Implementation of Ray Tracing with Reinforced Learning

Shuhao Tan
University of Maryland
College Park, Maryland
shuhao@cs.umd.edu

ABSTRACT

This paper describes an implementation attempt of a recent paper describing a ray tracer with reinforced learning technique. The implementation is based on a set of simplified assumptions. This paper will present and describe how the assumptions affect the final result and address the effectiveness of the reinforced learning technique through some scenes.

CCS CONCEPTS

• **Computing methodologies** → **Ray tracing**; *Reinforcement learning*;

KEYWORDS

Ray tracking, Reinforced learning, Q-Learning

1 INTRODUCTION

Ray tracing is a popoluar technique to obtain a photo-realistic image from a scene. Usually Monte-Carlo integration is employed over the space of all possible light paths from light sources to the camera, to calculate the light transported to the camera. A central question in this paradigm is to find a good way to reduce the variance generated by the randomness. Ideally, if the probability distribution of the light paths sampled by the procedure matches the transmitted energy distribution in the path space, the variance will be zero. It is generally impossible to achieve this goal since the scene setting can be very complicated. However, if a reasonable approximation of the radiance distribution can be established at each point of interest, one can guide the ray accordingly and expects to see decrease in variance. We looked at two papers [1, 2] exploiting this idea, and implemented one of the two papers in a simplified form.

More specifically, we will try to collect the radiance information in the progress of rendering, and construct an approximation on the fly. We will then use the learnt information to guide the ray tracker in the future rounds of rendering and continually refine our approximation along the way.

We will first review the two papers involved. We will then discuss the decisions we made in our implementation. We will then proceed to analyse and evaluate the technique through two representative scenes. We will conclude the paper with some discussion on the technique.

2 LITERATURE REVIEW

2.1 Overview of the papers

Müller, Gross and Novák [2] proposed a method blending progressive reinforced learning in to a path tracer with russian roulette. They build an adaptive spatial binary tree on top of the scene, with each node equipped with an adaptive quad tree. Each node of the spatial binary tree captures a small volume in the scene, while the quad tee represents the incident radiance field of the volume. The rendering is in multiple passes.

In each pass, whenever the next sample direction of the ray is needed, the node in which the starting point lies is located, and the direction is sampled according to the weights in the quad tree associated. After the whole path is constructed, the radiance along the path will be recorded in another quad tree in each leaf node visited in the spatial binary tree. After one pass is finished, the image of the previous pass is discarded. The spatial binary tree will be refined according to how many times a node is visited, and the quad tree will be refined according to how much flux is passed in each cell.

Dahm and Keller [1] use a different approach. Instead of performing the rendering in multiple pass, they keep updating the approximation on the fly. Moreover, they update the approximation without the knowledge of the complete path. Q-learning is employed to perform approximation update at each hop. The authors notice the similarity between the light transport equation

$$L(x,\omega) = L_x(x,\omega) + \int_{\mathscr{S}^+(x)} L(h(x,\omega_i),-\omega_i) f_s(\omega_i,x,\omega)\cos\theta_i d\omega_i \tag{1}$$

and the Q-learning update equation

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha(r(s,a) + \int_A \pi(s',a')Q(s',a')da' \tag{2}$$

where s is state and a is the action of the procedure. So they treat point location as state, the incident radiance as reward and choosing sample direction as action taken. Now the problem will reduce to guide the ray along a path with high energy. Hence at the end of the day the Q function should approximate the distribution of the radiance field.

The author model the state space on the scene surface. A Voronoi Decomposition of the scene surface is built, and each Voronoi cell is deemed as one state. a hemisphere is stratified and stored for each cell and is used as the discretized version of the Q-Function, also known as Q-Table. Whenever sample direction is needed, a point location query on the Voronoi Decomposition is made, and the direction is sampled on the associated hemisphere according to the weight. After that, the entry on the hemisphere gets updated according to the Q-Table update equation.

2.2 Comparison of the approaches

The two approaches described above are very different in many aspects.

Müller, Gross and Novák use 3D grid-like structure over the scene while Dahm and Keller use 2D decomposition on the scene

06 Dec 2017.

surface. Moreover, the structure of the spatial binary tree gets updated during the rendering process while the decomposition of the scene surface is more like a pre-processing step and remain unchanged afterwards. Intuitively, using the scene surface would be more memory efficient and it is indeed shown in their papers. A drawback would be that the resolution of the decomposition need to be specified mannually before rendering which might not be ideal for an automated process. Due to this difference, Dahm and Keller only need to store a hemisphere for each cell since all the points in the same cell share similar normal direction.

