Prelude

A typical dissertation will be structured according to (somewhat) standard sections, described in what follows. However, it is hard and perhaps even counter-productive to generalise: the goal is *not* to be prescriptive, but simply to act as a guideline. In particular, each page count given is important but *not* absolute: their aim is simply to highlight that a clear, concise description is better than a rambling alternative that makes it hard to separate important content and facts from trivia.

You can use this document as a LaTeX-based [**?**, **?**] template for your own dissertation by simply deleting extraneous sections and content; keep in mind that the associated `Makefile` could be of use, in particular because it automatically executes to deal with the associated bibliography.

You can, on the other hand, opt *not* to use this template; this is a perfectly acceptable approach. Note that a standard cover and declaration of authorship may still be produced online via

`http://www.cs.bris.ac.uk/Teaching/Resources/cover.html`

Dissertation Type: research

DEPARTMENT OF COMPUTER SCIENCE

How effective are Temporal difference learning methods for reducing the nur

Callum Pearce

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Engineering in the Faculty of Engineering.

Friday 19th April, 2019

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of MEng in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Callum Pearce, Friday 19th April, 2019

Contents

List of Figures

List of Tables

List of Algorithms

List of Listings

Executive Summary

In the field of Computer Graphics, Path tracing is an algorithm which accurately approximates global illumination in order to produce photo-realistic images. Path tracing has traditionally been known to trade speed for image quality. This is due to the lengthy process of accurately finding each pixels colour, whereby many light rays are fired through each pixel into scene, then directions for each ray are continually sampled until it intersects with a light source. Due to this, a variety of Importance sampling algorithms have been designed to avoid sampling directions which lead to rays contributing no light to the rendered image. The paths formed by sampling rays in these directions are known as zero contribution light paths. By not sampling zero contribution light paths, it is possible to significantly reduce the noise in rendered images using the same number of sampled rays per pixel in path tracing

Recently a Temporal Difference learning method was used by Nvidia to achieve impressive results in Importance sampling within a Path tracer. The algorithm essentially learns which directions light is coming from for a given point in the scene. It then uses importance sampling to favour shooting rays stored in those directions, reducing the number of zero contribution light paths sampled. With this success, there is plenty of potential to experiment with other Temporal Difference learning methods, particularly Deep Q-Learning. It is also important to assess both of these methods on their ability to accurately approximate Global Illumination to produce photo-realistic images. From this, my goal is to investigate the ability of two different temporal difference learning algorithms ability to reduce the number of zero contribution light paths in path tracing, whilst still accurately approximating global illumination. More specifically, the first temporal difference learning method will be that proposed by Nvidia, and the second will be my designed Neural-Q path tracing algorithm. I will be comparing these two methods in order to test the following hypothesis:

> The Neural-Q path tracer is further able to reduce the number of zero contribution light paths than an Expected SARSA Path tracer proposed by Nvidia, whilst still accurately simulating Global Illumination.

Outcomes

- Which is better able to reduce the number of zero contribution light paths expected SARSA or Deep Q-learning
- Can Expected SARSA learning handle multiple lights well in a scene & deep q-learning

Main areas of work

- I have written x lines of code to build a Path tracing engine from scratch which supports a variety of GPU accelerated Path tracing algorithms I have experimented with.
- I have spent x hours researching into the field of efficient light transport simulation for ray-tracing techniques.
- I have spent x hours researching into Reinforcement learning, particularly Temporal Difference learning and Deep Reinforcement learning, neither of which I have been taught before.
- I spent x hours implementing and validating the on-line Expected SARSA Path tracing algorithm proposed by Nvidia, which required me to implement the Irradiance Volume data structure as a prerequisite.
- I have spent x hours designing, implementing and analysing my own on-line Deep Q-learning Path tracing algorithm, along with a neural network architecture designed for the algorithm.

Supporting Technologies

1. I used the `SDL2` library for displaying and saving rendered images from my Path tracing engine.
2. I used the `OpenGL` mathematics library to support low level operations in my Path tracing engine. It includes GPU accelerated implementations for all of its functions.
3. I used the `CUDA Toolkit 10.1` parallel computing platform for accelerating Path tracing algorithms. This means the `CUDA nvcc` compiler must be used to compile my Path tracing engine.
4. All experiments were run on my own desktop machine with an Nvidia `1070Ti` GPU, Intel `i5-8600K` CPU and 16GB of RAM.
5. I used the C++ API for the `Dynet` neural network framework to implement all of my Neural Network code as it is able to be compiled by the `CUDA` compiler.

Notation and Acronyms

TD learning : Temporal Difference learning

Acknowledgements

An optional section, of at most 1 **page**

It is common practice (although totally optional) to acknowledge any third-party advice, contribution or influence you have found useful during your work. Examples include support from friends or family, the input of your Supervisor and/or Advisor, external organisations or persons who have supplied resources of some kind (e.g., funding, advice or time), and so on.

