UMBC

DISSERTATION APPROVAL SHEET

Title of Dissertation: Cache and Bandwidth Aware Real-time Subsurface Scattering

Name of Candidate: Tiantian Xie

Doctor of Philosophy, 2021

Graduate Program: Computer Science

Dissertation and Abstract Approved:

Marc Olano

Marc Olano

Associate Professor

Computer Science and Electrical Engineering

7/26/2021 | 2:04:33 PM EDT

NOTE: *The Approval Sheet with the original signature must accompany the thesis or dissertation. No terminal punctuation is to be used.

CURRICULUM VITAE

Name:	Tiantian Xie
Degree and date to be conferred:	Ph.D., 2021
Collegiate institutions attended:	University of Maryland, Baltimore County, Ph.D., 2015-2021 Sichuan University, Master's Degree, 2011-2014 Sichuan University, Bachelor's Degree, 2007-2011
Major:	Computer Science

Professional Publications:

Tiantian Xie, and Marc Olano (2021). Real-time Subsurface Control Variates: Temporally Stable Adaptive Sampling. Proceedings of the ACM on Computer Graphics and Interactive Techniques, 4(1), 1-18.

Tiantian Xie, Marc Olano, Brian Karis, and Krzysztof Narkowicz (2020). Real-time subsurface scattering with single pass variance-guided adaptive importance sampling. Proceedings of the ACM on Computer Graphics and Interactive Techniques, 3(1), 1-21.

Wei Wang, Tiantian Xie, Xin Liu, Yao Yao, and Ting Zhu (2019). ECT: Exploiting cross-technology transmission for reducing packet delivery delay in IoT networks. ACM Transactions on Sensor Networks (TOSN), 15(2), 1-28.

Zicheng Chi, Yao Yao, Tiantian Xie, Xin Liu, Zhichuan Huang, Wei Wang, and Ting Zhu (2018). EAR: Exploiting uncontrollable ambient RF signals in heterogeneous networks for gesture recognition. In Proceedings of the 16th ACM conference on embedded networked sensor systems (pp. 237-249).

Yao Yao, Yan Li, Xin Liu, Zicheng Chi, Wei Wang, Tiantian Xie, Ting Zhu (2018). Aegis: An interference-negligible RF sensing shield. In IEEE INFOCOM 2018-IEEE conference on computer communications (pp. 1718-1726). IEEE.

Wei Wang, Tiantian Xie, Xin Liu, and Ting Zhu (2018). ECT: Exploiting Cross-Technology Concurrent Transmission for Reducing Packet Delivery Delay in IoT Networks. In IEEE INFOCOM 2018-IEEE Conference on Computer Communications (pp. 369-377). IEEE.

Zicheng Chi, Zhichuan Huang, Yao Yao, Tiantian Xie, Hongyu Sun, and Ting Zhu (2017). EMF: Embedding multiple flows of information in existing traffic for concurrent communication among heterogeneous IoT devices. In IEEE INFOCOM 2017-IEEE conference on computer communications (pp. 1-9). IEEE.

Tiantian Xie, Zhichuan Huang, Zicheng Chi, and Ting Zhu (2017). Minimizing amortized cost of the on-demand irrigation system in smart farms. In Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks (pp. 43-46).

Zhichuan Huang, Tiantian Xie, Ting Zhu, Jianwu Wang, and Qingquan Zhang

(2016). Application-driven sensing data reconstruction and selection based on correlation mining and dynamic feedback. In 2016 IEEE International Conference on Big Data (Big Data) (pp. 1322-1327). IEEE.

Zicheng Chi, Yao Yao, Tiantian Xie, Zhichuan Huang, Michael Hammond, and Ting Zhu (2016). Harmony: Exploiting coarse-grained received signal strength from IoT devices for human activity recognition. In 2016 IEEE 24th International Conference on Network Protocols (ICNP) (pp. 1-10). IEEE.

Shengyang Li, Ping Yi, Zhichuan Huang, Tiantian Xie, and Ting Zhu (2016). Energy scheduling and allocation in electric vehicles energy internet. In 2016 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT) (pp. 1-5). IEEE.

Professional positions held:

Graduate Assistant, University of Maryland, Baltimore County, MD, 2015-2021.

Rendering Programmer Intern, Epic Games, NC, 2020.

Rendering Programmer Intern, Epic Games, NC, 2019.

ABSTRACT

Title of dissertation: CACHE AND BANDWIDTH AWARE
REAL-TIME SUBSURFACE SCATTERING

Tiantian Xie, Doctor of Philosophy, 2021

Dissertation directed by: Professor Marc Olano
Department of Computer Science and
Electrical Engineering

Photo-realistic subsurface scattering is a demanding feature in many real-time applications, especially in next-generation games and virtual productions where the uncanny valley needs to be addressed for real-time human skin rendering. Most importantly, it must be addressed in milliseconds or less without visible quality compromise. These quality and performance demands are prohibitively expensive when using Monte Carlo sampling for subsurface scattering. Moreover, real-time rendering is limited by hardware capability and GPU cache architectures. This dissertation explores novel algorithms for high-quality photo-realistic real-time subsurface scattering with cache incoherence and limited bandwidth.

To achieve this, a new generic taxonomy is proposed for heterogeneous real-time rendering to identify techniques that can improve bandwidth and cache utilization. A single pass, variance guided, and generic $O(1)$ real-time adaptive sampling technique is proposed to minimize bandwidth demands and improve cache utilization. This adaptive sampling pass works with different global temporal accumulation techniques (e.g., Temporal Anti-Aliasing and Deep Learning Super Sampling) to further improve quality. We propose a new technique, adaptive filtered importance sampling (AFIS), based on our single pass adaptive sampling technique and filtered importance sampling. A hybrid AFIS and the separable approximation technique allows the user to balance quality and performance. To deal with instability during dynamic lighting, a novel use of Control Variates (CV) in the sample domain instead of shading domain is proposed.

Our algorithm induces as little as one texture overhead to a real-time rendering engine, and has been battle tested in the Unreal Engine, a commercial game engine.

CACHE AND BANDWIDTH AWARE REAL-TIME SUBSURFACE SCATTERING

by

Tiantian Xie

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, Baltimore County in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2021

Advisory Committee:
Professor Marc Olano (Chair/Advisor)
Professor Adam W. Bargteil
Professor David Chapman
Dr. David J. Hill
Professor Matthias K. Gobbert

© Copyright by
Tiantian Xie
2021

Acknowledgments

I owe all my gratitude to all the people who made this dissertation possible and Epic Games for supporting this research.

First and foremost, I would like to thank my advisor Dr. Marc Olano at every stage of the dissertation for guiding me to explore challenging and invaluable research projects. He has always made himself available for either help in mathematical problems or advice of career development. It is a pleasure to work with and learn from such an extraordinary person.

I would like to acknowledge the help and support from all my committee members: Dr. Adam Bargteil, Dr. David Chapman, Dr. David Hill and Dr. Matthias Gobbert, and other faculty and staff members at UMBC.

I would also like to thank the rendering team at Epic Games for the internship opportunity to implement the subsurface scattering pass, which makes this dissertation possible. I owe special thanks to Brian Karis, Krzysztof Narkowicz, Patrick Kelly, Peter Sumanaseni, John Hable, Mike Seymour etc. for the career development, support and inspiration.

All my colleagues at VANGOGH lab have enriched my graduate life in many ways and deserve a special mention and many thanks including: Ari Rapkin Blenkhorn, Alex Dahl, Qingyuan Zheng, and Yuping Zhang.

In addition, I would like to acknowledge the love and help from my family members: my mom, Birong Zhao, my dad, Zhiqiang Xie, and my cat, YiBu.

To all my friends who have spent time with me and show support to me, I will

always appreciate it.

It's not possible to remember all. I apologize to those who I have unintentionally left out!

Last but not least, thank you all!

Table of Contents

List of Tables viii

List of Figures ix

List of Abbreviations xi

1 Introduction 1
- 1.1 Thesis Statement 3
- 1.2 Cache and Bandwidth Aware 3
- 1.3 Heterogeneous Real-time Rendering 4
- 1.4 Adaptive Sampling 5
- 1.5 Subsurface Scattering 7
- 1.6 Outline 8

2 Related Work and Background 9
- 2.1 Monte Carlo Integration 9
  - 2.1.1 Variance Reduction 10
    - 2.1.1.1 Importance Sampling 10
    - 2.1.1.2 Low-discrepancy Sequence 12
    - 2.1.1.3 Stratified Sampling 13
    - 2.1.1.4 Control Variates 14
    - 2.1.1.5 Temporal Reuse 17
  - 2.1.2 Adaptive Sampling 18
    - 2.1.2.1 Theory 19
    - 2.1.2.2 Bias 21
    - 2.1.2.3 Metrics 23
    - 2.1.2.4 Adaptive Rendering 25
- 2.2 Subsurface Scattering 26
  - 2.2.1 Real-time Model 27
    - 2.2.1.1 Dipole Profile 29
    - 2.2.1.2 Burley's Profile 30
  - 2.2.2 Real-time Acceleration Techniques 32

        - 2.2.2.1 Guassian Blur 32
        - 2.2.2.2 Pre-integration 34
        - 2.2.2.3 Separable Approximation 36

