quantum numerical integration achieves $O(1/N)$ convergence for N evaluations of F, in contrast to the $O(1/\sqrt{N})$ convergence of MC integration for N evaluations of $f(x)$. Note that the computational complexity for *one* evaluation is considered the same despite the differences between F and $f(x)$. Even with quasi MC integration, the best possible convergence is $O((\log N)^d/N)$

for d dimensions [Lemieux 2009]. Quantum numerical integration is thus asymptotically faster than non-quantum approaches.

We utilize quantum numerical integration based on the work by Nakaji [2020] and Shimada and Hachisuka [2020], although other methods can be employed. Our main focus is to evaluate $f(x)$ on quantum computers for the case where $f(x)$ represents the measurement contribution function in the path integral formulation for a path x. We then use quantum evaluation of $f(x)$ to prepare $|F\rangle$ as a superposition of all the possible paths in x in the evaluation of $f(x)$ to support quantum numerical integration.

3 OVERVIEW

We use *ray marching* as a basic building block in quantum light transport simulation. We take a ray and its first intersection point as input and outputs the resulting radiance that is emitted from the intersection point in the inverse direction of the ray. Fig. 2 illustrates a circuit of our quantum ray marching.

Unlike the classical counterpart or even quantum ray tracing with Grover's search, our quantum light transport can process an exponential number of paths in polynomial storage *and* computation cost because the states of ray marching are all superpositioned as qubits. Our quantum ray marching scales equivalently to non-quantum ray marching for M voxels, which makes it a more feasible than quantum ray tracing with Grover's search. The performance gain in our method is not directly coming from ray marching itself since it scales equivalently to the classical one when all the paths are processed in parallel. It is due to its compatibility to quantum numerical integration.

Our quantum ray marching works as a quantum evaluation of the integrand $f(x)$ which allows us to take a superposition of all the possible paths x and construct the qubit $|F\rangle$. As a result, our method can handle an exponential number of rays in a polynomial cost (both storage and time) and simultaneously achieves faster $O(1/N)$ convergence for N estimates, both of which fundamentally outperform algorithms on non-quantum computers. The above properties cannot be achieved by any of the combinations of classical and quantum approaches at the moment. For example, quantum ray tracing with Grover's search scales *worse* than non-quantum ray tracing, and it scales linearly to the number of rays. One might consider running ray tracing classically, loading the results into qubits, and performing quantum numerical integration to combine both. This hybrid of non-quantum ray tracing and quantum numerical integration achieves better scaling for the number of primitives, though the construction of the qubit $|F\rangle$ for quantum numerical integration takes a prohibitive amount of computation (i.e., need to generate all the possible paths x as an input to a quantum circuit to construct $|F\rangle$). Our approach is the first to provide a full pipeline of quantum light transport simulation that would be fundamentally faster than non-quantum counterparts.

4 QUANTUM RAY MARCHING

We consider ray marching against a voxel grid of 3D surfaces, not 3D volumetric and solid objects. Each voxel is thus either empty or contains a piece of 3D surfaces. Our main idea is to formulate ray marching as a (*quantum*) random walk which fits well to the model of quantum computing and would not involve a search problem.

4.1 Ray Marching as a Discrete Random Walk

Given a camera configuration and a pixel, we assume that we have already found the first hit point of a ray through the pixel. This initialization step can be done on a non-quantum or quantum computer using the existing methods. Let us denote the position of this first intersection point as x_0 and the direction of the ray as ω_0 and we start a (discrete) random walk process.

We define the state of our random walk as a position x_i in the 3D space and it moves along the ray direction ω_i at each step. A scattering event will change the direction to a new direction ω_s if it hits a surface. Following this model, each step i can be written as

$$\omega_{i+1} = \begin{cases} \omega_s & (p_i \text{ is on surface}) \\ \omega_i & (\text{otherwise}) \end{cases} \tag{5}$$

$$p_{i+1} = p_i + \Delta V \omega_{i+1}, \tag{6}$$

where ΔV is a distance along ω_i to the next voxel.

