Towards Quantum Ray Tracing - Supplementary Material

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

This document contains supplementary material associated with the document "Towards Quantum Ray Tracing". We first describe quantum searching in more detail and provide the steps and intuition behind this algorithm. We then provide a detailed description of the 3D scenes used in the experiments reported in the paper. We conclude with more details about our implementation of practical quantum ray tracing.

I. Quantum Searching

Consider a function acting on a binary string $f : \{0,1\}^n \rightarrow \{0,1\}$, or equivalently in decimal notation $f : X \rightarrow \{0,1\}$, where X denotes the set of all non-negative integers less than $N = 2^n$, $\{0, 1, \ldots N-1\}$. Define f(x) as

$$f(x) = \begin{cases} 1 & \text{if } x \text{ is a solution} \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Searching can be described as the problem of finding a value $x \in X$ such that $f(x) = 1$.

This section introduces quantum searching over an unstructured database. The cases when the number of solutions to the problem is known and unknown are presented, followed by the problem of searching for a minimum.

A. Grover's algorithm

If nothing is known about the structure of $f()$ and if there are t solutions (i.e., t different x values $\in X$ such that $f(x) = 1$), then a classical approach will find a solution after $N/(t+1)$ evaluations of $f()$ on average and $N - t + 1$ such evaluations in the worst case.

Let $\widehat{\mathcal{A}}$, referred to as the oracle, be the quantum operator implementing $f()$, such that

$$\widehat{\mathcal{A}}|0\rangle^{\otimes n}|0\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle|f(i)\rangle \tag{2}$$

$\widehat{\mathcal{A}}$ sets the output qubit to $|1\rangle$ when the value $|i\rangle$ represented by the n state qubits is a solution for $f()$. A single evaluation of the oracle computes the value of $f()$ for all points in X, this is quantum parallelism; however, these cannot be measured. Grover's quantum searching algorithm enables finding, with high probability, a solution for the search problem with $\mathcal{O}(\sqrt{N/t})$ evaluations of $\widehat{\mathcal{A}}$ and its inverse, therefore providing a quadratic advantage over the classical result [1].

Let the vector $|\Psi\rangle = \widehat{\mathcal{A}}|0\rangle^{\otimes n}|0\rangle$ be expressed as a linear combination of the orthogonal vectors $|\Psi_0\rangle$ and $|\Psi_1\rangle$, see figure 1. $|\Psi_0\rangle$ contains all the basis states $|i\rangle$ that do not satisfy $f()$, and $|\Psi_1\rangle$ contains the remaining basis states, i.e, those satisfying $f(i) = 1$. Clearly, $|\Psi\rangle = \sqrt{\frac{N-t}{N}}|\Psi_0\rangle + \sqrt{\frac{t}{N}}|\Psi_1\rangle$, up to some normalizing constant. More formally this partitions the Hilbert space of the quantum system into two subspaces which are labelled as "good" and "bad" subspaces [2].

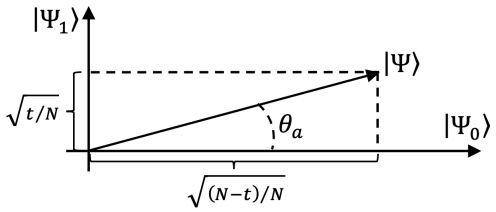


Fig. 1. $|\Psi\rangle$ represented as a linear combination of the subspaces defined by the non-solution and the solution basis states, i.e. $|\Psi_0\rangle$ and $|\Psi_1\rangle$, respectively.

The angle subtended by $|\Psi\rangle$ and $|\Psi_0\rangle$ is denoted by θ_a, with $\sin(\theta_a) = \sqrt{t/N}$ meaning that $|\Psi\rangle$ can also be represented as $|\Psi\rangle = \cos(\theta_a)|\Psi_0\rangle + \sin(\theta_a)|\Psi_1\rangle$. The probability of measuring a good state, i.e. a state in $|\Psi_1\rangle$, is $a = \sin^2\theta_a = \frac{t}{N}$. Grover's algorithm seeks to maximize this probability, thereby finding a solution to $f()$.

The oracle $\widehat{\mathcal{A}}$ implements $f()$ by marking states which are solutions to $f(x) = 1$. Grover's algorithm requires two additional operators.