Apart from that Müller, Gross and Novák separate different passes, using a static configuration of binary spatial tree and quad trees to guide the rays, while Dahm and Keller update the distribution on the fly. It is easy to see that the former one has theoretical guarantee that the final image is an unbiased approximation of the MC integration. Dahm and Keller's approach doesn't have such guarantee, while they do have a convergence guarantee based on the property of Q-Learning. As we will see in later sections, this might be a unfavorable behavior, as the result will highly depend on the rendering order of the renderer.

There are some other minor differences which we will address in the Evaluation section.

3 TECHNICAL DETAILS

We will go through some technical points and theoretical guarantees from Dahm and Keller in this section.

3.1 Q-Learning and Light Transport Equation

We start by presenting what it means by reinforced learning. In the setting of single agent reinforced learning, we assume an agent is performing a series of action. The environment has an unknown yet stable reward function that grants the agent reward based on the state and the action of the agent. The goal of the reinforced learning is to decide the action to take given a state that maximizes the prospective reward.

Q-Learning [4] attacks this problem by defining a Q-function that approximates the total prospective reward after taking an action. More formally, given an action a from a set of possible actions A, and a state s from the state space S, $Q(s,a)$ approximates the reward by taking action a when the state is s. Intuitively, we may want to take the action of the highest prospective reward. And we can update the Q function by

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha\left(r(s,a) + \gamma \max_{a' \in A} Q(s',a')\right) \quad (3)$$

where α is the learning rate, γ controls how optimistic we are about the approximation for the next hop, and $r(s,a)$ is the reward for taking action a and transit to state s'.

Alternatively one can update the Q function by taking weighted average among the possible next actions:

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha\left(r(s,a) + \gamma \sum_{a' \in A} \pi(s',a')Q(s',a')\right) \quad (4)$$

which is known as expected SARSA [3]. Extending this equation to continuous case is natural:

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha\left(r(s,a) + \gamma \int_{a' \in A} \pi(s',a')Q(s',a')da'\right) \quad (5)$$

Comparing this to Equation 1, one can easily see the two equations are of very similar form. Specifically, if we treat the point location as state, the direction of the outbound ray, BSDF value with cosine term as the weight, and transported light radiance as reward, we can identify each component in Q-Learning with a component in light transport equation. Hence we can have a Q function update equation as Equation 2.

Given the update scheme of Q function, it is natural that we sample the our ray with probability propotional to the predicted Q function, and update it at each step.

3.2 Convergence Guarantee

It is favorable that the rendering result converges. This is achieved by controlling the learning rate α. As long as a vanishing function $\alpha(s,a)$ is used, it is guaranteed that Q function converges. Though it should be noted that, the representation of Q function will be discrete and only the Q function along the sampled light path is updated each time, so there is no uniform convergence guarantee of Q function with respect to time.

4 IMPLEMENTATION

We chose to implement the paper by Dahm and Keller within the framework of nori.

4.1 Data Structure

Instead of using the scene surface decomposition, we choose to lay a uniform 3D grid over the scene space. Given that the grid will be sparse, we store each cell in a hash table to support dynamic creation and fast look up.

The Q-Table is stored as a 2D array in each cell. It is regarded as a uniform grid on $[0,1) \times [0,1)$. A sampled value on $[0,1) \times [0,1)$ according to the Q-Table will then be warped using **Warp::squareToUniformHemisphere**. Because we may have different normals within the same cell, we implement two variants of Q-Table. A simpler version stores only a hemisphere of the Q-Table, ignorant of how the normal is pointing. Because of the fixed normal direction, a 2D range tree is placed over the Q-Table to support fast update and sample. A version that stores the Q-Table for the whole sphere is also implemented. Each time when the normal is known, the hemisphere is extracted from the whole table and operations are performed on the extracted part of the whole table. Because the normal can be pointed any direction, so no accerlerating data structure is used in this version.

The resolutions of the grid and Q-Table are adjustable via scene file. The learning guider is abstracted as an independent class, so the choice of Q-Table to use, or even other ray guiding mechanisms other than Q-Learning can be specified via scene file.

4.2 Rendering Technique

The original paper only adds the learning technique on top of the simplest form of path tracer. Apart from implementing that, we

also incorporate the the learning technique into a path tracer with path reusing, russian roulette and multiple importance sampling. Again, the additional techniques can be configured in scene file.