To implement this process on a quantum computer, we can use the equivalent of a random walk for quantum computing, a *quantum walk* [Aharonov et al. 1993]. Both quantum and random walks transition from their current states to the new one step by step. The key difference is that, instead of deciding on a single state to visit at each step, a quantum walk creates a superposition of states at each step as if we were to step in *every possible direction* at each x_i. For instance, if we have n possible (discretized) scattering directions, a quantum walk can take all the directions into account by constructing a superposition of all the n directions and the next states can be similarly defined as a superposition of all the n possible next positions. Each step of a quantum walk thus becomes

$$|p_i\rangle |\omega_{i+1}\rangle = C |p_i\rangle |\omega_i\rangle \tag{7}$$

$$|p_{i+1}\rangle |\omega_{i+1}\rangle = S |p_i\rangle |\omega_{i+1}\rangle . \tag{8}$$

The unitary matrix C changes $|\omega_i\rangle$ based on the position of the walker. Similarly, S is another unitary matrix that moves the walker by $\Delta V \omega_i$. We evolve the quantum walk until a desired number of steps are taken, and then measure the resulting quantum state. Both p and ω are discretized and thus it forms a discrete random walk. Note that discretization of p and ω happen also in classical computers (e.g., 32-bit floating point numbers), so it is not specific restriction of our approach. If we measure this qubit, it collapses into a single state, giving us a random path among all the possible paths visited by the walker, proportional to the probability that the walker was in that state after our chosen number of steps.

4.2 Quantum Random Walk

We now explain the details of the algorithm. Our choices regarding encoding of values for positions and directions are explained later. We construct a quantum walk using two (quantum) registers (encoding the position of the walker and its direction), a coin gate, and a shift gate. Let us illustrate the function of these gates with a 1D example. The walk starts by initializing the position register with x_0. Next, the coin operator, C is applied to this register and $|0\rangle$ as

$$(I \otimes C) |x_0\rangle |0\rangle = |x_0\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle). \tag{9}$$

The name *coin* operator is derived from the fact that it replaces the direction deciding portion of the classical walk, which could

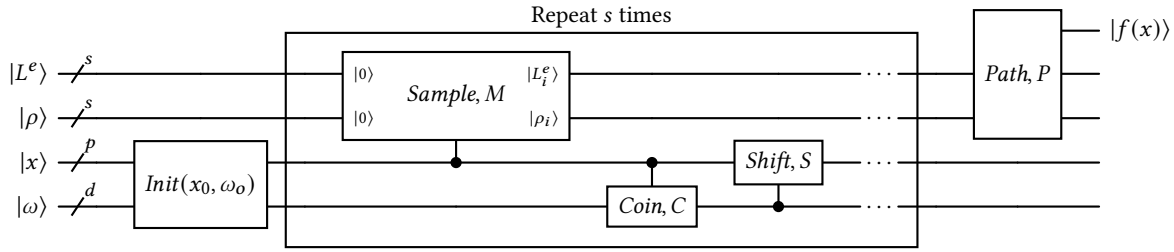

Figure 2: Overview of our ray marching circuit. The *Init* circuit initializes the position and direction of the walk. Next, the *Sample* gate encodes the emitted radiance and reflectance of the surface at the given position into a qubit from the $|L^e\rangle$ and $|\rho\rangle$ registers. After, the *Coin* gate scatters the ray in superposition over the outgoing directions. Finally, the *Shift* gate steps one unit along the ray. This process is repeated for each step of the walk. After all the samples are taken, *Path*, expanded in Fig. 3, evaluates the samples stored in $|L^e\rangle$ and $|\rho\rangle$ to evaluate $f(x)$ for all the paths in superposition.

theoretically be done by flipping a coin in this example. Instead of choosing a direction randomly, it creates a superposition of states that represent all the directions the walker can go in. In this example, $|0\rangle$ means moving the walker to the left and $|1\rangle$ means moving the walker to the right. The shift operator S is then applied to move the walker following the rules we have defined for the coin operator

$$S(|p_i\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)) = \frac{1}{\sqrt{2}}(|p_{i-1}\rangle |0\rangle + |p_{i+1}\rangle |1\rangle), \tag{10}$$

where we used p_i to denote the ith discretized location and we considered $p_i = x_0$. After repeating this process, we have a superposition $|p\rangle$ of all the possible positions by the walker.