$\widehat{\mathcal{S}_\chi}$ flips the sign of the amplitudes of marked solutions, i.e, those with the output qubit equal to $|1\rangle$ – see the bottom left image of figure 2.

$\widehat{\mathcal{S}_\mathcal{O}}$, on the other hand, flips the sign of only the zero state coefficient. The diffusion operator $\widehat{\mathcal{D}} = -\widehat{\mathcal{A}}\widehat{\mathcal{S}_\mathcal{O}}\widehat{\mathcal{A}}^{-1}$ performs a reflection over the mean of all basis states' amplitudes. In practice, this amplifies the amplitude of the marked states, whose amplitude was negative after applying $\widehat{\mathcal{S}_\chi}$ – see the top center image in figure 2, where the orange line indicates the average amplitude of all states and each state amplitude has been mirrored over this average. Equivalently, this corresponds to rotating state $|\Psi\rangle = \widehat{\mathcal{A}}|0\rangle^{\otimes n}|0\rangle$ by $2\theta_a$, thereby increasing the probability of measuring a "good" state from $\sin^2(\theta_a)$ to $\sin^2(3\theta_a)$ – see the top row of figure 2.

A Grover evaluation is defined as marking and amplifying the amplitude of "good" states, resulting in Grover's operator $\widehat{\mathcal{Q}} = \widehat{\mathcal{D}}\widehat{\mathcal{S}_\chi}$. Figure 2 shows this pictorially, where "bad" states are shown in red, and "good" states in green. Each successive evaluation of $\widehat{\mathcal{Q}}$ further rotates the state vector by $2\theta_a$. After r applications of $\widehat{\mathcal{Q}}$ the angle between $|\Psi\rangle$ and $|\Psi_0\rangle$ is given by $\theta_a^{(r)} = (2r+1)\theta_a$ and $|\Psi\rangle = \cos((2r+1)\theta_a)|\Psi_0\rangle + \sin((2r+1)\theta_a)|\Psi_1\rangle$. The probability of measuring a "good"

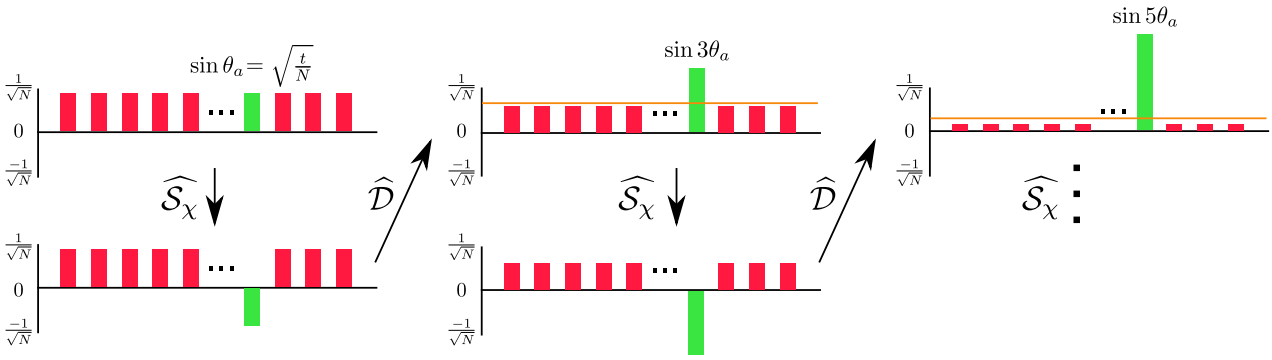


Fig. 2. Grovers algorithm for $t = 1$. Successive applications of $\widehat{\mathcal{Q}}$ progressively maximize the probability of reading a "good" state, marked in green. The operator $\widehat{\mathcal{S}_\chi}$ flips the sign of "good" states, and the operator $\widehat{\mathcal{D}}$ marks "good" states and flips those states around the average, shown as an orange line.

state is given by

$$p_{\Psi_1}(N, t, r) = \sin^2((2r + 1)\theta_a) \tag{3}$$

It can be shown, for sufficiently large N and $t \ll N$, that $p_{\Psi_1}(N, t, r)$ is maximized for

$$r = r_{opt} = \left\lfloor \frac{\pi}{4}\sqrt{\frac{N}{t}} \right\rfloor$$

This result gives quantum searching the complexity of $\mathcal{O}(\sqrt{\frac{N}{t}})$, a quadratic advantage over the classical complexity of $\mathcal{O}(\frac{N}{t+1})$. Since the sin is a periodic function, performing more than r_{opt} Grover evaluations will decrease p_{Ψ_1}.