3 Heterogeneous Real-time Rendering 39
- 3.1 GPU Cache 40
    - 3.1.1 Simplified GPU Data Cache Architecture 41
    - 3.1.2 Cache Miss Type 43
    - 3.1.3 Source of Incoherence 44
- 3.2 Framework 45
    - 3.2.1 Heterogeneous Computing Demands 47
    - 3.2.2 Heterogeneous Sample Demands 48
    - 3.2.3 Heterogeneous Memory Demands 49
- 3.3 Formulation 51
- 3.4 Summary 54

4 Real-time Adaptive Sampling 55
- 4.1 Introduction 56
- 4.2 Basic Metrics 57
- 4.3 Temporal Anti-aliasing 59
- 4.4 Metrics within Temporal Accumulation 60
    - 4.4.1 Circle Scenario 62
    - 4.4.2 Disocclusion 63
- 4.5 Local Guiding Integration 65
    - 4.5.1 Global TAA 66
    - 4.5.2 Deep Learning Super Sampling (DLSS) 69
- 4.6 Discussion and Limitations 70

5 Subsurface Scattering 72
- 5.1 Efficient Sample Generation 74
    - 5.1.1 Sampling Function 74
    - 5.1.2 Sampling Sequence 78
- 5.2 Adaptive Filtered Importance Sampling 79
- 5.3 Advanced Design 82
    - 5.3.1 Unification of Scattering 82
    - 5.3.2 Importance-Guided Acceleration 86
- 5.4 Summary 91

6 Real-time Control Variates 93
- 6.1 Introduction 94
- 6.2 Motivating Example 97
- 6.3 Control Variates 98
- 6.4 Theory 100
    - 6.4.1 In-frame Standard Control Variable 101
    - 6.4.2 In-frame Constant Control Variable 102

6.5 Online Solution . . . 104
  6.5.1 Online Covariance . . . 106
  6.5.2 Exponential Moving Covariance (EMC) . . . 106
  6.5.3 Exponential Moving Covariance Matrix (EMCM) . . . 107
  6.5.4 Coefficient Boundary . . . 108
  6.5.5 Online Joint Estimation Algorithm . . . 109
6.6 Offline CV Coefficient Estimation . . . 112
6.7 Static Lighting . . . 113
6.8 Discussion and Limitation . . . 115

7 Implementation and Results 116
7.1 Implementation . . . 116
7.2 Static Scene . . . 117
  7.2.1 Quality Comparisons . . . 118
  7.2.2 Adaptive Sampling Quality . . . 119
  7.2.3 Equal Quality Comparison . . . 122
  7.2.4 Real-time Counterpart Comparison . . . 122
  7.2.5 Performance Breakdown . . . 123
  7.2.6 Effect of Sample Budget on Time and Quality . . . 124
7.3 Control Variates on Static Scene . . . 125
7.4 Dynamic Scene . . . 127
7.5 Cache Analysis . . . 131
  7.5.1 GPGPU-Sim . . . 131
  7.5.2 Adaptive vs. Fixed Sampling Simulation . . . 132

8 Conclusion 135
8.1 Future Works . . . 137
8.2 Contributions . . . 139

A Separable Approximation to Burley's Model 142
A.1 Fitting . . . 142
A.2 Validation . . . 145

B Control Variates 147
B.1 Exponential moving covariance . . . 147
  B.1.1 Variable-weight covariance . . . 147
  B.1.2 Exponential moving covariance . . . 150
  B.1.3 EWMA covariance estimator . . . 151
B.2 Covariance of Two Batch Means . . . 151
B.3 CV Coefficient and Residual Function Selection . . . 152
  B.3.1 Scenario Setup . . . 152
  B.3.2 Result . . . 154
B.4 Theoretical Foundation . . . 156
  B.4.1 Updating Function . . . 156
    B.4.1.1 Time-invariant Scenario . . . 157

B.4.1.2 Time-variant Scenario 158
B.4.2 CV Coefficient, Mean and Variance of CV Residual 159
B.4.2.1 Stable Lighting 160
B.4.2.2 Dynamic Lighting 160
B.4.3 In-frame Constant Control Variable 161
B.4.3.1 Stable Lighting 161
B.4.3.2 Dynamic Lighting 162
B.5 More Static Lighting Insights 163
B.6 Additional Images 163

C Common Material Parameters for Burley's Normalized Profile 166

List of Tables

5.1 The realization of different subsurface scattering models. 84

6.1 Selected Symbols 104

7.1 Adaptive vs fixed phase breakdown for Digital Mike in Fig. 5.1 (ms). 123
7.2 The pass time (ms) and PSNR for rendering the scattering pass under different sample configurations 124
7.3 The mean metrics across all frames and the max performance ratio for adaptive sampling (AS), adaptive sampling with Online CV coefficient (+OCV), and with constant CV coefficient (+CCV). 127

B.1 The ReMSE of five configurations and the groud truth Res_{GT} 154

C.1 The measured parameters and the corresponding diffuse mean free path ℓ_d when $g = 0$. 166

List of Figures

1.1 *MetaHuman* rendered with our algorithm in real-time 1
1.2 Subsurface scattering in real and digital world. 2

3.1 Simplified GPU data cache architecture for Turing and RDNA 41
3.2 Comparison between (a) homogeneous and (b) heterogeneous real-time rendering 45
3.3 Cache and bandwidth aware real-time rendering algorithm taxonomy 46

4.1 Adaptive sampling visualization 56
4.2 Sample count estimation 63
4.3 Sample count estimation considering disocclusion 64
4.4 The local guiding framework for real-time adaptive sampling 66
4.5 Per-frame local variance guided Screen-space Subsurface Scattering (SSS) sampling to reach a target final rendering quality 67
4.6 Quality of different global temporal accumulation methods (TAA and DLSS 2.0) under two different intensities 69

5.1 Subsurface rendering comparison from close to far at 1920 × 1080 on NVIDIA Quadro P400 73
5.2 Approximation error for $cdf^{-1}(\xi)$ 75
5.3 Analytic inverse vs. our approximation 76
5.4 Infinite head (©Lee Perry-Smith) rendered with (a) MIS and (b) our approximation 77
5.5 Online sampling sequence generation 78
5.6 Subsurface scattering with sampling resolution. 83
5.7 Direct/diffuse region (black) and direct+distant (white) (b,c) for scene (a) 86
5.8 Subsurface pass overview 87
5.9 Design overview of the new subsurface scattering pass to support importance-guided acceleration 88

6.1 Dynamic subsurface scene requires temporally stable adaptive sampling. 94

6.2 Temporal instability leads to sample count over-estimation with real-time adaptive sampling. 97
6.3 Illustration of our novel application of CV 102
6.4 Real-time adaptive sampling with control variates 109
6.5 Sample count m_t estimation with online and offline CV coefficients a_t^* estimation. 113
6.6 Online Control Variates on Static Scene in real-time rendering engine 114

7.1 Subsurface ground truth comparison (without transmission) 116
7.2 Varying PSNR and κ for the Buddha scene with $b_{\max} = 64$ spp 120
7.3 Varying PSNR and κ for the close skin patch with $b_{\max} = 64$ spp . . . 121
7.4 Quality comparison of high scattering marble material. Observable vertical banding artifacts in (a) Separable. 121
7.5 Equal time comparison: fixed vs. adaptive. 123
7.6 Adaptive sampling history and quality analysis for teaser in Fig. 7.7 with static lighting. 125
7.7 CV Coefficient and sample count for Teaser in a static scene. Dynamic lighting and fire transparency are disabled for subsurface scattering quality comparison. 127
7.8 Real-time CV performance and quality test at resolution 2560x1440 for the three test scenes 128
7.9 The cycle level performance for (a) the sampling pass under (b) fixed and adaptive sampling schema 131
7.10 Total L1 and L2 cache read demand for near field rendering 133
7.11 L1 and L2 cache read miss rate (%) for near field rendering 134

A.1 Diffuse profile fitting result with three different methods. 143
A.2 The fitting from *falloff* color to Albedo (A) and DMFP (ℓ) with S_Δ. . 144
A.3 Fitting details in matlab. 144
A.4 Diffuse profile fitting for the default Separable configuration in UE4. . 145
A.5 Subsurface test for the beam light scene 146

B.1 The CV coefficient and variance under different configuration with 16 samples per frame time. 154
B.2 The CV coefficient and variance with time independent in-frame constant control variable. 163
B.3 More static lighting temporal insight with Control Variates 164
B.4 Additional qualitative comparison between without subsurface scattering, Separable and Ours. 165

List of Abbreviations

ATAA	Adaptive temporal anti-aliasing
AFIS	Adaptive filtered importance sampling
BRDF	Bidirectional reflectance distribution function
BSSRDF	Bidirectional scattering-subsurface reflection distribution function
CCV	Constant Control Variates
CDF	Cumulative density function
DLSS	Deep Learning Super Sampling
ECDF	Empirical cumulative distribution function
EMA	Exponential moving average
EMV	Exponential moving variance
FPS	Frames per second
FXAA	Fast approximate anti-aliasing
GPGPU	General-purpose GPU
GPU	Graphics Processing Unit
IS	Importance sampling
LDS	Local data share
LTE	Light transport equation
LUT	Look-up table
MC	Monte Carlo
MIS	Multiple importance sampling
MSE	Mean square error
OCV	Online Control Variates
PDF	Probability density function
PBR	Physically-based rendering
PCSS	Percentage-closer soft shadow
PSNR	peak signal-to-noise ratio
RMS	Root mean square
SDF	Signed Distance Function
SIMD	Single instruction, multiple data
SM	Streaming multiprocessor
SVD	Singular value decomposition
SVGF	Spatiotemporal variance-guided filter
TAA	Temporal anti-aliasing
UE4	Unreal Engine 4
VR	Virtual Reality
VRS	Variable Rate Shading