4.3 Progressive Rendering

We borrow the idea of progressive rendering from Müller, Gross and Novák, by allowing the renderer to exponentially increase the sample count per pixel and only keeps the last rendered image. The idea is that, Q-Table may take a few rounds to reach to convergence, and it is favorable to keep an image with little change in Q-Table to get a nearly unbiased image. The concrete impact of the technique will be discussed in later sections.

4.4 Side Products

As a side product, we implement a Blender add-on to import a nori scene into Blender. This allows us to tweak the scene easily as we like.

We also implement a visualization integrator to visualize the Q-Table we gathered though the rendering.

5 EVALUATION

We used two scenes to evaluate our result. The **DOOR** scene (made by Miika Aittala, Samuli Laine, and Jaakko Lehtinen, available at https://mediatech.aalto.fi/publications/graphics/GMLT/, licensed under Creative Commons Attribution 4.0) features two rooms where only one room has a light source, and the other room contains furniture. Two rooms are connected by a slightly opened door. The **WALL** scene (made by ourselves) features a single room with a wall in the middle. Only half of the room has light source, and there are objects in the other half. Both of them has a key feature that part of the scene is mostly contributed by a relatively small set of light paths.

5.1 The DOOR Scene

The images produced are shown in Figure 1. All of them use **total** of 1024 samples per pixel. (a) and (b) use 512 samples per pixel to produce the final image while casting another 511 samples per pixel during progressive rendering.

Performance. Surprisingly (a) takes the least time while (c) and (e) take slightly longer. (b) and (d) take significantly longer time to produce with a ratio of about 1.7. Running both (a) and (e) in progressive mode reveals that, initially (a) is much slower than (e), however, after the number of sample increases to 64, (a) becomes much faster and gradually outperforms (e). We believe this shows that, after initial learning, the paths chosen by the tracer become much shorter on average, so the performance increases. (d) uses the same sampling distribution all the time, so the average path length never decreases and results in bad performance. The reason that (b) is very slow might due to the large overhead of performing multiple importance weight calculation which requires multiple queries to the PDF from Q function.

With vs. Without Reinforced Learning. It is very clear that, by compare (a), (c) with (d), the reinforced learning shows significant improvement over simple cosine weighted path tracer. The handle of the left teapot is barely visible in (d) while it is very obvious in (a) and (c). We also observe the performance boosting by applying the learning technique as we mentioned earlier. We believe this proves that learning technique is effective to improve a simple path tracer.

With vs. Without Progressive Rendering. Nori renders the scene block by block. It turns out that this exposes a drawback of the technique. The Q Tables across the scene are not updated in the same pace. At very start, when not much information has been learnt, the variance is very large, as is seen at the center of (c). So this shows that this technique does rely on the specific behavior of the renderer, especially on the rendering order. We notice that by adding prograssive rendering, we can ensure a rather uniform update of Q Tables, (a) has a much more uniform variance than (c).

It is interesting though, if we look at (c) from center to the edge, the image itself demonstrates the improvement thanks to the learning technique as time passes.

Hemisphere vs. Sphere Q Table. Though not present in the figure, we actually compared the hemisphere version of Q Table with the sphere version offline. It turns out that despite a huge performance penalty due to lack of acceraleration structure for sphere version, we didn't observe any significant visual improvement in the result.

Choice of Learning Rate. We compared using learning rate $\alpha = 0.1$ with a vanishing learning rate. We confirm the claim in the original paper that a vanishing learning rate tends to yield better result in term of image quality.

Additional Rendering Technique. We observe that reinforced learning technique doesn't seem to accomodate well with other techniques like russian roulette, path reuse and multiple importance sampling. We don't get visual improvement, and there is a huge performance penalty. We will discuss the possible reasons and implications in great details in the later section.

Generated Q Function. We visualize the generated Q function in (f). As we can see, our result is very similar to the result in the original paper which shows that the Q function indeed approximate the incident radiance distribution at the end.

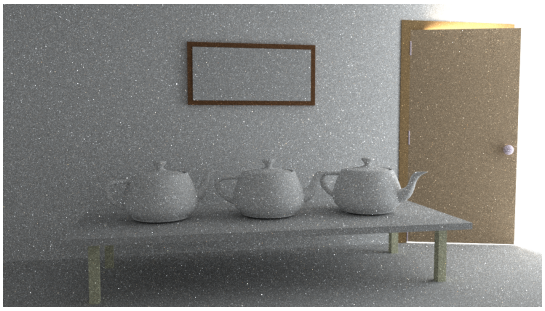
Memory footage. We dump the Q function at the end of the rendering. The table takes about 180MB. We believe this is solely due to the uniform grid choice we made.