0.0.1 Plan

1. Carl Henrik Ek - Validating my understanding of deep reinforcement learning
2. Neill Campbell - Deep reinforcement learning strategy

Chapter 1

Contextual Background

This chapter explains on a high level what path tracing is and how it accurately simulates light transport. Then importance sampling ray directions in light transport simulation is discussed, and how it can potentially reduce the number of zero contribution light paths and the associated benefits with this. Temporal difference learning as a branch of reinforcement learning is then introduced, along with how it can be used in importance sampling ray directions towards light sources. With a conceptual overview of theory my work is based on, I take a look at recent work which contributes to real-time accurate light transport simulation which my work aims to contribute to. Finally, an overview of the objectives and significant challenges of my investigation are described.

1.1 Path Tracing for Light Transport Simulation

Path Tracing is a Monte Carlo method for rendering photo-realistic images of 3D scenes by accurately approximating global illumination [5]. Figure **??** summarises on a high level how forward Path tracing produces a 2D image of a 3D scene. For each pixel multiple rays are shot from the camera through the pixel and into the scene. Any ray which intersects with an area light terminates, otherwise a new direction is sampled for the ray and it is fired again. This process is repeated until all rays have intersected with an area light, at which point the pixel colour value can be found by averaging the colour estimate of each ray fired through that pixel. Each rays colour estimate is calculated based on the material surface properties it intersects with before intersecting with the light and the intersected area lights properties. The more rays shot through each pixel (also known as samples per pixel), the more visually accurate the rendered image becomes, but at a higher computational cost.

Path tracing simulates global illumination, meaning it accounts for both direct and indirect illumination. Direct illumination being rays of light emitted from a light source, which reflect off exactly one surface before reaching the camera in the scene. Whereas indirect illumination are ray of light which reflect 2 or times before reaching the camera. In **??**, an identical scene is shown with only direct illumination (left) and the other with global illumination (right). The globally illuminated scene displays a range of effects due to Path tracings ability to accurately simulate light transport, which is not the case for the directly illuminated scene. Where light transport simulation refers to firing and summing up the contributions of light transport paths that connect from the camera to light sources [10], such as those displayed in **??**. For example, effects such as (a) colour bleeding, (b) soft shadows, and (c) indirect diffuse lighting are a product of accurate light transport simulation.

Light transport simulation methods are able to produce many complex light transport effects by a simple single pass of a rendering algorithm. This allows artists to increase productivity and perform less manual image tweaking in the production of photo-realistic images. Due to this, the Computer Graphics industry has seen a large resurgence in research and usage of light transport simulation rendering methods in the past decade [11].

My work in this thesis focuses on developing and assessing importance sampling techniques using Temporal Difference learning methods for light transport simulation in forward Path tracing. In particular, More specifically, for any intersection point in a 3D scene, I attempt to create an AI agent that learns and samples in directions light is coming from, reducing the total number of zero contribution

light paths. A zero contribution light path is one whose estimated colour values are almost zero for all (R, G, B) components, hence, they contribute almost no visible difference to the rendered image. We should instead focus our sampling on light paths which do contribute to the image, reducing the noise in pixel values and bringing them closer to their true values for the same number of sampled rays per pixel. Meaning, Importance sampling can reduce the number of rays needed to be sampled per pixel in order to receive a photo-realistic (also known as converged) image from Path tracing. An example of this reduction in noise can be see in **??**, where the naive forward Path tracing algorithms output is compared to Nvidia's on-line reinforcement learning Path tracer using Importance sampling. Note, any light transport simulation algorithm can benefit from the Temporal DIfference learning schemes which will be described [8, 10], as they are all derived from what is known as the rendering equation. This equation is used as a mathematical basis of modelling light transport.

It is paramount that Importance sampling Path tracing algorithms continue to accurately simulates global illumination in order to produce photo-realistic images in a single rendering pass, as this is the major selling point of Path tracing over other methods. Therefore, I will also be assessing the accuracy of the global illumination approximation made by the Importance sampling algorithms compared to that of the naive forward Path tracing algorithm.

1.2 Temporal Difference Learning for Importance Sampling Ray Directions

There are three important unanswered questions up to this point; a) what is temporal difference learning? b) How can temporal difference learning methods be used to importance sample new ray directions for a given intersection point in the scene? c) Why use temporal difference learning methods over other Importance sampling methods to do so?

1.2.1 What is Temporal Difference learning?

Temporal difference learning, which I will refer to from here on as TD learning, are a set of model free Reinforcement learning methods. Firstly, Reinforcement learning is the process of an AI agent learning what is the best action to take in any given state of the system it exists within, in order to maximise a numerical reward signal [15]. The AI agent is not told which actions are best to take in a given state, but instead it must learn which ones are by trialling them and observing the reward signal. Actions taken may not only affect the immediate reward, but all subsequent rewards received for taking future actions. For example, picture a robot rover whose duty it is to explore the surrounding area as much possible. A state in this case is a position in the world it is exploring, and its action are the directions to move in for a given distance. If it discovers a new area, it receives a positive reward signal. Now, if the robot chooses to explore a given area it may not be able to get back from, say a canyon, the robot is limited to searching areas reachable from the canyon. Hence, all subsequent reward signals are limited to what can be received from exploration of the canyon, compared to not entering the canyon and exploring areas which can be returned from first.