Chapter 1: Introduction

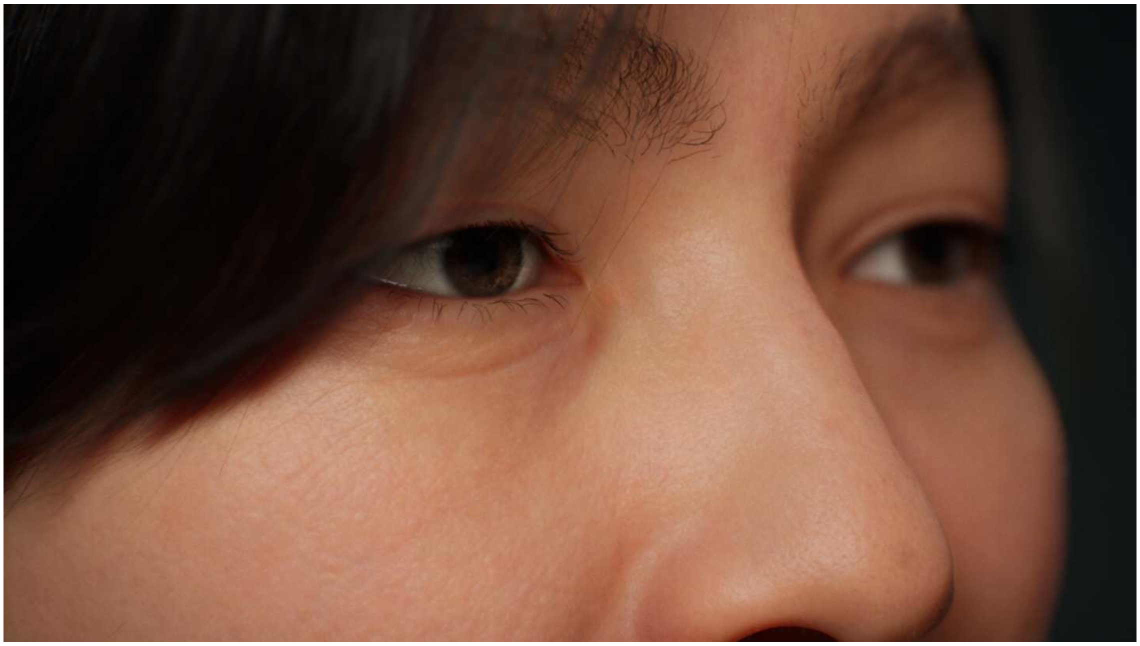
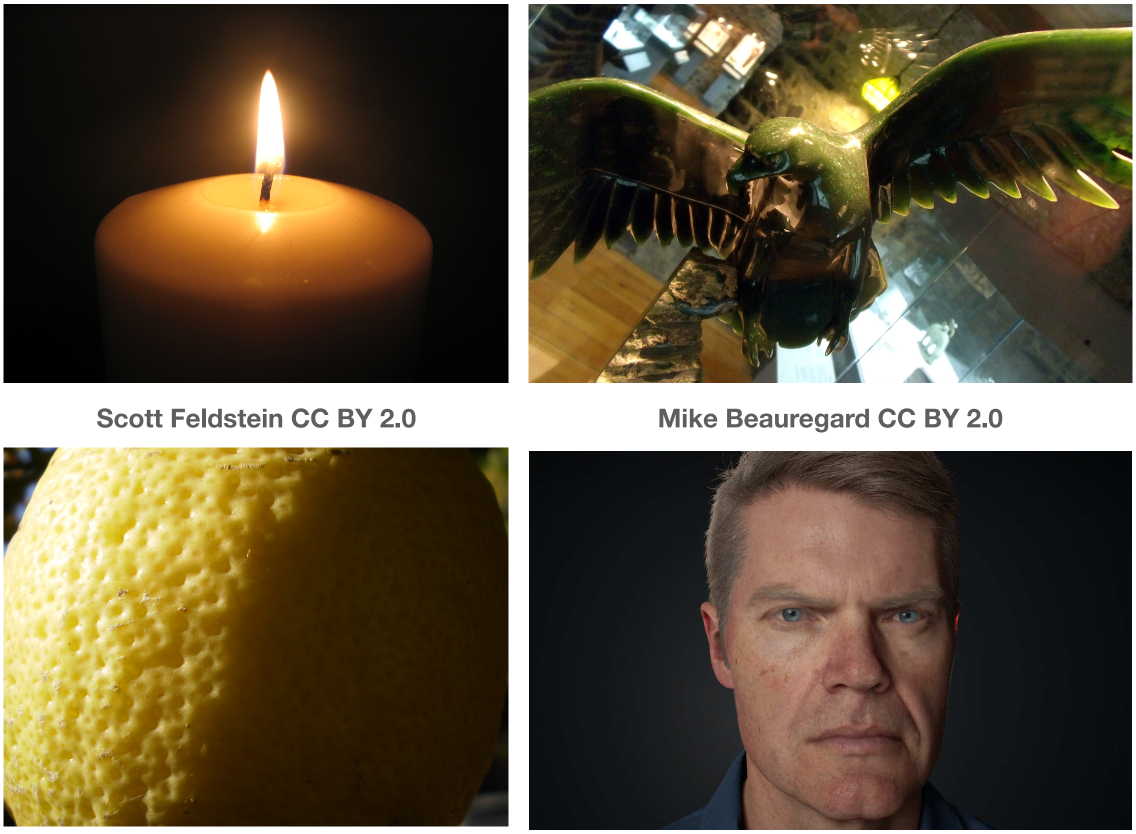


Figure 1.1: *MetaHuman* rendered with our proposed subsurface scattering algorithm in *Unreal Engine* in real time.

When light shines onto a surface, it bounces into the surface and bounces out some where else to create a soft look. This effect is called subsurface scattering. There are astonishing and enjoyable subsurface scattering effects in the real world for objects like candles, fruit, and jade (Fig. 1.2). But what we care most about in real-time rendering, especially in games, is photorealistic human skin rendering (shown in Fig. 1.1 and Fig. 1.2), without running into the uncanny valley, where artificial characters become horrible when they look close but not too close to human.



Scott Feldstein CC BY 2.0

Mike Beauregard CC BY 2.0

Paul Harrison CC BY 2.0

Digital Mike

Figure 1.2: Subsurface scattering in real and digital world.

However, photo-realistic rendering requires an expensive numerical method, Monte-Carlo sampling, to simulate the photon accumulation in the real world, where a large number of random photons are averaged for each pixel. We long for such a technique to render those characters with high perceived realism not only in game trailers, but in high frame rate gameplay. This constraint leaves only milliseconds or even sub-millisecond time for the whole subsurface scattering pass, no matter how complex the scattering is. Even an occasionally one-frame performance drop could ruin the immersive experience.

Going for photo-realistic for subsurface scattering will rely heavily on stochastic Monte Carlo sampling, a technique frequently used in offline rendering to create high quality images with large sample count. It is an even more challenging problem to do physically-based rendering in real time, since it has high incoherent cache access due to the cache architecture design, and requires high bandwidth demands due to the number of Monte Carlo samples within a frame.

What novel techniques can be use to target on high quality photo-realistic real-time rendering with the contemporary and next-generation hardware? This is the major research question this dissertation tries to explore to provide novel solutions.

1.1 Thesis Statement

Photo-realistic subsurface scattering based on Monte-Carlo sampling is expensive in real-time rendering because of incoherent cache accesses in contemporary GPU cache architecture and the high bandwidth demands within a frame. Mathematically sound adaptive sampling acceleration techniques can minimize sample counts and memory demands. To further reduce the computing demands for scalability, a hybrid combination of adaptive sampling and the separable approximation can achieve high frame-rate subsurface scattering with high quality.

1.2 Cache and Bandwidth Aware

In Monte Carlo sampling, each sample access might evaluate complex materials required for rendering. Moreover, the access does not always follow the assumption of good temporal and spatial coherence that current GPU architectures have. For example, to have a more physically correct subsurface scattering in screen space, not only do we need to sample the irradiance texture stochastically, which goes against the current architecture design, but we also need to access utility textures (e.g., pixel subsurface profile and/or normal) in the same stochastically way to resolve how different subsurface materials should be blended per pixel. This further exaggerates the cache incoherence and bandwidth demands. This is not unique to subsurface scattering. The ray-tracing enabled Turing architecture [NVIDIA, 2018] enables more physically correct rendering at the cost of cache-incoherent scene context access. The hardware acceleration of intersection reduces geometry accesses and computing time during intersection tests, but the complexity of materials used to simulate light and matter interaction prevents real-time rendering with stochastic sampling. This is a result of incoherent cache demands and limited bandwidth. This dissertation tries to propose cache and bandwidth aware methods at the algorithm level to increase performance.

1.3 Heterogeneous Real-time Rendering

There are many rendering techniques to accelerate real-time rendering. However, there is no such a taxonomy that we can look for to help the design of algorithms to minimize cache and bandwidth demands in algorithm level. Therefore, we created such a categorization based on homogeneity and heterogeneity of real-time rendering demands in three categories: i) Computing demands, ii) Sample demands, and ii) Memory demands. With this taxonomy, we can use mathematical formula to describe the demands and the cost of general algorithms that accelerates real-time rendering. In a brief summary, we have

1. Computing demands. The computing unit cost can be decomposed into the number of units, the type of units (e.g., raterizer and raytracing unit), the acceleration technique type (e.g., Shader levels of detail), and spatial reuse factors (due to variable/adaptive rate shading).
2. Sample demands. Samples are used to increase the quality. The sample demand can be reduced using methods that are both homogeneous across pixels (e.g., importance sampling, and temporal accumulation) and heterogeneous (e.g., adaptive sampling) across pixels.
3. Memory demands. The memory access cost can be reduced through procedural techniques, compression, mipmaps, virtual textures and geometry level of details.