We are interested in sampling the paths along the walk and not the final state, thus we apply another M gate

$$M |p\rangle |0\rangle |0\rangle = |p\rangle |L^e\rangle |\rho\rangle \tag{11}$$

to evaluate the emission L^e and reflectance ρ into two qubits via amplitude encoding the value in the 1 state of the qubit, given p. The distributions of directions are selected such that the product of the BRDF and the geometry terms cancels out with the PDF to leave only the reflectance. It is important to note that this will be applied to a superposition of positions in the scene and as such, will sample all the positions the walker is currently located at.

Coin Gates. Coin gates represent ray-surface (non)interactions. We explain three different coins, one models the vacuum the other two coins model different surfaces. We follow the work of Ahmad et al. [2020] and use position dependent coins. This approach uses different coins controlled by the position of the ray. The use of position dependent coins allows the definition of scattering behaviors for different surfaces.

The vacuum coin models null interaction and is thus simply the identity gate. One surface coin is the specular surface coin. This coin maps the incident direction to the appropriate matching reflected direction over the normal of the surface. This mapping can be precalculated and a circuit simply carries out this mapping. While implemented, we did not use this specular coin in our experiments.

Another surface coin is the Lambertian surface coin. Since surfaces now scatter light uniformly, the resulting direction qubit state should be a superposition of outgoing directions. A challenge here is that that different incident directions map to the identical set of scattered directions. Since quantum gates must be reversible, it may seem like this coin gate is not realizable. One potential solution would be to add another register to identify the direction at each step, but it would make circuits wider as we would need to introduce additional d qubits (one direction) for each step. Our alternative solution is to utilize of the fact that the amplitude of a state may be negative. We can then map different input ray directions to different quantum states by shuffling the signs of the amplitudes while keeping amplitudes (directions). By mapping each direction to a unique quantum state in this fashion, we can construct a unitary matrix that enables the desired scattering behavior while being reversible.

While theoretically possible, a more complicated circuit and more qubits would be needed for other scattering events (e.g., glossy surfaces), so we limit ourselves to simple materials in this paper.

4.3 Quantum Evaluation of Path Contribution

Just being able to sample a path x is not sufficient for a full pipeline of quantum light transport simulation. We need to evaluate the contribution function $f(x)$ in a quantum circuit where x is potentially a superposition of many paths represented as qubits. To proceed, note that the result so far is a quantum state encoding the samples along the light paths that have been traced;

$$|L^e\rangle |\rho\rangle = \bigotimes_{i=0}^{s} |L_i^e\rangle |\rho_i\rangle \tag{12}$$

where $s+1$ is the number of steps taken and $|L_i^e\rangle$ and $|\rho_i\rangle$ are the qubits storing emission and reflectance at the ith step of walk. Before we can apply any form of quantum amplitude estimation to this state, we need to calculate $f(x)$ using the values encoded in these qubits. To achieve this, we want to construct a circuit P:

$$P |0\rangle \bigotimes_{i=0}^{s} |L_i^e\rangle |\rho_i\rangle = (\sqrt{1 - f(x)^2} |0\rangle + f(x) |1\rangle) \bigotimes_{i=0}^{s} |L_i^e\rangle |\rho_i\rangle \tag{13}$$

where

$$f(x) = \sum_{i=0}^{s} L_i^e \prod_{k=0}^{i-1} \rho_k. \tag{14}$$

Evaluating $f(x)$ requires the ability to multiply and add values encoded in the amplitudes L_i^e and ρ_i. While arbitrary arithmetic

operations on amplitudes are challenging in general, we were able to take inspiration from the work of Wang et al. [2020] to evaluate $f(x)$ in amplitudes. Fig. 3 illustrates this process. To do so, we first add qubits $|Steps\rangle$ to basis-encode i. These additional qubits are used to create a superposition of all the relevant terms to control which terms should be multiplied and added to the output qubit (Fig. 3). We then perform multiplications using multi-controlled X gates where each gate will be controlled by the step number i and all the samples involved with the summation up to a specific step. The resulting amplitude $f'(x)$ becomes

$$f'(x) = \frac{1}{s+1} \sum_{i=0}^{s} L_i^e \prod_{k=0}^{i-1} \rho_k = \frac{1}{s+1} f(x) \tag{15}$$

where the division by $s+1$ is a side-effect of this calculation. We thus multiply $s+1$ to the result of amplitude estimation to recover F. Note that no random sampling of $f(x)$ is done here as it is evaluated at all the possible x as a superposition in qubits.