As mentioned TD learning methods are model free methods, meaning the methods do not require a model of the system dynamics they are placed in, instead they learn over time by interacting with the system. In other words, they learn from raw experience [15]. TD methods update their current estimates based on a combination of data received from interacting with the environment, and partly on their current learned estimates without waiting for the final outcome of events, this is known as bootstrapping. To illustrate the concept of bootstrapping, imagine you are driving home from work and you wish to estimate how long it will take you get home. By following a TD learning method, if you hit traffic you can update your current estimate of the time it takes you to drive home based on this new data, and your pre-existing estimate. Whereas compared to another set of Reinforcement learning methods known as Monte Carlo methods, you would have to wait until you got home to update your current estimate of how long it takes to get home from work. Meaning you have to wait for the final outcome before learning can begin, which is not the case for TD learning.

1.2.2 Temporal Difference learning methods for Efficient Light Transport Simulation

One of my main aims to reduce the number of zero contribution light paths sampled in Path tracing by the use of TD learning methods. In order to do so I must formulate the problem a reinforcement learning problem, which is done in detail in Chapter 2. However for a conceptual overview it suffices to explain what a state, action, and reward signal will be in the case of light transport simulation within Path tracing:

- **State**: A 3D intersection position in the scene for a given ray to sample the rays next direction from.
- **Action**: Firing the ray in a given direction (3D vector) from the current state.
- **Reward Signal**: The amount of light incident from the direction the ray was sampled in.

In this reinforcement learning setting, we can use TD-learning methods to create an AI agent which learns by taking different actions in different states and observes their reward signals to find out for each state which actions have the highest valuations. By then converting the action space into a probability distribution weighted by each actions learned valuation, the AI agent will more likely sample non-zero contribution light paths, reducing noise in rendered images. Note, the term valuation means the total expected reward for taking a given action, meaning valuation not only accounts for the immediate reward, but the expected reward for taking all future actions to come until the ray intersects with a light. Also, for the proposed AI agent, current actions can affect future rewards, as when the ray intersects a surface it looses some energy. Therefore, future rewards received after many intersections will be discounted compared to the reward of received immediately to match this behaviour. This means the agent will aim to minimise the average number of intersection a ray makes before intersecting with a light source, making it a good metric to test evaluate against to determine how well the AI agent is performing.

1.2.3 Why use Temporal Difference Learning for Importance Sampling?

Traditional Importance sampling techniques for Path tracing do not take into account the visibility of the object from light. A light blocker is shown in **??**, where the blocking object stops rays from directly reaching the light. Due to the unknown presence of blockers, traditional importance sampling methods can fail to avoid sampling zero contribution light paths. Therefore, scenes which are significantly affected by blockers will not receive the benefits from traditional Importance sampling and can even benefit more from an uniform sampling scheme [14].

Temporal difference learning methods are able to solve this problem [6]. As the AI agent described in the previous section learns which directions light is coming from in the scene and concentrates its sampling towards these directions. Directions leading to blockers will have a low value, hence it is unlikely the AI agent will sample rays in these directions.

1.3 Motivation

Rendering time of my graphics engine is not something I have tried to heavily optimise. I instead focus on producing higher quality images using the same number of samples per pixel in light transport simulation in hope that future work will find ways of optimising my methods for speed. Therefore, my work still aims to contribute to the wider goal seen in computer graphics to use accurate light transport simulation in the rendering of photo-realistic images for complex scenes in real-time. Speeding up the methods I use is a large topic in itself, requiring a deep investigation into the best software, hardware, and parallel programming paradigms to use.

1.3.1 Real time Rendering using Accurate Light Transport Simulation

The motivation for using accurate light transport simulation in real-time comes from the clear superior visual quality of images rendered using this techniques, compared to that of scanline methods which are currently used. Where scanline rendering, also known as rasterizing, is the current computer graphics

industry standard method for real-time rendering. Not only are renders for a wide range of scenes clearly superior from methods which accurately simulate light transport, but they also scale far better with the number of polygons used to build the scenes surfaces. Therefore, scanline rendering for scenes with extremely complex geometry in real-time is currently not and option. Accurate light transport simulation methods therefore have great potential to be used in ultra realistic simulations for applications such as scenario planning and virtual reality learning environments [13]. Also, many games sell realism as one of their most important features, therefore developing photo-realistic graphics in real-time has clear economic incentive for the video games industry which was valued at over $136 by the end of 2018 [2]. An economic incentive can also be seen for the film industry, where reductions in render times lead to a direct saving on compute time, as well as the hardware required to render full length films.

1.3.2 Recent Developments

Due to the incentives, a large amount of research and investment has been focused on purpose built hardware and Deep learning post-processing methods in an attempt to bring accurate light transport simulation into real-time. NVIDIA's Turing Ray Tracing Technology [12] represents a significant leap in the hardware to support light transport simulation. It allows for real-time graphics engines to be a hybrid of both scanline rendering, and ray-tracing. The 20 series Turing GPU architecture has significantly improved the speed of ray-casting for light transport simulation, and has the capacity for simulating 10 Giga Rays per second. However, using this hardware alone with current rendering methods is not enough to perform accurate light transport simulation for complex scenes in real-time.