Within the taxonomy, we have designed algorithms to make subsurface scattering efficient in real time. Note that this taxonomy is designed from the cache and bandwidth perspective. Please refer to Chapter 3 for more details.

1.4 Adaptive Sampling

Adaptive sampling is an efficient method to increase the rendering efficiency — less rendering time (i.e., less sample count) is required given a target variance. The algorithm is based on a statistical formulation of the relationship of sample count and variance (namely, the variance halves when the sample count doubles). Its efficiency is brought in by an early termination of several pilot samples to survey the variance. However, if the pilot sample count is too small, the variance is estimated inaccurate. When the pilot sample size is too large, we lose the efficiency. We address this issue for real-time rendering where we use an approximation function using sample histories, yet this history and the corresponding computing complexity is $O(1)$. This makes adaptive sampling efficient for real-time rendering.

Our algorithm uses history to improve the efficiency yet does not constrain the actual implementation of the global temporal accumulation algorithm. We demonstrate that it works with a variation of the standard Temporal Anti-Aliasing (TAA) algorithm. Although we do not contribute a new deep learning algorithm for this global accumulation in computer graphics, we demonstrate that our algorithm also works with a pre-existing general deep learning technique, Deep Learning Super Sampling (DLSS) designed for super sampling, as the global temporal accumulation algorithm.

Since the variance estimation is based on a temporal survey of the context to estimate spatial variance, the temporal variance could lead to over-estimation. In this dissertation, control variates are applied to remove as much of temporal variance to achieve temporally stable adaptive sampling. To achieve online estimation, we propose to use a simple yet mathematically founded novel technique for online covariance estimation, exponential moving covariance matrix.

1.5 Subsurface Scattering

In order to achieve the full potential of physically-based rendering of subsurface scattering, we pursue Monte Carlo sampling with our proposed adaptive sampling technique based on Burley's normalized diffuse reflectance profile [Christensen and Burley, 2015]. To make it efficient, several more techniques have been utilized and extended, including importance sampling, and both stratified sampling and low-discrepancy sequence.

For importance sampling, we first simplify the radius importance sampling function for efficient sample generation. Next, we apply filtered importance sampling [Křivánek and Colbert, 2008] to increase the sample information per sample and reduce the bandwidth demand. The filtered importance sampling algorithm has also been fused with our adaptive sampling algorithm as adaptive filtered importance sampling (AFIS). It further reduces the rendering time as subsurface scattering has heterogeneous sample demands, which makes the algorithm run in sub-linear time.

Since subsurface scattering can be decomposed into direct scattering (diffuse) within a pixel, and distant scattering of more than a pixel, we can separate each region and use stratified sampling to combine the final value. We have applied stratified sampling in the 2D sampling sequence domain for the decomposition. Since the direct scattering is assumed to access a flat pixel value, we only monitor the variance of the distant scattering for adaptive sampling.

Since adaptive sampling is only effective when not all scenes are complex, we also seek to minimize computing demands by using multiple acceleration techniques for a pass, subsurface scattering. Specifically, we proposed a framework to support the dynamic switching of techniques between separable subsurface scattering and our method based on AFIS on the fly. This gives the possibility of high resolution gaming, where main characters can explore the full physically-based rendering quality, while others with separable for performance and quality balance.

1.6 Outline

Chapter 2 introduces the background and related work for Monte Carlo integration and real-time subsurface scattering.

Chapter 3 provides a taxonomy of heterogeneous real-time rendering algorithms in terms of cache and bandwidth demands.

Chapter 4 introduces a real-time adaptive sampling algorithm for Monte Carlo sampling.

Chapter 5 shows the subsurface scattering framework and other advanced designs.

Chapter 6 introduces an online algorithm based on control variates to perform temporally stable adaptive sampling.

Chapter 7 provides the details of implementation and results of subsurface scattering.

Chapter 8 presents conclusions and future research directions.

Chapter 2: Related Work and Background

2.1 Monte Carlo Integration

The integration of functions in photo-realistic rendering (e.g., subsurface scattering, ambient occlusion, soft shadow, glossy reflection, etc.) often do not have an analytic solution. For a given function $f(x)$ to integrate as $F = \int f(x)dx$, we can use Monte-Carlo integration methods to solve it as

$$F_N = E[f(x)]_N = \frac{1}{N}\sum_{i=1}^{N} f(X_i) \tag{2.1}$$

where random variables X_i are uniformly distributed. As long as the function can be evaluated at X_i, the integration can be solved. However, the Monte Carlo estimator brings in error with a reduction rate of $O(\sqrt{n})$. Namely, it requires four times as many samples to reduce the error by half. One of the major research problems in rendering is to minimize the variance with the minimal number of samples, especially for real-time rendering where the bandwidth and computing power are limited given a millisecond-level time budget on different hardware platforms. To make full use of the single instruction, multiple data (SIMD) GPU architecture, it leads to an even lower equal sample count per shading unit for less capable devices.

Therefore, variance reduction techniques with low cost are critically needed, even if they introduce some acceptable bias.

2.1.1 Variance Reduction

In this section, we review all the variance reduction techniques that our algorithm will utilize to make a stochastic sampling algorithm performant for high quality real-time rendering.

2.1.1.1 Importance Sampling

One of the most prominent methods in Monte Carlo rendering for variance reduction is importance sampling (IS), introduced first to graphics in bidirectional path tracing [Lafortune and Willems, 1993, Veach and Guibas, 1995]. The basic idea is that the expectation of a function with respect to one distribution can be approximated by a weighted draw from another distribution [Tokdar and Kass, 2010] as

$$
\begin{aligned}
E_p[f(x)] &= \int f(x)p(x)dx && (2.2)\\
&= \int f(x)\frac{p(x)}{q(x)}q(x)dx && (2.3)\\
&= E_q\left[\frac{f(x)p(x)}{q(x)}\right] && (2.4)\\
&= E_q[f(x)w(x)] && (2.5)
\end{aligned}
$$

where $w(x) = \frac{p(x)}{q(x)}$ and $p(x)$ is a probability density function, $q(x)$ is another probability density function. $E_q[\cdot]$ denotes that the expectation is with respect to $q(x)$ where $x \sim q$ and for $f(x)q(x) \neq 0$, $q(x) > 0$. Then, we have the variance for the correlated sampling of function f and w based on x as

$$\begin{aligned}
Var(f \cdot w) =& E[f^2 w^2] - E[fw]^2 && (2.6)\\
=& Cov(f^2, w^2) + (Var(f) + E[f]^2)(Var(w) + E[w]^2) && (2.7)\\
&- (Cov(f, w) + E[f]E[w])^2. && (2.8)
\end{aligned}$$

This variance cannot be further reduced as we cannot have the independent assumption between f and w. However, if $f(x) = c$ is known as a constant, we have

$$\begin{aligned}
Var(f \cdot w) &= 0 + c^2(Var(w) + E[w]^2) - c^2 E[w]^2 && (2.9)\\
&= c^2 Var(w). && (2.10)
\end{aligned}$$

Under such circumstance, the smaller the variance of $w(x)$ is with respect to q, the smaller the total variance is, which means the closer $q(x)$ approximates to $p(x)$ over the domain x, the more efficient the importance sampling technique is.

In rendering, especially real-time rendering, the target density function $p(x)$ is often known or the CDF is known. For the first case, we can assign $q(x) = p(x)$ to make full use of IS. If $q(x)$ is too complex to sample in real-time, a lightweight approximation function $\tilde{q}(x)$ can be fitted beforehand. However for the later case, when an analytic inverse solution might be impossible to derive, a triangle-cut solution [Heitz, 2020] to inverse the CDF could be applied, or we can approximate again.

Another solution would be using multiple importance sampling [Grittmann et al., 2019] where multiple density functions are employed together to approximate $p(x)$. However, sometimes there is no analytic density function, like the lighting distribution function. Those functions can be estimated from the data and described by a representation of the data, like structured importance sampling [Agarwal et al., 2003], then we can sample the representation based on a random variable.

In real-time rendering, even when we have the analytic function to describe $q(x)$. It is not an easy task to be performant as it requires stochastic access of the data, which leads to a *cache incoherence crisis* where frequent loading from off-chip memory is inevitable.

2.1.1.2 Low-discrepancy Sequence

To efficiently generate samples for MC integration, the sampling space needs to be uniformly sampled. Generally, we have

$$E[f(x)] = \underbrace{\int_{\mathbf{u}\in\mathcal{D}} \frac{f(Q^{-1}(\mathbf{u}))p(Q^{-1}(\mathbf{u}))}{q(Q^{-1}(\mathbf{u}))} d\mathbf{u}}_{\text{uniform sampling}} \tag{2.11}$$

with the integral warping $x = Q^{-1}(\mathbf{u})$, where $\mathbf{u}$ is in uniform space and $\mathcal{D} = [0, 1)^n$. $Q(x) = \int_x q(t)dt$ is the cumulative density function. The PDF $q(Q^{-1}(\mathbf{u}))$ is the change-of-variable Jacobian determinant.