If t is not significantly smaller than N, the expression for r_{opt} no longer applies. However, in such cases measuring the uniform superposition $|\Psi\rangle = \widehat{\mathcal{A}}|0\rangle^{\otimes n}|0\rangle$ will succeed with probability t/N, which might be computationally effective given that t/N will be significantly larger than 0. By repeatedly sampling the superposition k times, a good state will be found with probability equal to $\frac{t}{N}\sum_{i=1}^{k}\left(\frac{N-t}{N}\right)^{i-1}$. If $t > N/2$, i.e. over half the states are solutions, Grover's algorithm no longer amplifies the amplitude of good states, but a solution can be found by sampling $|\Psi\rangle$ at most twice.

A good solution is found with probability $p_{\Psi_1}(N, t, r_{opt})$, with a false positive being returned otherwise. Classically verifying whether the returned value x is a solution is $\mathcal{O}(1)$. Grover's algorithm can be reevaluated if a false positive is measured.

However, the successful application of Grover's algorithm requires that the number of solutions t is known beforehand.

B. Adaptive Exponential Search

When there is an unknown number of solutions t, the parameters derived from it, such as a, θ_a and r_{opt}, are also unknown. This is especially the case in many graphics problems; for example, a ray may intersect with an unknown number of primitives. Boyer et al. [3] proposed a hybrid quantum-classical algorithm to this type of problem, which still allows for a quadratic query complexity gain w.r.t. a classical approach.

This algorithm is referred to as Adaptive Exponential Search (or `QSearch()` – see algorithm 3) and is based on the classical exponential search algorithm [4]. It is an iterative algorithm, which executes Grover's algorithm multiple times with different numbers r of evaluations of $\widehat{\mathcal{Q}}$ – line 12 in algorithm 3. For each iteration l the respective r_l is randomly selected within an interval ranging from 1 to M_l – line 11. It is this upper limit, denoted by $M_l = \lceil c^l \rceil$, that grows exponentially at each iteration l of `QSearch()` up to a maximum of $\sqrt{N}$ – line 10. c is a constant such that $c \in)1, 2)$. After each execution of Grover's algorithm, with the respective r_l, the resulting state is measured – line 13. The returned value i is then classically evaluated, to check whether $f(i) = 1$ (line 14). If the test succeeds the algorithm terminates. This classical evaluation of $f()$ for a single point of its domain is $\mathcal{O}(1)$, therefore it does not add up to the algorithm's complexity.

Prior to the iterative process the uniform superposition is randomly sampled – line 2. The measured value is then classically checked; this is the evaluation of $f(i)$ in line 3. Randomly sampling will, with high probability, successfully handle the cases where the number of solutions t is large compared to N, particularly the case where $t \geq N/2$. The iterative exponential searching process is entered only when random sampling fails.

The algorithm converges in $\mathcal{O}(\sqrt{\frac{N}{t}})$ steps, therefore exhibiting the same complexity as Grover's algorithm.

`QSearch()` never returns a false positive, since solutions are always classically checked at either lines 3 or 14. If found=False then it is either a true or a false negative. A true negative corresponds to the no solutions case, i.e., $t = 0$. A false negative is understood as returning a value i in the search domain with $f(i) = 0$, when in fact $t > 0$, i.e., there are solutions $j \neq i$, such that $f(j) = 1$. Determining whether i is a false or a true negative cannot be done without classically visiting all points of the search domain (in the worst case). The probability of false negatives, $\overline{p}_{QS}$, can be arbitrarily reduced by repeating `QSearch()` multiple times. A novel estimate for $\overline{p}_{QS}$ is developed in the main paper.