Post-processing methods are designed to take a noisy input image produced by a render which simulates light transport, and then reconstruct the image to remove the noise present in the image. Generally these methods rely on pre-trained deep neural networks to reconstruct the image far quicker then it would take for the renderer to produce an image of the same visual quality [1]. Once again NVIDIA has made significant advancements in this area with NVIDIA OptiX AI Accelerated Denoiser, which is based on their newly designed recurrent denoising autoencoder [3]. OptiX has been successfully integrated in to many of the top rendering engines which accurately simulate light transport, such as RenderMan [4] and Arnold [7]. Whilst post-processing has significantly reduced the number of samples required to render photo-realistic images, there is still more work to be done to produce these images in real-time.

By using importance sampling by TD learning to reduce the number of samples required for accurate light transport simulation, the same standard of noisy image can be fed into an AI accelerated denoiser with fewer samples per pixel in light transport simulation. Running a rendering engine optimised in this way on purpose built hardware could make accurate light transport simulation for rendering photo-realistic images closer than it ever has been to real-time.

1.4 Challenges and Objectives

As previously mentioned, there already exists an example of TD learning used for importance sampling ray directions in a forward Path tracer [6]. However, further methods of analysis need to be conducted upon this new method to determine its performance for reducing the number of zero contribution l ight paths for different scenes with different settings. It is difficult to assess this as there are infinitely many scenes the method can be used to render, so coming to a clear conclusion is difficult. Another difficult task is that of designing an algorithm for an AI agent to learn what are the favourable directions to sample in a scene are using the deep Q-learning method. This includes some important unanswered questions, such as; is it possible for a deep neural network to model all Q values for a continuous scene space? If so, what is a suitable network architecture? All of which I will describe in more depth in Chapter 3. Then the actual task of implementing such an algorithm in a graphics engine written from scratch is non-trivial due to the technologies which will need to be combined together. The algorithm must also run fast enough to collect large amounts of data from, otherwise a justified conclusion on its performance cannot be made. Therefore, the algorithm will have to be parallelized and run on a GPU.

As previously mentioned, my main goal is to investigate the ability of two different temporal difference learning algorithms ability to reduce the number of zero contribution light paths in path tracing, whilst still accurately approximating global illumination. Which can be broken down in to the following objectives:

1. Reimplement Nvidia's state of the art on-line Temporal Difference learning Path Tracer in order to further investigate its ability to reduce the number of zero contribution light paths.
2. Design and implement an on-line Deep Q-Learning variant of the Path tracing algorithm and investigate its ability to reduce the number of zero contribution light paths sampled.
3. Assess both Nvidia's state of the art on-line Temporal Difference learning Path tracer, and the Deep Q-Learning Path tracer' on their ability to accurately simulate Global Illumination.

Chapter 2

Technical Background

2.1 Monte Carlo Integration and Importance Sampling

The theory of Monte Carlo integration and importance sampling underpins how the noise in images rendered by path tracing can be reduced when using the same number of sampled rays per pixel. Therefore, it is necessary to have a good understanding of Monte Carlo integration and its properties, as well as importance sampling before applying it to path tracing.

2.1.1 Monte Carlo Integration

Monte Carlo Integration is a technique to estimate the value of an integral, Equation 2.1 represents this integral for a one-dimensional function f.

$$F = \int_a^b f(x)dx \tag{2.1}$$

The idea behind Monte Carlo integration is to approximate the integral by randomly sampling points (x_i) to evaluate the integral at, and then averaging the solution to the integral for all the sampled points. More formally, basic Monte Carlo integration approximates a solution to this integral using the numerical solution in Equation 2.2. Where $\langle F^N \rangle$ is the approximation of F using N samples.

$$\langle F^N \rangle = (b-a)\frac{1}{N}\sum_{i=0}^{N-1} f(x_i) \tag{2.2}$$

$$\langle F^N \rangle = \frac{1}{N}\sum_{i=0}^{N-1} \frac{f(x_i)}{\frac{1}{(b-a)}} = \frac{1}{N}\sum_{i=0}^{N-1} \frac{f(x_i)}{pdf(x_i)} \tag{2.3}$$

An important property of Monte Carlo Integration is that it produces an unbiased estimate of an integral. A bias estimate would not converge on the true solution to the integral. Basic Monte Carlo integration only produces a non-bias estimate when sample points x_i are randomly sampled from a uniform distribution. To extend this to Generalized Monte Carlo integration where sample points may be sampled from any distribution, the function evaluated at point x_i must be divided by the probability density function (*pdf*) evaluated at x_i. Generalized Monte Carlo integration is shown in Equation 2.3, which from here onwards I will refer to as Monte Carlo integration. Dividing by the *pdf* ensures the estimate $\langle F^N \rangle$ is unbiased, as areas with a high *pdf* will be sampled far more, but their contribution weighting $(\frac{1}{pdf})$ to final estimate will be lower. Whereas areas with a low *pdf* will be sampled less, but their contribution weighting to the final estimate will be higher to offset this.