To uniformly sample $\mathbf{u}$, different low-discrepancy sequences (e.g., Sobol, Halton, Hammersley, n-Rooks or R_2) are preferred rather than a naive random sequence to minimize the perceived errors in real-time rendering. The discrepancy of a sequence [Pharr et al., 2016] can be measured as

$$D_N(B, P) = \sup_{b \in B, B \subseteq \mathcal{D}} \left| \frac{\#\{\mathbf{u}_i \in b\}}{N} - V(b) \right| \tag{2.12}$$

where a sequence of N samples $\mathbf{u}_1, ..., \mathbf{u}_N$ are drawn according to a sequence generator within b, $B = \{[0, v_1] \times ... \times [0, v_i] \times ... \times [0, v_n]\}$ and $v_i \in [0, 1)$. $V(b)$ measures the volume of interest. $\#\{\cdot\}$ measures the number of samples.

For real-time rendering pipelines that have some capability of temporal accumulation, an optimal low-discrepancy sequence would be a low-discrepancy n dimension sequence which also has low-discrepancy in projected $(n-1)$ dimension space for Monte-Carlo integration within a frame. To the best of our knowledge, it is still an open problem in the literature. In this dissertation, we do not explore to improve the sampling sequence for real-time rendering. However, we will use real-time friendly low-discrepancy sequence, R_2 sequence for sample generation. To mitigate the correlation between neighbor pixels, a real-time efficient hash function [Jarzynski and Olano, 2020] is applied to generate random samples, e.g., based on the frame number and the pixel coordinate.

2.1.1.3 Stratified Sampling

Stratified sampling is another useful technique to reduce variance, where the domain are subdivided into strata. Each stratum is assured to allocate samples with different sampling strategies. Then the variance can be minimized for the modified MC integral

$$E[f(x)] = \sum_{i=1}^{M} \int_{\mathbf{u} \in \mathcal{D}_i} \frac{f(Q^{-1}(\mathbf{u}))p(Q^{-1}(\mathbf{u}))}{q(Q^{-1}(\mathbf{u}))} d\mathbf{u} \tag{2.13}$$

where $\mathcal{D} = \sqcup_i^M \mathcal{D}_i$ and each $\mathcal{D}_i$ is a *stratum*. This strata boundary subdivision is often constant as a grid in computer graphics [Pharr et al., 2016]. However it can also be allocated based on some probability density function where $P_i = \frac{|\mathcal{D}_i|}{|\mathcal{D}|}$. $|\cdot|$ measures the volume. Then, N sample MC estimator is

$$F_N = \sum_{i=1}^{M} P_i \left[\frac{1}{N_i} \sum_{j=1}^{N_i} \frac{f(Q^{-1}(\mathbf{u}_j))p(Q^{-1}(\mathbf{u}_j))}{q(Q^{-1}(\mathbf{u}_j))} \right], \mathbf{u}_j \in \mathcal{D}_i \tag{2.14}$$

where $N_i = P_i \cdot N$. The stratification ensures that important features have high changes of being sampling overall. In addition, stratification increases sampling efficiency if some strata integrals are known to be of low frequency where low sample count (e.g., one sample) yields the exact result. This actually leads to our design of more efficient real-time adaptive sampling with the unification of scattering in section 5.3.1.

2.1.1.4 Control Variates

The control variates method is a variance reduction technique that decomposes a Monte-Carlo integration into the sum of a known integral with some scaling coefficient and a more easily estimated residual [Ripley, 2009]. It has been studied and applied in many fields, like finance [Alexander, 1999], operations research [Nelson, 1990, Hesterberg and Nelson, 1998], and computer networks [Lavenberg et al., 1982]. It requires an estimation of the optimal CV coefficient.

For a function to integrate $F = \int_{\mathcal{D}} f(x)dx$, we assume that there is a function $g(x)$, the control variate function, that the integration of the given domain $\mathcal{D}$ is already known as G. Then the function can be rewritten as

$$F = a \cdot G + \int_{\mathcal{D}} f(x) - a \cdot g(x)dx \tag{2.15}$$

where a is a parameter to scale function $g(x)$ so that the signal $f(x) - a \cdot g(x)$ is as flat as possible. During sampling, $f(x)$ and $g(x)$ are sampled with the same random number as coupled sampling. In this way, the difference given x will be relatively small due to the flatness of the signal. The variance will only be introduced by slopes inside the signal. Then, the CV estimator

$$\langle F \rangle = a \cdot G + \langle F - a \cdot G \rangle. \tag{2.16}$$

$\langle \cdot \rangle$ denotes the MC sampling expectation for simplicity. It should demonstrate low variance as

$$Var(\langle F\rangle) = Var(a \cdot G + \langle F - a \cdot G\rangle) \tag{2.17}$$

$$= Var(\langle F\rangle) + a^2\, Var(\langle G\rangle) - 2a\, Cov(\langle F\rangle, \langle G\rangle), \tag{2.18}$$

where when $a = Cov(\langle F\rangle, \langle G\rangle)/\,Var(\langle F\rangle)$ (by setting the derivative of Eq. 2.18 to zero to solve for a), the variance is minimized as

$$Var(\langle F\rangle) = (1 - Corr(\langle F\rangle, \langle G\rangle)^2)\, Var(\langle F\rangle), \tag{2.19}$$

where $Corr(\cdot)$ is the correlation coefficient. Therefore, as long as the selected control variate function $g(x)$ is correlated with $f(x)$, we can achieve variance reduction. If $g(x)$ approximates to $f(x)$ so close that $Corr(\langle F\rangle, \langle G\rangle)$ becomes 1 or -1, the variance is reduced to zero. Even when the correlation is zero, the measured variance will be exactly the same as $Var(\langle F\rangle)$. This characteristics gives us a very good motivation to use CV when variance is used in an estimation process.

In computer graphics, CV has been explored in offline rendering in many applications with a pre-defined constant CV coefficient to separate out the main part (e.g., residual ratio tracking [Novák et al., 2014], and multiple correlated sampling [Szécsi et al., 2004]). It has also been explored to find the optimal CV coefficient through penalized least squares [Fan et al., 2006], regressions [Rousselle et al., 2016, Kondapaneni et al., 2019], and deep learning [Müller et al., 2020]. However, no existing research provides any guidance or applications for real-time rendering where the time budget per frame is only several milliseconds and we cannot afford those expensive calculations. Moreover, the goal here is slightly different — to reduce the monitored variance due to temporal changes instead of to reduce the MC estimator variance as those prior techniques do in static scenes. The monitored variance is then used for sample count estimation in real-time adaptive sampling. In this dissertation, we propose such an online CV coefficient estimation algorithm based on a novel online covariance matrix formula in the context of adaptive sampling. In the traditional usage, $f - a \cdot g$ is expected to be a constant. However, this finding used in graphics does not lead to the optimal result. It is expected to be zero when temporal information is considered instead. We also provide a constant lightweight approximation in the spirit of *major part separation* for easy adoption. For more details, please refer to Section 6.

2.1.1.5 Temporal Reuse

Real-time rendering demonstrates high spatio-temporal coherence [Scherzer et al., 2012]. This coherence has been used for temporal anti-aliasing [Karis, 2014] (TAA) and accumulation with shading or geometry information to amortize cost and improve the rendering quality. The major problems dealt with in the literature are blurring, ghosting, lag and flickering [Xiao et al., 2018, Patney et al., 2016, Iglesias-Guitian et al., 2016] due to the use of exponential moving average that blends history with the current frame, and heuristics to tell if they are from the same object.

Temporal information has also been integrated with other systems to amortize sample costs. Recent work replaces the TAA result with ray tracing when ghosting and blurring are detected based on variance and depth [Marrs et al., 2018]. The *spatio-temporal variance-guided filtering* (SVGF) method includes geometry information and an image-space wavelet filter to achieve one sample per pixel real-time path tracing [Schied et al., 2017]. Schied et al. [2018] adaptively change the exponential weights based on temporal gradients to improve the scene temporal stability (reducing temporal overblurring) caused by fast light changes when using SVGF.

In this work, we guide the stochastic sampling process through temporally-accumulated variance and effective sample count. This allows us to amortize the sample count to meet quality requirements and time budget per pixel. We do this without modifying the existing TAA process or constraining its implementation, providing more stable and lower variance input to the TAA process, while still relying on that TAA to combine samples across frames. Unlike existing work [Iglesias-Guitian et al., 2016] where linear models and inverse covariance matrix are fit through recursive least squares to provide variance information, we use a light-weight algorithm that needs only one additional texture to be passed between frames.

2.1.2 Adaptive Sampling

Adaptive sampling is a sampling technique to allocate more samples in regions with high variation [Pharr et al., 2016]. It enables faster convergence than uniform sampling [Dammertz et al., 2010, Moon et al., 2014]. However, the major drawback is if the pilot samples are also used to estimate the mean, the result would be biased [Kirk and Arvo, 1991]. This is also the dilemma that prevents the adoption of adaptive sampling in real-time rendering — most real-time renderers do not want to waste sample count on unrendered pilot samples. In this section, we review some adaptive sampling algorithms, the source of bias and several applications of adaptive sampling.

We are especially interested in those adaptive sampling algorithms potential for real-time rendering. Namely, the algorithm should not increase the sampling efficiency at the cost of the overhead of the adaptive sampling algorithm. Of those algorithms, we find the statistical variance method mathematically sound and efficient for real-time rendering. Therefore, we give a full introduction of the theory and potential bias. At last, we have a brief introduction to other potential metrics to indicate future potential research directions.