C. Minimum finding algorithm

The minimum finding algorithm [5], [6] is an iterative approach that searches for a value $i \in X$ which is a solution for $f()$ (equation 1) and also minimizes a function $g : X \to \mathbb{Z}$. A new function $f_{min}()$ is defined such that `QSearch()` can

1: **procedure** QSEARCH($\widehat{\mathcal{A}}, f(\cdot)$)
2: $i \leftarrow$ measure ($\widehat{\mathcal{H}}\,|0\rangle$) ▷ sample the superposition
3: **if** $f(i) == 1$ **then**
4: found $\leftarrow True$
5: **else**
6: $M_0 \leftarrow 0$; $l \leftarrow 0$; found$\leftarrow False$
7: c constant, such that $c \in (1, 2)$
8: **while** not found and $M_l < \left\lceil \sqrt{N} \right\rceil$ **do**
9: $l \leftarrow l + 1$
10: $M_l \leftarrow \min(\lceil c^l \rceil, \left\lceil \sqrt{N} \right\rceil)$
11: $r_l \leftarrow$ randInt$(1 \ldots M_l)$
12: $|\Psi\rangle \leftarrow \widehat{\mathcal{Q}}^{r_l}\widehat{\mathcal{A}}\,|0\rangle$
13: $i \leftarrow$ measure ($|\Psi\rangle$)
14: **if** $f(i) == 1$ **then**
15: found $\leftarrow True$
16: **end if**
17: **end while**
18: **end if**
19: **return** (i, found)
20: **end procedure**

Fig. 3. Adaptive Exponential Search algorithm.

still be used. $f_{min}()$ unites the two conditions imposed by $f()$ and $g()$:

$$f_{min}(x, currMin) = \begin{cases} 1 & \text{if } f(x) == 1 \text{ and } g(x) < currMin \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Note that $f_{min}()$ is similar to $f_{int}()$, the intersection operator defined in the paper's introduction. $f_{min}()$ returns 1 for all solutions where $g(x) < currMin$ holds, and not only for $g()$'s minimum, which is unknown. A new quantum oracle, $\widehat{\mathcal{A}}_{min}$, is defined which marks solutions according to $f_{min}()$:

$$\widehat{\mathcal{A}}_{min}|0\rangle^{\otimes n}|0\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle|f_{min}(i)\rangle \tag{5}$$

The minimum finding approach (algorithm 4) iteratively calls `QSearch()` in an attempt to find a value less than the current minimum. The actual minimum is unknown, therefore the number of iterations is limited to some constant $\#IT$. Since initially no estimate is known for the minimum, `QSearch()` is used with operator $\widehat{\mathcal{A}}$ – line 6. Once a solution is found, it is used as the current minimum and operator $\widehat{\mathcal{A}}_{min}$ is used from that point on – line 10. Whenever `QSearch()` succeeds in finding a new minimum, which is less that the current threshold, this threshold is updated – line 12.

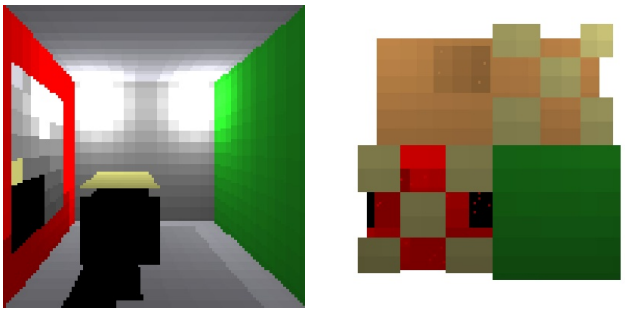
This algorithm has been demonstrated to find the minimum, with high probability, with complexity $\mathcal{O}(\sqrt{N})$, therefore maintaining the quadratic advantage with respect to a classical approach [5], [6].

II. Experimental Setup - Scenes Description

We use the following scenes to compare the approaches:

1: **procedure** MINIMUM($\widehat{\mathcal{A}}, f(), g(), \#IT$)
2: $currMin \leftarrow -1$ ▷ no minimum threshold yet
3: $solution \leftarrow -1$; $it \leftarrow 1$
4: **while** $it \leq \#IT$ **do**
5: **if** $currMin == -1$ **then**
6: i, found $\leftarrow$ QSearch ($\widehat{\mathcal{A}}$, $f()$)
7: **end if** ▷ improve on current minimum threshold
8: prepare $f_{min}()$ from $f()$, $g()$ and $currMin$
9: prepare $\widehat{\mathcal{A}}_{min}$ from $f_{min}()$
10: i, found $\leftarrow$ QSearch ($\widehat{\mathcal{A}}_{min}$, $f_{min}()$)
11: **if** found **then**
12: $currMin \leftarrow g(i)$; $solution \leftarrow i$
13: **end if**
14: $it \leftarrow it + 1$
15: **end while**
16: **return** solution
17: **end procedure**

Fig. 4. Minimum finding algorithm.