5.2 The WALL Scene

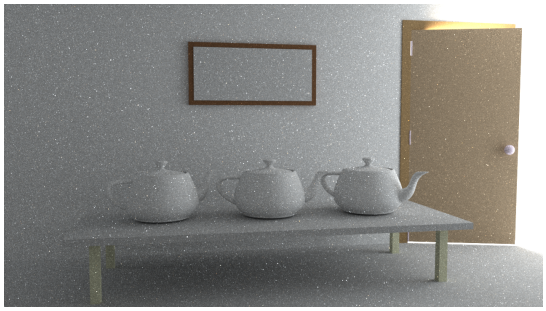
The WALL scene consists two parts of the same room, separted by a wall cutting in the middle, leaving some space to left light pass through. A white cone and a red suzanne are placed in the dark side. the dark side is illuminated only by indirect illumination. In Figure 2, a **total** of 32 samples per pixel is used for each image.

Almost all the points we made in the previous subsection can be re-stated for this scene. We would like to emphacize that, we can see (a) works poorly on the bright side compared to (c). This implies that, if direct illumination is available, traditional techniques are highly favorable over the learning technique.

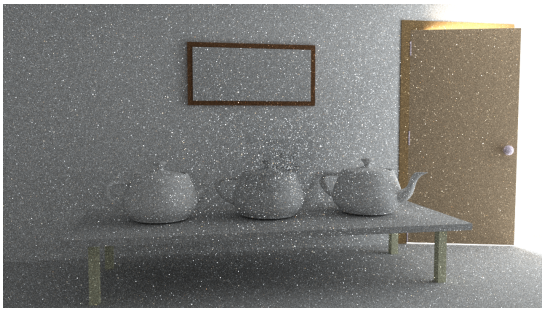
We would also mention that the visualization in this case also indicates the correct learning behavior of our implementation by showing a higher Q function value towards the open end.



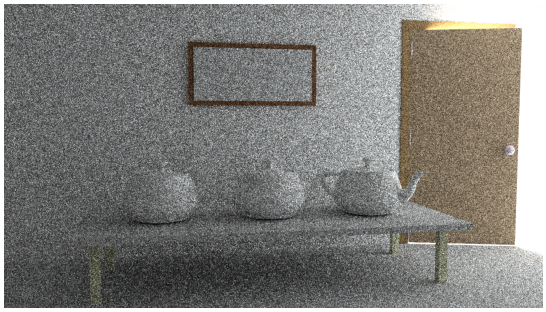
(a) Reinforced Learning with Progressive Rendering



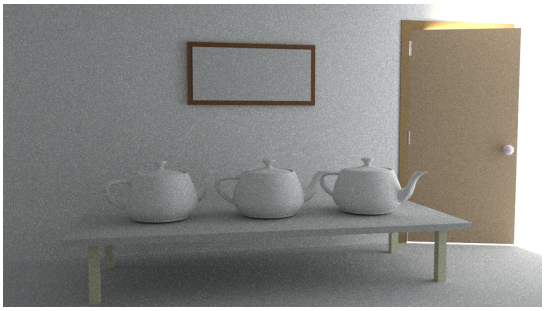
(b) Reinforced Learning with Russian Roulette, Path Reuse, and Multiple Importance Sampling



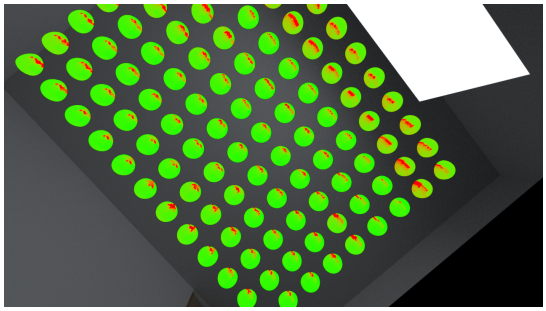
(c) Reinforced Learning without Progressive Rendering



(d) Simple Path Tracing



(e) Russian Roulette, Path Reuse, and Multiple Importance Sampling



(f) Q Function in the lighting room

Figure 1: The DOOR scene

6 DISCUSSION

We will provide in-depth discussion on our findings when we try to implement the paper.