2.1.2.1 Theory

A statistically optimized adaptive sampling has been proposed in the literature [Lee et al., 1985]. If we integrate a function $f(x)$ with a weight function $p(x)$ over the domain $\mathcal{D} = R^n$, the expected value is

$$E_p[f(x)] = \int_{\mathcal{D}} f(x)p(x)dx, \int_{\mathcal{D}} p(x)dx = 1. \tag{2.20}$$

When $x_1, x_2, ..., x_N$ are independent identically distributed samples according to $p(x)$, the Monte Carlo estimator has

$$F_N = \frac{1}{N} \sum_{i=1}^{N} f(x_i). \tag{2.21}$$

With the strong law of large number, $\lim_{N \to \infty} F_N = E[f(x)] = E[F_N]$, and

$$\begin{aligned} Var(F_N) &= E[(F_N - E[f(x)])^2] && (2.22) \\ &= \frac{E[f^2(x)] - E[f(x)]^2}{N} && (2.23) \\ &= \frac{Var(f(x))}{N}. && (2.24) \end{aligned}$$

With the increasing number of N, the Monte Carlo variance is reduced. The β probability that N is large enough to reduce the variance below a given threshold under the normal theory is

$$P(\frac{N s_N^2}{Var(f(x))} < \chi_\beta^2(N-1)) = \beta, \tag{2.25}$$

where $s_N^2 = \frac{1}{N} E[(f(x_i) - F_N)^2]$ is the sample variance of the data. Note that the Chi-Squared has the same form in normal theory as

$$\chi^2(k-1) = \frac{(k-1) \cdot s^2}{\sigma^2}, \tag{2.26}$$

and k is the number of degrees of freedom. With respect to the variance of the Monte Carlo estimator F_N, we have

$$P(Var(F_N) < \frac{s_N^2}{\chi_\beta^2(N-1)}) = 1 - \beta. \tag{2.27}$$

Then the variance threshold can be determined as

$$Var_0(F_N) = \frac{\sigma_w^2}{\chi_\beta^2(N_{\max}-1)}, \tag{2.28}$$

where σ_w^2 is the target maximal variance, and $N_{\max}$ is the maximum number of samples allowed due to computing budgets. Then, we can get the minimal sample count N based on the variance threshold.

Reinterpretation of Chi-Square. Under the normal theory assumption of Chi-Squared in Eq. 2.26, it gives us a new interpretation in the domain of adaptive sampling, which is the number of samples required to reach the variance σ^2 of a normal population, given the k observations with a standard derivation of the data s. Our proposed real-time adaptive sampling is extended from this reinterpretation in Eq. 4.1.

2.1.2.2 Bias

Adaptive sampling leads to premature stop when the initial samples cannot observe the high frequency information. This premature stop results in bias. To illustrate the bias from adaptive sampling statistically, a general form of adaptive sampling algorithm [Kirk and Arvo, 1991] is shown in Algorithm 1. This algorithm captures most cases in adaptive sampling. In the algorithm, a pilot sample set of size n is first drawn from a random variable X with a distribution (Line 2). If a variance metrics $\mathit{Variantion}(\cdot)$ on the sample set S_n is less than a variance threshold (Line 3), the sampling population will be regarded as easy. The neighboring values are of low frequency. The sampling result is a mean metric on the sample set (Line 4). Otherwise the population is a hard case where the neighbor values are of high frequency. Further oracle operations are performed to calculate the true mean (Line 6).

Algorithm 1 General Adaptive Sampling Algorithm

```
Require: $X, n, \epsilon$
1: // Draw $n$ i.i.d. random pilot samples from $X$
2:     $S_n = \{x_1, x_2, ..., x_n\}$
3: if $\mathit{Variation}(S_n) < \epsilon$ then
4:     $\mu = \bar{S_n}$
5: else
6:     $\mu = \mathit{Oricle}(X)$
7: end if
8: return $\mu$
```

Since a pixel in rendering is mostly determined by a finite domain (e.g., the auxiliary buffers around the target pixel), we can safely assume that the random variable X is a weighted sum of a measure of the auxiliary buffers. Equivalently, it is k distinct values $v_1, v_2, ..., v_k$ summed with the corresponding probabilities $p_1, p_2, ..., p_k$.

Then the ground truth mean can be denoted as

$$v = \sum_{i=1}^{k} p_i v_i, \tag{2.29}$$

where the weight is normalized by $\sum_{i=1}^{k} p_i = 1$. With respect to algorithm 1, we can rewrite the expectation based on conditional expectations in accordance to low, and high frequency as

$$E[\mu] = E[\mu|low] \cdot P[low] + E[\mu|high] \cdot P[high] \tag{2.30}$$

where $E[\mu|high] = v$ and $P(high) = 1 - P(low)$. Since any sample set S_n can be created by k-tuple $(n_1, n_2, ..., n_k)$ with $\sum_{i=1}^{k} n_i = n$, and the n_i samples are of value v_i, the probability of the k-tuple based on multinomial distribution has

$$P[n_1, n_2, ..., n_k] = \frac{n!}{n_1!...n_k!} \prod_{i=1}^{k} p_i. \tag{2.31}$$

In the low frequency case, if ϵ is sufficient small, all n samples can be the same value. It leads to

$$P[low] = \sum_{i=1}^{k} p_i^n \tag{2.32}$$

$$E[\mu|low] = \frac{\sum_{i=1}^{k} p_i^n v_i}{P[low]} \tag{2.33}$$

Then the adaptive sampling result function can be simplified to

$$E[\mu] = v + \sum_{i=1}^{k} p_i^n (v_i - v) \tag{2.34}$$

where the right part is the bias. The bias diminishes consistently when n increases. In order to minimize the bias, the initial pilot sample size n needs to be sufficient large. For applications, like subpixel anti-aliasing, several pilot samples might be good enough. But for complicated applications like percentage closer softer shadow, ambient occlusion, or subsurface scattering, they might require even larger n, which makes adaptive sampling inappropriate. In this dissertation, we try to tackle this issue for real-time applications. We keep the pilot size n small and adaptive, yet the algorithm unbiased with temporal reuse of sample count statistics, because our adaptive sampling process determines the sample count instead of the shading result for each frame (Section 4).

2.1.2.3 Metrics

This section presents the adaptive sampling variance metrics. Zwicker et al. [2015] reviews the lastest adaptive sampling and reconstruction algorithms in a systematic way by dividing the metrics type into two categories ‘a priori’ and ‘a posteriori’.

‘A priori’ Metrics. ‘a priori’ metrics rely on a local analysis of the light transportation equation. For example, it can be described by local geometry, BRDF models, depth [Soler et al., 2009], and local light field. After estimation, they can

often be directly used to determine the number of samples. More specifically, 1) we can perform frequency analysis in Fourier domain [Durand et al., 2005], then when the frequency information is combined with a single sample, it is able to determine the local sampling density based on Nyquist's sampling theorem and local bandlimits. The same idea also extends to the wavelet domain. Bagher et al. [2012] extract a bandwidth buffer to estimate sampling rate, but, at the cost of at least 16 spp before actual sampling. This spectrum can also be described through a covariance matrix for high dimension space [Belcour et al., 2013, 2014]. 2) Derivative analysis (e.g., 1st and 2nd order derivatives) can also be performed with the basic idea that the higher the gradient the more complex the surface is, the more samples are required. Ramamoorthi et al. [2007] uses the magnitude of first-order gradient to perform hierarchical sampling of uniform grids. The higher the gradient, the more samples are drawn for the next level.

'A Posteriori' Metrics. These methods rely on actual Monte-Carlo sampling of the light transportation equations to statistically determine if more samples are still needed for a pixel or region. Common features include: contrast [Mitchell, 1987], perceptual metrics [Bolin and Meyer, 1998], variance through MSE, mutual information, and chi-square. To determine the metric, spatial information is used. For example, MSE requires accessing neighbor pixels to estimate the variance. Bolin and Meyer [1998] proposed a hierarchical adaptive sampling technique in wavelet space. It iteratively places a sample at the location with largest perceptual error globally after a traversal of a quad tree that stores intermediate errors.

Some other algorithms make use of both 'a priori' and 'a posteriori' metrics.

For example, for inhomogeneous single-scattering media, Ren et al. [2008] used gradient based interpolation to reconstruct the source radiance inside the volume with a small set of samples. However, there exist sharp shading regions where denser sampling is required. To perform adaptive sampling, they proposed an error matric based on the product of the numerical gradient of radiance values at six surrounding point along the basis direction, and the actual radiance difference between neighbor sampled radiance within a valid sphere. Then, an recursive version of algorithm 1 (where the Oracle function is replaced by the function itself and new samples are concatenated to S_n) is performed until a given number of recursion is reached.

2.1.2.4 Adaptive Rendering

Most recent applications of adaptive sampling are for adaptive rendering. They adjusts sampling density and filtering to improve rendering efficiency. Sparse linear models and optimal sampling windows are recursively updated to best reconstruct neighboring pixels [Moon et al., 2015]. An error estimator guides sampling to high variance regions. Adaptive sampling algorithms often require significant memory and computing power to solve an optimization function, e.g., in the gradient domain [Manzi et al., 2016] for path tracing. Typically, two passes are required [Moon et al., 2014]. In the first phase, a small number of uniform samples are allocated to gather the error. In the second pass, additional samples are allocated based on error metrics in the first pass. An extended version of the adaptive sampling is iterative adaptive sampling which continues sampling until a certain variation criteria is met. We seek innovations to use only one pass. We propose a single pass adaptive sampling model that only needs to store one additional texture across frames and with little calculation overhead. The proposed one pass model is able to pick up dynamic light and motion changes and adjust samples effectively. Recently, researchers at NVIDIA applied neural networks (U-Net) as adaptive sampling metrics on feature buffers (e.g., normals, depth and albedo at first hit) and the projected rendering in the previous frame to estimate sample count maps in a single pass as well [Hasselgren et al., 2020, Hofmann et al., 2021]. However, it's too expensive and not suitable for this adaptive sampling task.