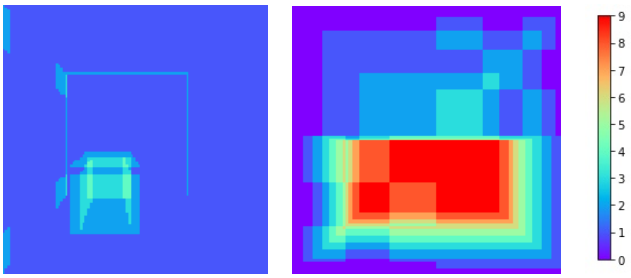


(a) Qornell Box. (b) Depth Complexity scene.

Fig. 5. Scenes used for obtaining experimental results

- **Quantum Cornell Box** – denoted by the Qornell Box (figure 5a). The scene is lit by two point light sources and includes a specular mirror in the left wall. The respective image space depth complexity map is presented in figure 6a. Several versions are used, with the number of geometric primitives, N, ranging from 8 to 512. However, for all these versions at most 6 such primitives are intersected by any ray, allowing for a depth complexity t independent on the total number of primitives;
- **Depth Complexity** – this synthetic scene, with all primitives parallel to the image plane, exhibits varying depth complexity (figure 6b). The upper left and lower right quadrants have a depth complexity of 2 and 9, respectively, with smooth variations around the borders due to perspective projection. The remaining two quadrants have similar depths, but a checkerboard pattern of polygons is included to increase the image plane's spatial frequency. The impact of different depth complexities across the image plane in the number of `QTrace()` iterations (and consequently, oracle evaluations) is assessed.in the paper For scalability analysis there are 2 versions of the scene, with 32 and 64 primitives, both exhibiting exactly the same depth complexity and rendering to the same image

(figure 5b).



(a) Qornell Box scene. (b) Depth Complexity scene.

Fig. 6. Image space depth complexity maps (i.e., number of primitives intersected by each pixel's primary ray) at 128x128 resolution

```
1: procedure f_int(r, p, min)
2:     intersect ← True
3:     for i ∈ ['X','Y','Z'] do
4:         rpc_i ← rpc_eval(r, p, i)
5:         if rpc_i < p.min_i or rpc_i > p.max_i then
6:             intersect ← False
7:         end if
8:     end for
9:     if rpc_eval(r, p,'D') ≥ min then
10:        intersect ← False
11:    end if
12:    return intersect
13: end procedure
```

Fig. 7. Practical $f_{int}(r, p, min)$

III. Practical Quantum Ray Tracing

The current stage of development of quantum hardware is often referred to as Noisy Intermediate Scale (NISQ), imposing severe constraints on the depth and number of gates of near-term quantum circuits [7]. The ray tracing algorithm proposed in the associated paper is still to deep to allow robust execution on current quantum hardware. Validating it on classical simulators of quantum circuits in useful time requires constraining the ray tracing problem in order to make the algorithm practical.

This section describes the restrictions and design options imposed onto the oracle in order to develop a practical quantum ray tracing algorithm. These include constraining the complexity of the scenes and the precision of the 3D coordinates, as well as performing some parts of the ray intersection computation on the classical computer, rather than on the quantum device. There are no fundamental reasons why these restrictions will hold in the future, as error-corrected quantum computers become a reality and the depth / coherence time of quantum circuits progressively increase.

This section starts by describing the geometric setup and processes required for a practical prototype implementation and then develops the associated ray tracing oracle.

A. Geometric Setup

Existing quantum systems impose severe limits on both the depth and width of executable circuits, i.e., on the number of gates along the circuit's longest path and on the number of qubits, respectively. Additionally, no functional units are included to support arithmetic operations over any numerical data type, such as integer or floating point number representations. Support for these operations must be explicitly included in the user's program, increasing the circuit's depth and width. To maintain circuit's complexity within the limits supported by current simulators, this work imposes the following conditions on the geometric setup:

1) all numerical values (e.g. coordinates) are represented using fixed point representation;
2) axis aligned rectangles (AAR) are the only supported geometric primitives.