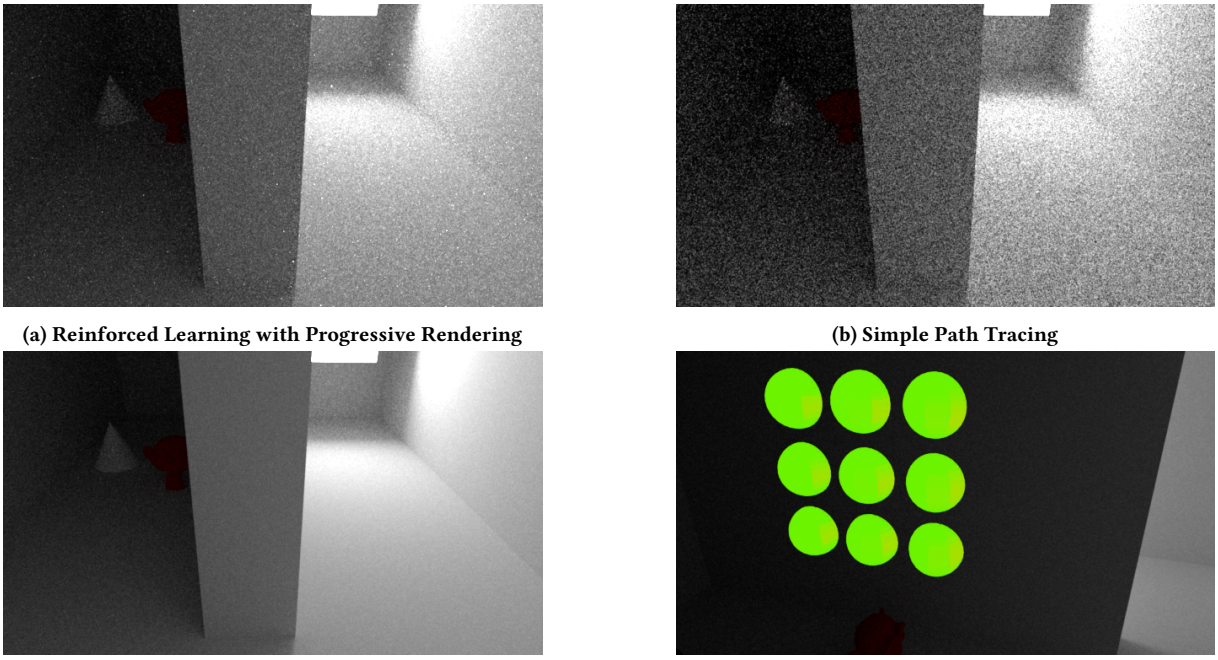
6.1 Grid vs. Scene Surface

We would like to first address the implementation decision between grid and scene surface. We believe that a uniform grid has inherent flaws to be applied in general scenes.

The main problems arises due to the representation of the Q Table. We stratify either the hemisphere or the sphere and store one value for a single stratrum. If we are using a grid, there is no gurantee that all the surface points within a cell share similar normals. It might happen that there are corners in the cell, or we can even have completely opposite normals within one cell like a double sided plane.

It isn't really significant if the scene is aligned to the axes, as then the correspondent hemisphere for axis-aligned normals are also perfectly aligned to stratrums. Problem arises when we have a huge change of normals in one cell and they are not axis-aligned. As we see in Figure 3, there are visible artifacts, dark and bright stripes, when number of samples is low. The door can be seen as a double-sided surface. The back of the door is in a bright room while the front is in a relatively dark room. On the stratified sphere, directions nearly parallel to the door surface cannot be distinguished. So in the view of a point on the front of the door, directions that are nearly parallel to the door have very large energy, while points on the back see the directions that are nearly parallel have relatively low energy.

This is why we can observe black stripes on the door. Black regions are where rays are guided to nearly parallel location where energy isn't really high but the PDF assigned is large. This error



(a) Reinforced Learning with Progressive Rendering

(b) Simple Path Tracing

(c) Russian Roulette, Path Reuse, and Multiple Importance Sampling

(d) Q Function on the darker side

Figure 2: The WALL Scene

Figure 3: Artifacts due to non-aligned surface (32spp)

would propagate easily in the early stage and would stay in the approximation of Q function due to the vanishing learning rate. We suspect the random high frequency noise observed in our results mainly roots from this.

By using scene surface and Voronoi Decomposition described in the original paper, we can ensure that normals in each cell are almost the same and such artifact won't appear. Alternatively in the other paper, the adaptive approach of the grid and sphere stratification can gradually separate the ambiguos parts in the scene and avoid the artifact in the final image.