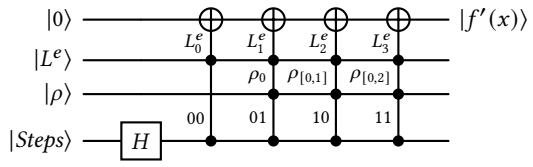

Figure 3: Internals of the *Path* gate for a three step walk. Each multi-controlled X gate calculates one of the terms of the sum in Eqn. 15. The ancillary register, $|Steps\rangle$ is used to select the term of the sum.

4.4 Implementation

We explain the reasoning behind our implementation choices below. We release our implementation to facilitate the reproducibility.

Encoding of Numbers. We encode the origin and direction via basis encoding which is basically fixed-point binary encoding of values as states. Basis encoding allows us to trivially perform the same class of computations on binary numbers in classical computers. For origins, we use an integer value to represent a voxel index, and a fixed-point fractional value to represent a sub-voxel location within the voxel. For directions, rather than representing a direction vector as three values, we tabulate a predefined set of directions as a 1D array of 3D vectors and index it via one integer value. This representation is still basis encoding since one state corresponds to one direction, but dramatically simplifies the circuit.

Another option, amplitude encoding, represents a value as a magnitude of a specific state, allowing us to represent a real number with an amplitude value without any discretization. We use this encoding for L^e and ρ to evaluate $f(x)$ (thus $f'(x)$).

Scene Representation. We choose to represent a scene as a voxel grid for simple lookup and indexing of the needed scene information. At each voxel, we store material properties: emission, reflectance, (averaged) surface normal, and type of that voxel. We make two assumptions to simplify the circuit; the emitted radiance is uniform in all directions and all surfaces are Lambertian. The voxel type defines the type of the coin gate as described earlier.

Currently, the scene description will need to be built as a quantum circuit which calculates the voxel data on the fly (conceptually the same as procedural modeling) since quantum computers have no model of memory or storage at this moment. It is, however, not a fundamental requirement of our quantum ray marching and we expect that a quantum storage [Giovannetti et al. 2008] will allow us to store a scene description, just like how a scene is stored in non-quantum computers.

One potential benefit of a quantum storage of voxels is that the number of qubits needed to store voxels will be *logarithmic* to the number of voxels. For instance, suppose that we have a binary voxel where we have 1 for a (partially) occupied voxel and 0 otherwise. If we have 2^m such voxels, we can encode it in a superposition using m qubits only, where the state corresponds to an index of each voxel. It is in contrast to non-quantum computers where we would need to store 2^m bits, thus voxels can scale much better on quantum computers than on classical computers.

Amplitude Estimation. The result of our quantum light transport circuit is a qubit that stores F' (integral of $f'(x)$) at the desired point in the scene as the amplitude of the 1 state. To estimate F', we use the approach from Nakaji [2020] and Shimada and Hachisuka [2020]. Unlike the approach taken by Johnston [2016], it does not use phase estimation. As identified by Shimada and Hachisuka [2020], the use of amplitude amplification instead of phase estimation can cut down on the number of qubits and multi qubit gates.

5 RESULTS

As a proof of concept, we conducted numerical experiments for our method. The aim is to numerically confirm that our method is capable of simulating quantum light transport. We tested our methods both in 2D and 3D. We also tested if our method can achieve the claimed convergence rate of quantum numerical integration by comparing our result with the reference solution computed classically for the same input scene data with same discretization. The scene complexity is limited by available computing environments of quantum computing at the moment. Nevertheless, our scenes are much more complicated than scenes rendered by another fully quantum rendering algorithm to the date [Alves et al. 2019].