Another important property of Monte Carlo integration is that by the law of large numbers, as the number of samples (N) approaches infinity, the probability of the Monte Carlo approximation $(\langle F^N \rangle)$ being equal to the true value of the integral (F) converges to 1. This law is stated in Equation 2.4. By this property Monte Carlo Integration works well for multidimensional functions, as convergence rate of the approximation is independent of the number of dimensions, it is just based on the number of samples using in the approximation. Whereas this is not the case for deterministic approximation methods,

meaning they suffer from what is known as the curse of dimensionality. For path tracing, the integral which is approximated is a 2 dimensional function, hence Monte Carlo integration is used.

$$Pr(\lim_{N\to\infty} \langle F^N \rangle = F) = 1 \tag{2.4}$$

The standard error of the Monte Carlo integration approximation decreases according to Equation 2.6. Where the standard error describes the statistical accuracy of the Monte Carlo approximation. Where σ_N^2 is the variance of the solutions for the samples taken, and is calculated by Equation 2.5 using the mean of the solutions for the samples taken (μ). Due to Equation 2.6, in practice four times as many samples are required to reduce the error of the Monte Carlo integration approximation by a half. Also, the square root of the variance is equal to the error of the approximation, so from here on when I refer to reducing the variance I am also implying a reduction in the error of the approximation.

$$\sigma_N^2 = Var(f) = \frac{1}{N-1}\sum_{i=0}^{N}(f(x_i) - \mu)^2 \tag{2.5}$$

$$\text{Standard Error} = \sqrt{Var(\langle F^N \rangle)} = \sqrt{\frac{\sigma_N^2}{N}} = \frac{\sigma_N}{\sqrt{N}} \tag{2.6}$$

2.1.2 Importance Sampling for Reducing Approximation Variance

Currently I have only discussed Monte Carlo integration by sampling points x_i to solve the integral using a uniform distribution. However the purpose of introducing Equation 2.3 was to create a custom *pdf* which can be used for importance sampling to reduce the variance of the Monte Carlo integration approximation. To understand how and why importance sampling works, first observe Figure 2.1 where a constant function is given with a single sample point evaluated for $f(x)$. This single sample is enough to find the true value for the area beneath the curve i.e. integrate the function with respect to x. However, Figure 2.2 requires many more samples to accurately calculate the area beneath the curve, as with only a few evaluated sample points the variance of the estimated area beneath the curve will be high. Therefore, it requires fewer samples to approximate a function which is closer to being constant for all possible input values.

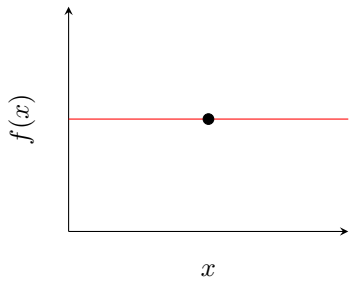

Figure 2.1: Constant Function with a sample point

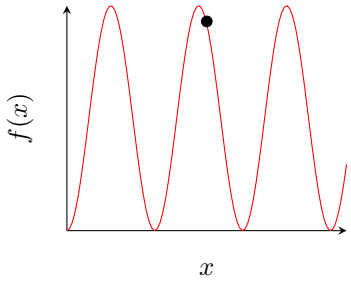

Figure 2.2: Non-linear Function with a sample point

Now, most functions are not constant however it is possible to turn a function into a constant function, and this is exactly what can be done within Monte Carlo integration. To convert a function f to a constant function, a function f' can be introduced which produces the same output as f for every input, but scaled by a constant c. The function f is then divided by f' to produce a constant function, as shown in Equation 2.7.

$$\frac{f(x)}{f'(x)} = \frac{1}{c} \tag{2.7}$$

This can be applied to Monte Carlo integration stated in Equation 2.3, by choosing a probability density function (*pdf*) which produces the same output as f for all inputs, but divided by some normlazing constant factor c, keeping *pdf* as a probability distribution. Therefore, we are able to calculate the true value of the integral thought Monte Carlo integration as shown in Equation 2.8. Where $\frac{1}{c}$ is true value for the integral in Equation 2.1.

$$\langle F^N \rangle = \frac{1}{N} \sum_{i=0}^{N-1} \frac{f(x)}{pdf(x)} = \frac{1}{N} \sum_{i=0}^{N-1} \frac{f(x)}{cf(x)} = \frac{1}{N} \sum_{i=0}^{N-1} \frac{1}{c} = \frac{1}{c} \tag{2.8}$$

For most cases it is not possible to know the correct probability distribution function which can convert the Monte Carlo integration problem into integrating a constant function. However, if one has prior knowledge regarding 'important' regions of the functions input space, it is possible to create a probability density function whose shape matches f more closely then a uniform probability distribution. By Important areas of the function input space, I mean areas where when given an input produce a large contribution to the function integral. For example in Figure 2.3, the most important regions are around the top of the functions peak.