6.2 Additional Rendering Techniques

In this subsection, we will discuss the reasons we believe that the learning technique cannot be combined naturally with some traditional techniques.

Russian Roulette. The idea of russian roulette is to keep the expectation of path length managable. When combined with reinforced learning, russian roulette will likely weaken the speed-up we get from shorter and shorter path length we acquire from better guiding during the rendering process. Moreover, since we may not get a full light path at the end, we may not add new reward/radiance from light source into Q Tables.

Path Reuse. In a scene with long and complicated light path, it is likely that a path will be blocked by objects when the path is short. When the path is directly visible to the light source, the majority of the contribution of all possible extended path should come directly from the light source. This is in line with the behavior that the Q

function guides the ray to high energy direction. So path reuse may not bring significant improvement when combined with reinforced learning.

Multiple Importance Sampling. We believe multiple importance sampling is inherent in reinforced learning. As the size of emitter will be reflected accordingly on the radiance field, directly sampling according to the Q function should incorporate multiple importance sampling within itself. However, we believe more general form of multiple importance sampling could be useful to improve the quality.

6.3 Light Transport Equation Revisited

We will provide what we belive a clearer insight of the two papers in terms of the light transport equation. Let $L_i(x, \omega)$ be the radiance of fewer than or equal to x bounces from emitter to x with direction ω. This can be viewed as a vector in the function space. Let T be the transport operator on L that adds a bounce.

$$T(L)(x, \omega) = L(x, \omega) + \int_{\mathscr{S}^+(x)} L(h(x, \omega_i), -\omega_i) f_s(\omega_i, x, \omega) \cos\theta_i d\omega_i \tag{6}$$

Note T is linear, so we can also view this as a Markov Process in function space. Q-Learning basically selects a finite subset of vector entries (a.k.a points in function domain), and apply a one-step transition restricted to those entries. In the view of photons, this is equivalent to moving photons with a one-step random walk.

The other paper takes another view. One can also write

$$L(x, \omega) = \int_{\text{paths from emitter to } x} \delta(\omega, \omega') L(\hat{p}) f(\hat{p}) d\hat{p} \tag{7}$$

where $\hat{p}$ is a path, ω' is the direction of path into x, and f is the PDF of $\hat{p}$. This way we will see the other paper is basically performing a Monte-Carlo integration in the path space.

6.4 Future Work

Multiple Importance Sampling Between Traditional Technique and Learning Technique. We see traditional techniques performs extremely well in direct illuminated environment. So it woule be ideal if we can combine the two technique. One may try to construct the result using both techniques together, and then use multiple importance weight to weigh the two results to get better result combined.

Adaptive Scene Surface Decomposition. Currently the decomposition of scene surface is static, it would be extremely useful if there is a way to adaptively refine the decomposition during the rendering process.

Neural Netowrk. There has been work on approximating Q function using a neural network. The main difficulty here is how to train the neural network, as the training time should be part of the rendering time to render general scenes.

Combining Approximations of Radiance. As we see in Section 6.3, there are two different views of approximating incident radiance. They are not conflicting though. A natural question would be whether it is possible to combine the two to accerlerate the learning.

Exploiting Continuity. We know the incident radiance field is smooth. Currently we update only one Q Table entry each time. A natural question will be whether we can exploit the continous nature of radiance field, and perform a weighted update on multiple entries in a more refined Q Table.

7 CONCLUSION

Through implementing the path tracer with reinforced learning, we gained some insight in how to approximate incident radiance field. We acquired both positive and negative results towards the learning technique. We offered explanation on the negative results and suggested future work to remedy the drawbacks. We believe incorporating Machine Learning techniques into ray tracing is definitely helpful and is yet to be studied more thoroughly.

REFERENCES

[1] Ken Dahm and Alexander Keller. 2017. Learning Light Transport the Reinforced Way. *CoRR* abs/1701.07403 (2017). arXiv:1701.07403 http://arxiv.org/abs/1701.07403

[2] Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical Path Guiding for Efficient Light-Transport Simulation. *Computer Graphics Forum* 36, 4 (June 2017), 91–100. https://doi.org/10.1111/cgf.13227

[3] Richard S Sutton and Andrew G Barto. 2017. *Introduction to reinforcement learning, 2nd edn.* MIT Press Cambridge, MA, USA.

[4] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8, 3-4 (1992), 279–292.