Fully Quantum 2D Light Transport. We identified that quantum computers (real or simulated) are not yet capable to run our complete circuit for 3D scenes. We thus tested our circuit for simplified $2D$ ray marching. Only by stepping down into $2D$, we could run our circuit on a simulator of quantum circuits. The scene is modeled as a 8^2 grid. At each cell, we store the same information that we would store in a 3D scene. The only change is that the directions are now defined on a unit circle instead of a sphere. We use our method to create a light map by sampling the incoming light at the center of every cell in the scene. We then use the resulting light map to render a higher resolution image, using bilinear interpolation to sample from the light map. The computation took roughly *five days* for the full image. Fig. 4 shows the resulting images of our method. Quantum ray marching is capable of capturing both direct and indirect lighting in the scene; for example, the light being reflected off the colored walls causing some color to spill onto the

floor. While the scene is extremely simple, it demonstrates that our method can simulate light transport fully by quantum computing.

Emulated 3D Light Transport. Simulating our quantum circuit is very computationally expensive even for the simple 2D scene and we found that rendering 3D scenes through simulated quantum circuits is infeasible. We thus took a different approach to evaluate our method in 3D. Instead of feeding our quantum circuit to a generic quantum circuit simulator, we developed a classical ray-marcher that samples the scene under the same discretization and scattering behavior as our quantum algorithm (i.e., branching exponentially). Our emulator yields the same samples that would be obtained by the quantum walk but it simply takes exponential time with the number of steps instead of a polynomial time. This emulator replaces the sampling part of our quantum circuit. We call this approach *emulated* to distinguish it from the fact that our 2D result are from simulation of quantum circuits. This emulation allows us to validate the properties of our method without compromising anything, except that the actual running time is now exponential. To further simplify the problem, we limited the number of paths. These paths are selected at random from the set of all the paths and are then used to initialize $|L^e\rangle$ and $|\rho\rangle$. We can apply the P gate (Fig. 3) to generate the final path contribution. We then use the amplitude estimation method [Nakaji 2020] to estimate the final value.

Figs. 1 and 5 show the resulting rendered images. The images contain structural noise due to discretization and a limited number of paths used to keep the computation cost feasible. Unlike previous quantum ray tracing methods [Lu and Lin 2022a; Santos et al. 2022], our approach can intrinsically handle paths with multiple bounces as well as other effects such as soft shadows.

Convergence Rates. To compare convergence rates, we need a common metric for computation cost across classical and quantum computers, which is difficult as how the methods function are inherently different. We decided to use the number of samples in MC integration or the number of instances of quantum amplitude estimation as the common metric. For the classical method, we will be using the number of paths traced, and, for our method, we will use the number of times our ray marching circuit was executed. We argue that this metric is fair as it is the way which both methods sample the scene by one estimation. We could use the number of light transport paths sampled, but we believe it is unfair for MC integration since our quantum method cannot read out F' directly while it samples an exponential number of light transport paths.

Fig. 1 shows that our method converges with the expected behavior. The classical MC method has the convergence with a slope of about -0.471. Our method has a better-than-expected (< -1) slope of about -1.407. While this convergence is better than the classical approach, the actual error of our method starts higher than classical MC, which is likely because the scene is dominated by low-variance light transport effects. The error plot however shows that our method converges as expected.

6 DISCUSSION

Though there has been limited work in the field of computer graphics, quantum numerical integration been an active area of research in other fields. One of these fields is finance, where the problem of pricing financial derivatives is classically done via MC integration. Those prior work [Li and Neufeld 2023; Rebentrost et al. 2018; Woerner and Egger 2019] have looked at the application of quantum mean estimation methods for this problem. For these methods, the challenge of state preparation for quantum numerical integration was side-stepped by assuming that the solution to the *integral* is readily computable. While this assumption is impractical since this solution is what we wanted to compute, it allows for such quantum methods to be tested and analyzed. Our work, in contrast, shows how to exactly compute the *integrand* for the path integral for light transport on quantum computers and we numerically tested our method without making this impractical assumption. The most similar work to our method is by Chakrabarti et al. [2023]. They proposed to use quantum random walks to estimate the volume of a given shape. Their method uses continuous quantum walks and simulated annealing in contrast to our approach that relies on discrete walks and mean estimation.