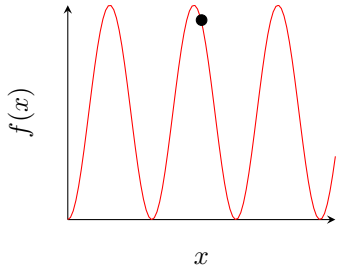

Figure 2.3: A really Awesome Image

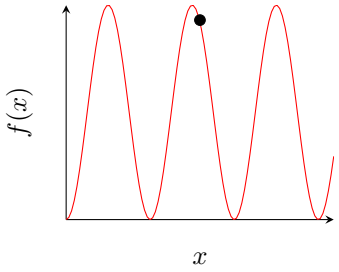

Figure 2.4: A really Awesome Image

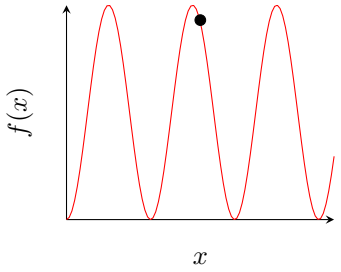

Figure 2.5: A really Awesome Image

As previously explained, Figure2.3 represents a probability density function which has a similar shape to the function which is being integrated. Therefore the variance in the Monte Carlo integration approximation will be lower then that of the uniform distribution shown in Figure 2.4. Figure 2.5 presents an example where the created probability density function does not resemble the shape of the function which is being integrated. Using this *pdf* in Monte Carlo integration in this case will actually increase the variance in the output compared to that from a uniform *pdf* shown in Figure 2.4. This is due to regions which have high importance according to the *pdf* are actually less important regions for the function f, meaning areas of the input space receive incorrect contribution weighting, causing the variance in the Monte Carlo integration approximation to rise.

2.2 Markov Decision Processes and TD-Learning

2.3 Linking TD-Learning and Light Transport Simulation

2.4 The expected SARSA Path tracer

2.4.1 Plan

Breakdown

- The rendering equation and what each component is, how this relates to global illumination
- Path tracings use of the rendering equation. How monte carlo comes into play. The iterative version of the path tracing algorithm. Concept of a light path.
- Importance sampling in terms of BRDF and relating this to reducing variance in pixel colours leading to less noise via a reduction in variance, but incorrect importance sampling can do the opposite. Give examples of classical importance sampling techniques and their performance. Critic them and clearly present where their shortcomings are and how they are unavoidable.
- Introduce reinforcement learning: Markov Decision Process, Bellman Equation, Temporal Difference Learning and its strong points and weaknesses, how does it differ to traditional monte-carlo (might not be relevant). Proved to converge on the true valuation function for a given state-action pair when run infinitely

- Give Nvidia's derivation of their learning rule. How does the Markov Decision Process relate to a rendered scene, i.e. what is the AI doing for us here. Provide a justification of parameter matching. Essentially cover all reinforcement learning theory of the paper here, with a justification (mathematical) and visual examples of why it works.
- Discretizing the state space is required for Q-learning to be applied, shortcoming is that it may not work very well with infinite state spaces. Introduce the Irradiance volume and how it can be used to rather store actual irradiance values to instead store Q-values. The irradiance distribution for a given point in the scene. Sampling the irradiance volumes around the scene onto geometry.
- Present the full algorithm proposed by Nvidia, displaying irradiance volumes learned Q-values (as an image of hemispheres) throughout the process and stating how these update a cumulative distribution to sample from.
- Introduce concept of Deep Q-learning and how it no longer needs a discretized state space. However it still requires action space to be discretized (unlike an actor-critic setup). What is the role of the network and what other function approximators can be trialled. Explain in quite some detail the DeepMind Atari paper which introduced Deep Reinforcement learning.

Preliminary

1. Define what a ray-tracing rendering algorithm consists of and the difference between global and direct illumination. Acknowledge other ray-tracing algorithm l ike bi-directional path-tracers, Rendermans algorithm, photon mapping.
2. Define terms like BRDF, radiance, irradiance and the rendering equation
3. Explain the details of the path-tracing algorithm in depth. It should be completely clear the relation between path-tracing and the rendering equation. It should be clear where the Monte Carlo approach comes in and why importance sampling within path-tracing can yield less noisy and more accurate results, potentially in the same fixed time-budget
4. Introduce the concept of importance sampling in computing global illumination with some early examples of its success, use in industry and recent papers on efficient light transport simulation. State the reasoning behind why it still continues to accurately simulate global illumination, in other words, why zero-contribution light paths do not contribute to the image.
5. Introduce reinforcement learning: Markov Decision Process, Bellman Equation, Temporal Difference Learning and its strong points and weaknesses, how does it differ to traditional monte-carlo (might not be relevant). Proved to converge on the true valuation function for a given state-action pair when run infinitely
6. State the derived learning rule supplied by Ken Dahm and visualize the matching terms as well as a justification why each parameter matches. What is the value and the incentive, diminishing return for rewards far in the future etc
7. State new on-line algorithm proposed by Ken Dahm and details for discretizing the state and action space into the Irradiance Volume data-structure which was previously introduced
8. Introduce the concept of deep reinforcement learning, describing how DeepMind used the technique for playing Atari games. Given a state give me the state-action values for all actions possible in that state. Then how we can apply this to our scene to model the state space and continuous.