2.2 Subsurface Scattering

In this section, we first review the research literature for physically-based subsurface scattering for real-time rendering. Then, we introduce some background for different models and real-time acceleration techniques.

Early physically-based rendering algorithms for subsurface scattering used Monte Carlo path tracing to solve the radiative transfer equation. Even with the introduction of Bidirectional Scattering-Surface Reflection Distribution Function (BSSRDF) [Nicodemus et al., 1977] to simplify subsurface integration onto surface domain, these algorithms took hours to generate an image on hardware of the day. Jensen introduced a BSSRDF light diffusion model with the dipole [Jensen et al., 2001] and multi-dipole [Donner and Jensen, 2005] approximations. The simpler dipole method only considers isotropic subsurface scattering in highly scattering media, but introduced the *diffuse reflectance profile*, which allowed compact representation and efficient evaluation.

Two different directions that reduce the complexity for real-time rendering when using diffuse profiles are online sampling and pre-integration. Online sampling is designed with an assumption that subsurface scattering can be approximated by a weighted sum of artist-friendly kernels [Borshukov and Lewis, 2005] or dipole-based Gaussian kernels [d'Eon et al., 2007]. The sample count is further reduced by shifting from texture space [dEon and Luebke, 2007] per subsurface object to screen space [Jimenez et al., 2009, 2015] after deferred rendering. In contrast, pre-integration methods offload online sampling based on the dipole approximation into texture look-ups based on the assumptions that subsurface scattering is visible when there is light gradient change [Penner and Borshukov, 2011] and the distance to shadow can be correctly reconstructed.

Recent advances in physically-based subsurface rendering fit an approximation to Monte-Carlo results instead of fitting the dipole model, which is already an approximation [Christensen and Burley, 2015]. This technique has been recently incorporated into the Unity game engine with pre-calculated sampling [Golubev, 2018].

2.2.1 Real-time Model

At a surface point p, the outgoing radiance for direction ω_o is

$$L_o(p, \omega_o) = \int_{\partial\mathbf{\Omega}} \int_{\mathbf{S}^2} L_i(q, \omega_i) S(q, w_i, p, w_o) d\omega_i dq, (q \in \partial\mathbf{\Omega}), \tag{2.35}$$

where L_i is the incoming radiance at point q from the direction ω_i. ω_i integrates over all incoming directions of the hemisphere $\mathbf{S}^2$. q integrates over the local surface patch $\partial\mathbf{\Omega}$. S is the *bidirectional scattering-surface reflectance distribution function* (BSSRDF). S depicts the ratio of reflected radiance at p for direction ω_o given an incoming flux from direction ω_i at q:

$$S(q, w_i, p, w_o) = \frac{dL_o(p, \omega_o)}{d\Phi_i(q, \omega_i)} [m^{-2} \cdot sr^{-1}]. \tag{2.36}$$

Eq. 2.36 is often simplified by assuming a radially symmetric S in a homogeneous semi-infinite planar medium as

$$S(q, w_i, p, w_o) = C F_t(q, \omega_i) R(r_q) F_t(p, \omega_o), \tag{2.37}$$

where $r_q = ||p - q||$, $F_t(\cdot)$ is the directional fresnel transmission term, $R(\cdot)$ is the *diffuse reflectance profile* and C is a constant term. Then, for a given flux reflection direction ω at p, the simplified subsurface scattering function with symmetric diffusing profile at the surface point p becomes

$$L_o(p, \omega) = \int_{\partial \mathbf{\Omega}} R(r_q) \cdot \boldsymbol{b}(p, q, \omega) \cdot dq \tag{2.38}$$

$$\boldsymbol{b}(p, q, \omega) = \int_{\mathbf{S^2}} C F_t(p, \omega) F_t(q, \omega_i) L_i(q, \omega_i) \langle \omega_i, n_q \rangle d\omega_i, \tag{2.39}$$

To use this formula, e.g., in deferred rendering, the internal integration can be pre-calculated into irradiance textures for all lights as $\boldsymbol{b}$. The outer one can be implemented as a post-processing pass. In the real-time rendering literature, there are two branches of work as to how diffuse reflectance profile can be modeled.

2.2.1.1 Dipole Profile

Most work [Jensen et al., 2001, Donner and Jensen, 2005, d'Eon and Irving, 2011, Habel et al., 2013] since Grosjean [1959] decomposes light (or neutrons) into reduced-intensity, single-scattering, and multi-scattering terms as

$$S = S^{(0)} + S^{(1)} + S_d \tag{2.40}$$

based on a Neumann series and handles them separately. The reduced intensity transmission $S^{(0)}$ is the direct radiance without scattering, and can be implemented by standard transparency algorithm based on the Lambert-Beer Law. $S^{(1)}$ is an integration of all single scattering events in the media [Jensen et al., 2001]. The multi-scattering term S_d models all the remaining scattering events and is simplified with dipole diffusion approximation. The corresponding diffuse reflectance profile

R_d based on classic dipole is

$$R_d(r) = \frac{\alpha' z_r(1+\sigma_{tr}d_r)e^{-\sigma_{tr}d_r}}{4\pi d_r^3} - \frac{\alpha' z_v(1+\sigma_{tr}d_v)e^{-\sigma_{tr}d_v}}{4\pi d_v^3} \tag{2.41}$$

where $\alpha' = \frac{\sigma_s{}'}{\sigma_t{}'}$ is the reduced albedo. $\sigma_s{}'$ and $\sigma_t{}'$ are the reduced scattering coefficients and reduced extinction coefficient with $\sigma_t{}' = \sigma_a + \sigma_s{}'$. σ_a is the absorption coefficient. $\sigma_{tr} = \sqrt{3\sigma_a\sigma_t{}'}$ is the effective transport coefficient. $z_r = \frac{1}{\sigma_t{}'}$ and $z_v = \frac{1+4(1+F_{dr})/(3(1-F_{dr})))}{\sigma+t'}$ are the positive and negtive source positions with a reference to $z = 0$. d_r and d_v are the corresponding distance to the sources from point q with $d = \sqrt{r^2+z^2}$. F_{dr} is the diffuse Fresnel reflectance that can be approximated by polynomial expansions [Egan et al., 1973] as

$$F_{dr} \approx \begin{cases} -0.4399 + \frac{0.7099}{\eta} - \frac{0.3319}{\eta^2} + \frac{0.0636}{\eta^3}, \eta < 1 \\ -\frac{1.4399}{\eta^2} + \frac{0.7099}{\eta} + 0.6681 + \frac{0.0636}{\eta^3}, \eta > 1 \end{cases} \tag{2.42}$$

where η is the ratio of index of refraction.

Although more accurate dipole models are available (e.g., multipole) in the literature of computer graphics, we do not cover deeper details for the scope of real-time rendering and our major focus is on Burley's profile.

2.2.1.2 Burley's Profile

Instead of approximating scattering events separately, Christensen and Burley [2015] directly approximate the diffuse reflectance profile $R(\cdot)$ based on empirical

MC simulation data including all scattering terms. $R(\cdot)$ can be well approximated by a sum of two exponential functions in terms of distance r as

$$R(r) = A\frac{e^{-r/d} + e^{-r/(3d)}}{8\pi dr}, \tag{2.43}$$

where A is the surface albedo with $A = \int_0^\infty R(r)2\pi r dr$ [Jensen and Buhler, 2005] to make the profile normalized as Burley's Normalized Profile. The d term is fit to a free path length parameter ℓ based on the configuration Θ,

$$d = \begin{cases} \ell/(1.85 - A + 7|A - 0.8|^3) & \Theta = \Theta_1 \\ \ell/(1.9 - A + 3.5(A - 0.8)^2) & \Theta = \Theta_2 \\ \ell/(3.5 + 100(A - 0.33)^4) & \Theta = \Theta_3 \end{cases} \tag{2.44}$$

where we have:

1. *Search light configuration* ($\Theta = \Theta_1$). Light enters the volume perpendicular to the surface. ℓ is the volume mean free path.

2. *Diffuse surface transmission* ($\Theta = \Theta_2$). ℓ is the volume mean free path for a rough material after ideal diffuse transmission.

3. *Diffuse mean free path as parameter* ($\Theta = \Theta_3$). ℓ is the diffuse mean free path on the surface, which is denoted as ℓ_d.

We choose Θ_3 as being more artist friendly for typical game uses. With this choice, we are able to convert fitted material parameters from real world [Jensen et al., 2001] to Burley's model with the following formulas [Christensen and Burley, 2015]:

$$\ell_d = \frac{1}{\sigma_{tr}} \tag{2.45}$$

$$\sigma_{tr} = \sqrt{\frac{\sigma_a}{D}} \tag{2.46}$$

$$D = \frac{\sigma_t + \sigma_a}{3\sigma_t^2} \tag{2.47}$$

where D is the diffusion coefficient. $\sigma_t = \sigma_a + \sigma_s$ is the extinction coefficient and σ_s is the scattering coefficient. $\sigma_s' = \sigma_s(1-g)$ is reduced scattering coefficient that is affected by g, the mean cosine of the scattering angle. g parameterizes how light is forward ($g > 0$) or backward ($g < 0$) scattered. When $g = 0$ light is scattered uniformly at each scattering event. A computed result for diffuse mean free path for common materials [Jensen et al., 2001] is provided in Table C.1 when $g = 0$. Materials from the table will be used in this dissertation.