There are two metrics in which a quantum circuit can scale and both of them are important for the usability of an algorithm. The first is the width of the circuit, or how many qubits are needed. With current quantum computers being very limited in the number of qubits, the width is a strict restriction in being able to run on physical devices. It is also an issue for simulating the circuit on a classical computer as the size of the quantum state grows exponentially in the number of qubits. Our approach scales in $O(s + log(p) + log(d))$ in the number of qubits needed. That is, it scales logarithmically in the number of positions p along an axis and directions d that can be represented in the scene, but linear in the number of steps s needed.

The other metric is the depth of the circuit. Our circuit scales in depth by $O(s(p^3 + d))$. Scaling cubically in scene size is not ideal, but it is still viable. The cubic scaling regarding the dimension does happen also in non-quantum ray marching if each voxel's data is generated on the fly as in procedural modeling. In most cases, such procedural generation would not scale this badly, so our analysis is only the worst case. The main issue is that quantum computers do not have any form of memory or storage that can be referenced. This restriction means that each voxel must be represented in the circuit. As there are p^3 voxels, we get p^3 gates at each step. In practice, not all voxels have completely different materials, allowing us to simplify the circuit to use fewer gates by combining gates. This issue can also be avoided if quantum random memory became available [Giovannetti et al. 2008]. The material values could then be accessed in a similar fashion as on a classical computer.

Limitations. There are limitations came from our current implementation. Foremost is its reliance on large many-qubit gates. Current generation of quantum computers support a limited set of one and two qubit gates, so these larger many-qubit gates need to be decomposed into many one or two qubit gates. This decomposition significantly increases the depth of the circuit. The increase in the depth in turn increases the amount of time the system needs to stay coherent, and eventually leads to the chance of errors. The other limitation is the use of amplitude encoding to store sample contributions. We need to have a separate scaling to represent a light intensity greater than 1. It likely would be possible to use basis

encoding for the sample contributions as well, but doing so would lead to even more complicated circuits. We do not claim to attain quantum supremacy on ray marching since nothing is fundamentally different between our quantum ray marching and classical ray marching running on an exponential number of parallel processes. Quantum supremacy for ray marching is still an open question.

Future Work. Though we are only capable of modelling and rendering simple scenes currently, this is not a fundamental limitation of our method but is a limitation on the current capabilities of quantum computers and simulators. It is interesting to conduct experiments on more complex scenes once we have better quantum computing environments. We could also simulate a broader set of materials by introducing more coin gates, such as participating media with volumetric scattering. We are also interested in running our algorithm on actual quantum computers once they became capable of running our 3D approach.

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for their helpful comments and feedback. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) under Grant No. RGPIN-2020-03918.

REFERENCES

Dorit Aharonov. 2003. A Simple Proof that Toffoli and Hadamard are Quantum Universal. arXiv:quant-ph/0301040 [quant-ph]

Yakir Aharonov, Luiz Davidovich, and Nicim Zagury. 1993. Quantum Random Walks. *Phys. Rev. A* 48 (Aug 1993), 1687–1690. Issue 2. https://doi.org/10.1103/PhysRevA.48.1687

Rashid Ahmad, Uzma Sajjad, and Muhammad Sajid. 2020. One-dimensional Quantum Walks with a Position-Dependent Coin. *Communications in Theoretical Physics* 72, 6 (2020), 65101–.

Carolina Alves, Luís Paulo Santos, and Thomas Bashford-Rogers. 2019. A Quantum Algorithm for Ray Casting Using an Orthographic Camera. In *2019 International Conference on Graphics and Interaction (ICGI)* (2019-11). IEEE, Faro, Portugal, 56–63. https://doi.org/10.1109/ICGI47575.2019.8955061

Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. 1995. Elementary Gates for Quantum Computation. *Phys. Rev. A* 52 (Nov 1995), 3457–3467. Issue 5. https://doi.org/10.1103/PhysRevA.52.3457

Shouvanik Chakrabarti, Andrew M. Childs, Shih-Han Hung, Tongyang Li, Chunhao Wang, and Xiaodi Wu. 2023. Quantum Algorithm for Estimating Volumes of Convex Bodies. *ACM Transactions on Quantum Computing* 4, 3, Article 20 (may 2023), 60 pages. https://doi.org/10.1145/3588579