Chapter 3

Deep Q-Learning Path tracer

3.0.1 Plan

Breakdown

1. State learning rule for deep Q-learning and the difference from deep Q-learning to q-learning. Maybe some of the difficulties associated with deep q-learning versus q-learning, and some of the general advantages.

2. Derive the learning rule for deep q-learning network which I used, once again justifying terms throughout the derivation.

3. Explain concept of eta-greedy policy used. Explain exploration vs exploitation but we will talk about this more later

4. Describe how the current method is used for diffuse surfaces. Introduce the pseudo code for the new algorithm. Give a description of each stage and what it does. Relating back to properties such as bias rendering and pointing out assumption made by the path tracer.

5. Present and explain the network architecture. Explain in depth about how the state was modelled as a point relative to all vertices to give the network information about the position of the vertex relative to the rest of the world compared to passing in a single position. Relate this to Atari games, we get an image showing where we are relative to the world rather than just a single position in the world.

6. Present some results side by side against a default path tracer and Nvidia's reinforcement learning approach. Pointing out aspects of the image and reasoning for certain parts.

Chapter 4

Critical Evaluation

A topic-specific chapter, of roughly 15 **pages**

This chapter is intended to evaluate what you did. The content is highly topic-specific, but for many projects will have flavours of the following:

1. functional testing, including analysis and explanation of failure cases,
2. behavioural testing, often including analysis of any results that draw some form of conclusion wrt. the aims and objectives, and
3. evaluation of options and decisions within the project, and/or a comparison with alternatives.

This chapter often acts to differentiate project quality: even if the work completed is of a high technical quality, critical yet objective evaluation and comparison of the outcomes is crucial. In essence, the reader wants to learn something, so the worst examples amount to simple statements of fact (e.g., "graph X shows the result is Y"); the best examples are analytical and exploratory (e.g., "graph X shows the result is Y, which means Z; this contradicts [1], which may be because I use a different assumption"). As such, both positive *and* negative outcomes are valid *if* presented in a suitable manner.

4.0.1 Plan

Data to collect

- Build 4 different scenes:
  - Simple geometry, Indirectly illuminated scene: Here both reinforcement learning methods should perform excellently
  - Simple geometry, Directly illuminated scene: Here all methods should perform well
  - Complex geometry, Indirectly illuminated scene: Can both methods do this - will take a lot of training, deeper NN potentially
  - Complex geometry, Directly illuminated scene: Can both methods do this - will take a lot of training, deeper NN potentially
- Number zero-contribution light paths/ light paths that do not intersect with a with a light after n bounces therefore they become irrelevant for all methods with accumulated frames on the x-axis
- Variance in points around the room to train network in order to make training batches as varied as possible (this is a weird one, essentially assessing the fact that we do not need a replay buffer).
- eta-greedy constant for loss curve for training the network & decaying eta-greedy policy graph for the loss as well
- Visual representation of Q-values being higher in directions near light source: Map q-values to hemispheres in the scene and get a close up, clearly indicating its ability to sample in the correct direction

- 1 SPP, 16 SPP, 32 SPP, 64 SPP, 128 SPP, 256 SPP for all three methods on 4 different scenes to evaluate their effectiveness: Assessing accuracy of global illumination approximation
- Limitations: Number of angles which can accurately be learned by the network, accuracy needs to be compared with expected SARSA approach for a single radiance volume at a given point in the scene. Size of the scene which can be learnt accurately.

Preliminary

1. Exploration vs Exploitation for both techniques, exploration can yield to better results plus exploitation does not accurately simulate light, relate to the rendering equation and how light works in the physical world.
2. Show for about 4 different scenes the results for a n different numbers of samples; the images, average path length, number of light paths which actually contribute to the image which are sampled between all techniques. I will have to analyse which reduces the number of zero contribution paths the most, but also still assess if the image is photo-realistic.
3. Also analyse default Q-learnings ability on top of expected SARSA
4. Justify reasoning for choosing to analyse Q-Learning, Expected SARSA and DQN (because they have good results for other cases and TD learning fits the online learning procedure)
5. Assess the number of parameters required, configuration is important for these algorithms, if it is very difficult to get right, then the time spent configuring may not be worth it compared to actually rendering the image. E.g. default path-tracing there are not other parameters apart from the number of samples per pixel, expected SARSA requires the user to specify the memory which is allowed to be used by the program, this requires careful consideration, as well as the threshold the distribution cannot fall below, the deep Q-learning algorithm requires less config but potentially different neural network architectures should be investigated to further reduce the number of zero-contribution light paths.
6. Ease of implementation
7. Parallelisability of each algorithm, path-tracing is far easier to parallelise as it requires minimal memory accesses by the program to infer pixel values, as opposed to expected SARSA which requires many. Deep-q learning has more customizability in terms of parallelizing (needs more research)
8. Memory usage: Path-tracing is minimal, Expected SARSA is unbounded, Deep Q-Learning is bounded by the size of the neural network, but the memory it requires is still significant (needs more research)
9. DQN vs Expected Sarsa: Do not have to wait for an iteration to begin importance sampling on the newly learned Q values for a given point, neural network is continually trained and inferred from. Continuous state space vs discretized required for storage in expected SARSA.