Rays are parameterized by origin, direction and maximum depth using fixed point representation, as mentioned above.

The oracle prepares the uniform superposition over all $p \in S$ and implements the ray tracing intersection function $f_{int}(r, p, min)$:

$$f_{int}(r, p, min) = \begin{cases} 1 & \text{if } f(r,p) = 1 \text{ and } g(r,p) < min \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

The proposed implementation of the latter is depicted as pseudo-code in algorithm 7.

Each axis aligned rectangle, i.e. each geometric primitive p, is represented by six fixed-point numbers corresponding to the minimum and maximum coordinates along each axis; the minimum and maximum coordinates have equal values for the axis aligned plane embedding the rectangle. The intersection algorithm computes the intersection point between the ray and the plane embedding the primitive p. This intersection point is denoted by `rpc`, standing for **r**ay **p**lane intersection **c**oordinates. The method `rpc_eval(`$r, p, axis$`)` (line 4 of algorithm 7) returns, as a fixed point value, `rpc` for the given axis ('X', 'Y' or 'Z') or, alternatively, the depth of that intersection point, i.e. the length of the ray from its origin up to the ray plane intersection ('D'). Then, given `rpc` and the axis aligned rectangle coordinates, a valid intersection is reported if the former projects into the latter and the depth is less than the currently known minimum depth. Note that `rpc_eval()` calls the minimum finding algorithm, which calls `QSearch()` , which will then execute Grover's quantum search circuit; this is an hybrid quantum-classical algorithm.

B. The practical ray tracing oracle

Figure 8 presents the actual implementation of the oracle $\widehat{\mathcal{A}}$.

The superposition on $index$ is prepared using n Hadamard gates. The operator $\widehat{\mathcal{RT}}_r$ implements, for all primitives p indexed by $index$, the intersection function $f_{int}(r, p, min)$, according the logical sequence sketched in algorithm 7. The axes X, Y, Z and clipping depth are processed in sequence, as illustrated by the sub circuits $\widehat{c}_X$, $\widehat{c}_Y$, $\widehat{c}_Z$ and $\widehat{c}_D$. Each of these $\widehat{c}_{[X,Y,Z,D]}$ operators will set the associated ancilla qubit

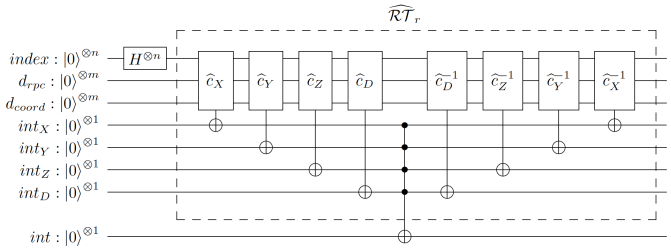


Fig. 8. Operator $\widehat{\mathcal{A}}$.

$int_{[X,Y,Z,D]}$ to $|1\rangle$ if the respective intersection conditions (as described in algorithm 7) are met. The output qubit int is set to $|1\rangle$ if and only if all four ancilla qubits are equal to $|1\rangle$. Due to quantum parallelism these computations occur simultaneously for all primitives p indexed by the superposition in $index$. Finally, all computations on $int_{[X,Y,Z,D]}$ are reversed, such that operator $\widehat{\mathcal{RT}}_r$ only changes the quantum state of the output qubit int.

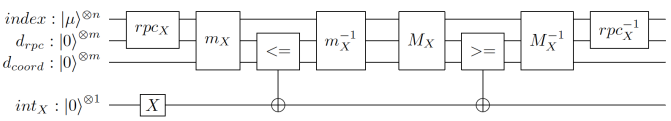


Fig. 9. Operator $\widehat{c}_X$.