D. Deutsch. 1989. Quantum Computational Networks. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences* 425, 1868 (1989), 73–90. http://www.jstor.org/stable/2398494

Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. 2008. Quantum Random Access Memory. *Phys. Rev. Lett.* 100 (Apr 2008), 160501. Issue 16. https://doi.org/10.1103/PhysRevLett.100.160501

Lov K. Grover. 1996. A Fast Quantum Mechanical Algorithm for Database Search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (Philadelphia, Pennsylvania, USA) *(STOC '96)*. Association for Computing Machinery, New York, NY, USA, 212–219. https://doi.org/10.1145/237814.237866

Eric R. Johnston. 2016. Quantum Supersampling. In *ACM SIGGRAPH 2016 Talks* (2016-07-24) *(SIGGRAPH '16)*. Association for Computing Machinery, New York, NY, USA, 1. https://doi.org/10.1145/2897839.2927422

Eric R. Johnston, Nic Harrigan, and Mercedes Gimeno-Segovia. 2019. *Programming Quantum Computers: Essential Algorithms and Code Samples.* O'Reilly Media, Sebastopol, California, USA. https://www.oreilly.com/library/view/programming-quantum-computers/9781492039679/

Karthik S. Joshi, S. K. Srivatsa, and R. Srikanth. 2018. Path Integral Approach to One-dimensional Discrete-time Quantum Walk. arXiv:1803.00448 [quant-ph]

James T. Kajiya. 1986. The Rendering Equation. *Comput. Graph. (Proc. SIGGRAPH)* 20, 4 (1986), 143–150.

Marco Lanzagorta and Jeffrey K. Uhlmann. 2005. Hybrid Quantum-Classical Computing with Applications to Computer Graphics. In *ACM SIGGRAPH 2005 Courses* (2005-07-31) *(SIGGRAPH '05)*. Association for Computing Machinery, New York, NY, USA, 2–es. https://doi.org/10.1145/1198555.1198723

Christiane Lemieux. 2009. *Monte Carlo and Quasi-Monte Carlo Sampling.* Springer New York, New York, USA. https://books.google.ca/books?id=wj5OyydZ5bkC

Yongming Li and Ariel Neufeld. 2023. Quantum Monte Carlo algorithm for solving Black-Scholes PDEs for high-dimensional option pricing in finance and its proof of overcoming the curse of dimensionality. arXiv:2301.09241 [quant-ph]

Xi Lu and Hongwei Lin. 2022a. A Framework for Quantum Ray Tracing. arXiv:2203.15451 [quant-ph]

Xi Lu and Hongwei Lin. 2022b. Improved Quantum Supersampling for Quantum Ray Tracing. arXiv:2208.05171 [quant-ph]

Kouhei Nakaji. 2020. Faster amplitude estimation. *Quantum Information and Computation* 20, 13&14 (nov 2020), 1109–1123. https://doi.org/10.26421/qic20.13-14-2

Patrick Rebentrost, Brajesh Gupt, and Thomas R Bromley. 2018. Quantum computational finance: Monte Carlo pricing of financial derivatives. *Physical Review A* 98, 2 (2018), 022321.

Luís Paulo Santos, Thomas Bashford-Rogers, João Barbosa, and Paul Navrátil. 2022. Towards Quantum Ray Tracing. arXiv:2204.12797 [cs.GR]

Naoharu H. Shimada and Toshiya Hachisuka. 2020. Quantum Coin Method for Numerical Integration. *Computer graphics forum* 39, 6 (2020), 243–257.

Ingo Wald and Vlastimil Havran. 2006. On building fast kd-Trees for Ray Tracing, and on doing that in O(N log N). In *2006 IEEE Symposium on Interactive Ray Tracing*. IEEE, Salt Lake City, UT, USA, 61–69. https://doi.org/10.1109/RT.2006.280216

Shengbin Wang, Zhimin Wang, Guolong Cui, Lixin Fan, Shangshang Shi, Ruimin Shang, Wendong Li, Zhiqiang Wei, and Yongjian Gu. 2020. Quantum Amplitude Arithmetic. arXiv:2012.11056 [quant-ph]

Manuela Weigold, Johanna Barzen, Frank Leymann, and Marie Salm. 2022. Data Encoding Patterns for Quantum Computing. In *Proceedings of the 27th Conference on Pattern Languages of Programs* (Virtual Event) *(PLoP '20)*. The Hillside Group, USA, Article 2, 11 pages.