Chapter 5

Conclusion

A compulsory chapter, of roughly 5 pages

The concluding chapter of a dissertation is often underutilised because it is too often left too close to the deadline: it is important to allocation enough attention. Ideally, the chapter will consist of three parts:

1. (Re)summarise the main contributions and achievements, in essence summing up the content.
2. Clearly state the current project status (e.g., "X is working, Y is not") and evaluate what has been achieved with respect to the initial aims and objectives (e.g., "I completed aim X outlined previously, the evidence for this is within Chapter Y"). There is no problem including aims which were not completed, but it is important to evaluate and/or justify why this is the case.
3. Outline any open problems or future plans. Rather than treat this only as an exercise in what you *could* have done given more time, try to focus on any unexplored options or interesting outcomes (e.g., "my experiment for X gave counter-intuitive results, this could be because Y and would form an interesting area for further study" or "users found feature Z of my software difficult to use, which is obvious in hindsight but not during at design stage; to resolve this, I could clearly apply the technique of Smith [7]").

5.0.1 Plan

1. Summarise contributions:
   (a) Implementing a path tracer from scratch to analyse in depth the difficulties and issues that come with Ken Dahm's algorithm. Including memory usage, parallelisation and parameter usage.
   (b) Analysis of different reinforcement learning approaches pitched together clearly on a variety of scenes
   (c) Analysis of neural networks ability to learn the irradiance distribution function
   (d) Online deep-reinforcement learning algorithms effectiveness of learning irradiance distribution function
2. If DQN does not work well provide some further analysis on potential other alternatives which could be used.
3. Future Work: Policy learning to model continuous action & state space
4. DDQN and other deep reinforcement learning strategies

Bibliography

[1] Steve Bako, Thijs Vogels, Brian McWilliams, Mark Meyer, Jan Novák, Alex Harvill, Pradeep Sen, Tony Derose, and Fabrice Rousselle. Kernel-predicting convolutional networks for denoising monte carlo renderings. *ACM Trans. Graph.*, 36(4):97–1, 2017.

[2] Bloomberg. Peak video game? top analyst sees industry slumping in 2019.

[3] Chakravarty R Alla Chaitanya, Anton S Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. Interactive reconstruction of monte carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics (TOG)*, 36(4):98, 2017.

[4] Per Christensen, Julian Fong, Jonathan Shade, Wayne Wooten, Brenden Schubert, Andrew Kensler, Stephen Friedman, Charlie Kilpatrick, Cliff Ramshaw, Marc Bannister, et al. Renderman: An advanced path-tracing architecture for movie rendering. *ACM Transactions on Graphics (TOG)*, 37(3):30, 2018.

[5] Per H Christensen, Wojciech Jarosz, et al. The path to path-traced movies. *Foundations and Trends® in Computer Graphics and Vision*, 10(2):103–175, 2016.

[6] Ken Dahm and Alexander Keller. Learning light transport the reinforced way. *arXiv preprint arXiv:1701.07403*, 2017.

[7] Iliyan Georgiev, Thiago Ize, Mike Farnsworth, Ramón Montoya-Vozmediano, Alan King, Brecht Van Lommel, Angel Jimenez, Oscar Anson, Shinji Ogaki, Eric Johnston, et al. Arnold: A brute-force production path tracer. *ACM Transactions on Graphics (TOG)*, 37(3):32, 2018.

[8] Henrik Wann Jensen. Global illumination using photon maps. In *Rendering Techniques 96*, pages 21–30. Springer, 1996.

[9] James T Kajiya. The rendering equation. In *ACM SIGGRAPH computer graphics*, volume 20, pages 143–150. ACM, 1986.

[10] Alexander Keller, Ken Dahm, and Nikolaus Binder. Path space filtering. In *Monte Carlo and Quasi-Monte Carlo Methods*, pages 423–436. Springer, 2016.

[11] Jaroslav Křivánek, Alexander Keller, Iliyan Georgiev, Anton S Kaplanyan, Marcos Fajardo, Mark Meyer, Jean-Daniel Nahmias, Ondrej Karlík, and Juan Canada. Recent advances in light transport simulation: some theory and a lot of practice. In *SIGGRAPH Courses*, pages 17–1, 2014.

[12] NVIDIA. *NVIDIA Turing Architecture Whitepaper*, 2018.

[13] Zhigeng Pan, Adrian David Cheok, Hongwei Yang, Jiejie Zhu, and Jiaoying Shi. Virtual reality and mixed reality for virtual learning environments. *Computers & graphics*, 30(1):20–28, 2006.

[14] Ravi Ramamoorthi, John Anderson, Mark Meyer, and Derek Nowrouzezahrai. A theory of monte carlo visibility sampling. *ACM Transactions on Graphics (TOG)*, 31(5):121, 2012.

[15] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press, 2011.

Appendix A

An Example Appendix

Content which is not central to, but may enhance the dissertation can be included in one or more appendices; examples include, but are not limited to

- lengthy mathematical proofs, numerical or graphical results which are summarised in the main body,
- sample or example calculations, and
- results of user studies or questionnaires.

Note that in line with most research conferences, the marking panel is not obliged to read such appendices.