Figure 9 depicts the quantum circuit $\widehat{c}_X$ processing axis X and generalizes to $\widehat{c}_Y$, $\widehat{c}_Z$ and $\widehat{c}_D$:

- rpc_X -computes the X coordinate of the intersection point between the ray and the plane containing the geometric primitive, writing it into the d_{rpc} register;
- m_X - (respectively M_X) loads the minimum (respectively maximum) X coordinate of each primitive indexed by $index$, writing it into the d_{coord} register. These operators can be understood as the loading of the scene description, on a per axis basis, into the quantum circuit. Their role is the same as the $\widehat{Load}$ operator described in the paper, but instead of loading in parallel the six parameters describing each primitive, these parameters are loaded in sequence, allowing for a six-fold reduction on the number of qubits;
- $\leq$ - (respectively $\geq$) negates qubit int_X for those primitives in the superposition where $m_X \leq rpc_X$ (respectively $M_X \geq rpc_X$). Since i_X is initially set to $|1\rangle$ (gate X), its final value will be $|1\rangle$ if and only if $m_X \leq rpc_X \leq M_X$;
- m_X^{-1} -(and M_X^{-1}, rpc_X^{-1}) all intermediate computations are reversed, such that only the quantum state of superposition $index \otimes int_X$ is changed by operator $\widehat{c}_X$.

Evaluating the ray plane intersection point – Evaluating $rpc_{[X,Y,Z,D]}$ (the coordinates and depth of the intersection point between the ray and the plane embedding the primitive) requires performing a significant number of fixed point operations, which is well above current quantum systems' capabilities. There is no theoretical reason why these computations can not be performed by the quantum computer; in fact they can and there are quantum fixed and floating point libraries available [8], [9], [10], [11]. However, including these computations in the quantum circuit would increase its depth, width and number of gates far beyond what can be expected to be executable in a robust manner in the short/medium term. Even simulating such quantum circuits would be infeasible due to exponential growth of time and memory requirements. To resolve this, the hybrid quantum classical renderer used for the experiments in this work classically computes these coordinates. The $rpc_{[X,Y,Z,D]}$ components appropriately set the qubits in d_{rpc}, but the coordinates are looked up on a classically evaluated table, rather than evaluated by the quantum circuit. Classically evaluating these coordinates is $\mathcal{O}(N)$, where N is the number of primitives. Quantum evaluation of these coordinates would be $\mathcal{O}(1)$ due to quantum parallelism. Whereas this option seems to compromise this paper's goal of presenting a $\mathcal{O}(\sqrt{N})$ complexity ray tracing algorithm, we argue that:

- the intersection evaluation is still performed using the quantum circuit;
- 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 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.

REFERENCES

[1] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. of the 28th Annual ACM Symposium on Theory of Computing*, ser. STOC '96. ACM, 1996, pp. 212–219. [Online]. Available: http://doi.acm.org/10.1145/237814.237866

[2] G. Brassard, M. B., M. Mosca, and A. T., "Quantum amplitude amplification and estimation," *Contemporary Mathematics*, vol. 305, 2002.

[3] M. Boyer, G. Brassard, P. Hayer, and A. Tapp, "Tight bounds on quantum searching," *Fortschritte der Physik*, vol. 46, no. 5, 1998.

[4] J. L. Bentley and A. C.-C. Yao, "An almost optimal algorithm for unbounded searching," *Information Processing Letters*, vol. 5, no. 3, pp. 82 – 87, 1976.

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

[6] A. Ahuja and S. Kapoor, "A quantum algorithm for finding the maximum," *arXiv: Quantum Physics*, 1999.

[7] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, no. 79, 2018.

[8] T. Haener, M. Soeken, M. Roetteler, and K. M. Svore, "Quantum circuits for floating-point arithmetic," in *Reversible Computation*, J. Kari and I. Ulidowski, Eds. Springer International Publishing, 2018, pp. 162–174.

[9] T. Häner, R. Roetteler, and S. Krysta, "Optimizing quantum circuits for arithmetic," *ArXiv*, vol. arXiv:1805.12445 [quant-ph], 2018. [Online]. Available: https://doi.org/10.48550/arXiv.1805.12445

[10] Y. Zhang, "Four arithmetic operations on the quantum computer," *Journal of Physics: Conference Series*, 2020.

[11] R. Seidel, N. Tcholtchev, S. Bock, C. K.-U. Becker, and M. Hauswirth, "Efficient floating point arithmetic for quantum computers," 2021. [Online]. Available: https://arxiv.org/abs/2112.10537