Turner Whitted. 1979. An Improved Illumination Model for Shaded Display. *SIGGRAPH Comput. Graph.* 13, 2 (aug 1979), 14. https://doi.org/10.1145/965103.807419

Stefan Woerner and Daniel J Egger. 2019. Quantum risk analysis. *npj Quantum Information* 5, 1 (2019), 15.

Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent sitzmann, and Srinath Sridhar. 2022. Neural Fields in Visual Computing and Beyond. *Computer Graphics Forum* 41, 2 (2022), 641–676. https://doi.org/10.1111/cgf.14505 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14505

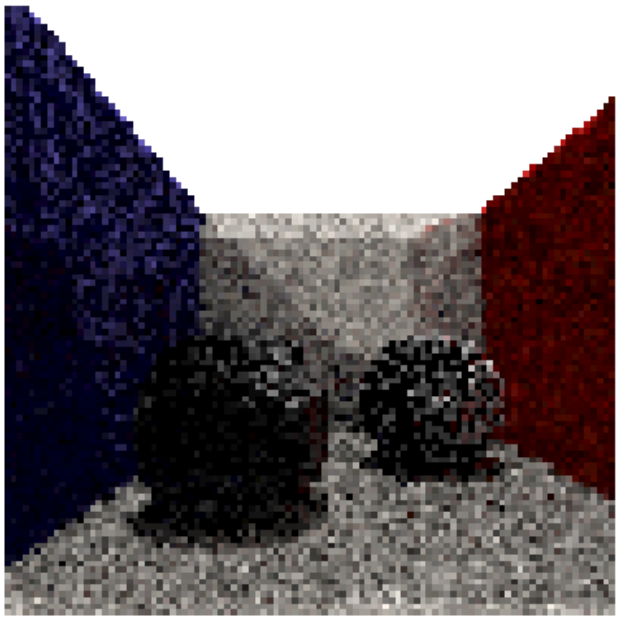

Figure 5: Different test scene of two spheres in a box with an overhead light rendered by the same algorithm. While the result is very noisy due to the current limitations of the quantum computing environment we used, the image shows how shadows and some interreflections can be captured via a fully quantum algorithm for the first time.

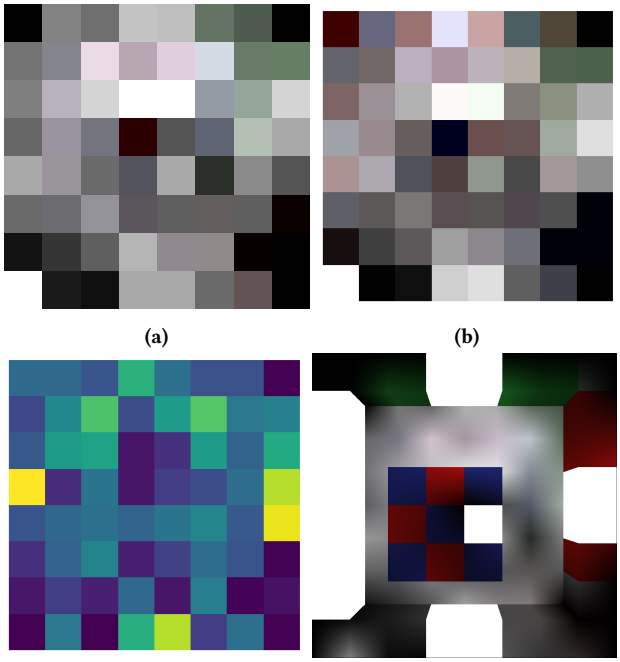

Figure 4: (a) The 8×8 light map of the $2D$ scene sampled at each point using a quantum walk of three steps. (b) The same light map generated by tracing exponential rays classically. (c) The mean square error of the classical and quantum light maps. The results match overall beside some remaining noise. (d) The final higher-resolution image using the quantum light map.