2.2.2 Real-time Acceleration Techniques

In order to make subsurface scattering run efficiently on GPU and to be more photo-realistic, subsurface scattering has shifted to texture space and then to screenspace. There are several acceleration techniques proposed in the literature, of which Gaussian blur, pre-integration and separable are largely adopted in real-time rendering engines. In this dissertation, we propose another technique that accelerates subsurface with higher rendering quality based on our novel adaptive sampling technique as adaptive filtered importance sampling. It can also be used for other rendering tasks.

2.2.2.1 Guassian Blur

Real-time realistic subsurface scattering is first explored by Borshukov and Lewis [2005] for human face rendering for the movie 'The Matrix Reloaded'. Instead of expensive techniques, like path tracing or volumetric scattering simulation, they use an artist-friendly kernel to blur the gathered irradiance texture to approximate subsurface scattering for human face rendering for the movie 'The Matrix Reloaded'. The work by d'Eon and his colleagues in the NVIDIA human head demo [d'Eon et al., 2007] brings the realistic part of dipole diffusion profile, and the real-time part of using Gaussian filters to approximate dipole, together into texture space. They created the first realistic and real-time subsurface scattering for human skins that matched an offline renderer.

A diffusion profile is approximated by a sum of zero-mean isotropic Gaussian as

$$R_d(r) \approx A_g(r) = \sum_{i=1}^{k} w_i G(v_i, r), \tag{2.48}$$

we need to minimize the error function [d'Eon et al., 2007]

$$V(v_1, ..., v_k | R_d) = \int_0^\infty r(R_d(r) - \sum_{i=1}^{k} w_i G(v_i, r))^2 dr, \tag{2.49}$$

where $A_g(r)$ is the Gaussian approximation, the isotropic Gaussian has $G(v,r) = \frac{1}{2\pi v}e^{-\frac{r^2}{2v}}$. k is the number of Gaussian. w_i is a normalization weight with $\sum_i^k w_i = \int_0^\infty 2\pi r R_d(r)dr$. With Levenberg-Marquardt optimization to minimize Eq. 2.49, a relative RMS power error (defined as $\frac{\sqrt{V(v_1,\ldots,v_k|R_d)}}{\sqrt{\int_0^\infty r(R(r))^2 dr}}$) reports an error of 1.25% for green wavelength of marble material [Jensen et al., 2001] when $k = 4$, and an error of 0.093% when $k = 8$. For a different diffusion profile, a separate set of Gaussians needs to be fitted.

The authors also found a closed formula that can fit to dipole more easily without worrying about convergence when $k = 4$. Because dipole models are a sum of two or more pole functions and the variances. The variance v are almost consistent. For more detail, please refer to the original paper.

Since symmetric 2D Gaussians can be decomposed into two 1D convolutions, vertically and horizontally. It makes the GPU implementation efficient. Moreover, if we have multiple radial profiles, they can be combined into new Gaussians one by one before integration as

$$\sum_{i=1}^{k_1} w_i G(v_i, r) * \sum_{j=1}^{k_2} w_i' G(v_i', r) = \sum_{i=1}^{k1} \sum_{j=1}^{k_2} w_i w_j' \left[G(v_i, r) * G(v_j', r)\right]. \tag{2.50}$$

2.2.2.2 Pre-integration

Since subsurface scattering is most visible where lighting changes, [Penner and Borshukov, 2011] moves in another direction to accelerate real-time subsurface scattering for skin, by offloading online Gaussian sampling into pre-integrated textures

(BRDF for lighting, and penumbrae for shadowing). During shading, it is sufficient to perform texture lookups based on curvature and shadow parameters.

Smooth surface. Subsurface scattering is most observable in a gradient change region. A fast approximation is to use normal and shadow penumbrae. The pre-integrated 2D texture $T(\frac{1}{r}, N \cdot L)$ for different curvature can be fit with

$$D(\theta, r) = \frac{\int_{-\pi}^{\pi} \max(0, \cos(\theta + x)) \cdot R(2r\sin(x/2))dx}{\int_{-\pi}^{\pi} R(2\sin(x/2))dx}, \tag{2.51}$$

where r is the radius and the plane curvature $\kappa = \frac{1}{r}$. θ is the angle offset from the normal N. R is the Diffusion profile as a sum of Gaussians. This model assumes that the material is on a smooth sphere.

Small details. However, small details are often present, like small wrinkles and pores on skin that are often represented with a detail normal map on a smooth mesh. To approximate a correct result for those regions, the normal can be pre-filtered with the diffusion profile for each channel (R, G, B). Since four normal maps are bandwidth-demanding in real-time rendering, only red normal is created while the other two as blendings between the specular normal and the clamped blurry red normal.

Shadows. For shadow regions, the pre-integrated 2D texture based on penumbra $T(\frac{1}{w}, s)$ can be fit by

$$P_S(s, w) = \frac{\int_{-\infty}^{\infty} P'(P^{-1}(s) + x) \cdot R(x/w)dx}{\int_{-\infty}^{\infty} R(x/w)dx}, \tag{2.52}$$

where $P(\cdot)$ is the shadow penumbra, a 1D falloff function that determines how light transitions from full light to dark. It maps the penumbra width (distance to full light) to the original shadow value s in world space. $P^{-1}(\cdot)$ is the inverse function to query the penumbra width. $P'(\cdot)$ is a new shifted shadow penumbra function that allows full subsurface scattering integration before falloff due to shadowing. This shift leaves a segment of penumbra that the subsurface scattering can be saved in the pre-integrated 2D texture.

2.2.2.3 Separable Approximation

Low rank approximation. Instead of separating out individual Gaussians from the Gaussian approximation of the diffusion profile (e.g., a sum of 4 Gaussians), and applying blur one by one [d'Eon et al., 2007], Jimenez et al. [2015] proposed to use low-Rank approximation of the discrete version of the whole profile with Singular Value Decomposition (SVD) as

$$A_s = U\Sigma V^T, \tag{2.53}$$

$$U = (u^{(1)}|u^{(2)}|...|u^{(m)}), \tag{2.54}$$

$$V = (v^{(1)}|v^{(2)}|...|v^{(m)}), \tag{2.55}$$

$$\Sigma = diag(\sigma_1, \sigma_2, ..., \sigma_m), \tag{2.56}$$

where $A_d \in R^{m\times m}$ is the discrete version in Cartesian Coordinates of the continuous profile R (or $A_g(r)$ as a convenient real-time approximation) in polar Coordinates.

$u^{(i)}$ and $v^{(i)}$ are the ith column of the matrix. σ is in descending order. Then the r-rank approximation based on the *Eckart-Young theorem* [Eckart and Young, 1936, Stewart, 1993] is to keep the first r largest singular values while replacing others by zero as

$$A_s^{\{r\}} = U\Sigma^{\{r\}}V^T, \tag{2.57}$$

$$\Sigma^{\{r\}} = diag(\sigma_1, ..., \sigma_r, 0, ..., 0). \tag{2.58}$$

To increase the cache coherence in real-time rendering by using 1D filters, the discrete profile $A_d^{\{r\}}$ is further decomposed into separable models a_i with

$$A_s^{\{r\}} = \sum_{i=1}^{r} \sqrt{\sigma_i}u^{(i)} \otimes \sqrt{\sigma_i}v^{(i)} = \sum_{i=1}^{r} a_i \otimes a_i \tag{2.59}$$

where $\otimes$ is the outer product. Since the profile R is radially symmetric, we have $u^{(i)} = v^{(i)}$. It makes the 1D filter function $a_i = \sqrt{\sigma_i}u^{(i)}$. Then the 2D r-rank filter can be applied to an irradiance texture with r separable horizontal and vertical filters. In the ith pass, both horizontal and vertical have the same weight a_i.

Pre-integrated models. However, if the irradiance is known to be additively separable $\textit{б}(x, y) = \textit{б}(x) + \textit{б}(y)$ with $\frac{\partial \textit{б}}{\partial x \partial y} = \frac{\partial \textit{б}}{\partial y \partial x} = 0$, a ‘rank-1 approximation’ pre-integrated separable kernel is

$$A_p(x, y) = \frac{1}{||R||_1} a_p(x) a_p(y), \tag{2.60}$$

which can be derived in the Cartesian Coordinates for Eq. 2.38 from

$$
\begin{aligned}
L_o(x,y) &= \int R(x-x',y-y')\boldsymbol{b}(x-x',y-y')dx'dy', && (2.61)\\
&= \int a_p(x-x')\boldsymbol{b}(x-x')dx' + \int a_p(y-y')\boldsymbol{b}(y-y')dy', && (2.62)\\
&= \int\int \frac{1}{||R||_1} a(x-x')a(y-y')\boldsymbol{b}(x-x',y-y')dx'dy', && (2.63)\\
&= \int\int A_p(x-x',y-y')\boldsymbol{b}(x-x',y-y')dx'dy', && (2.64)\\
a_p(x) &= \int R(x,y)dy, && (2.65)\\
a_p(y) &= \int R(x,y)dx, && (2.66)\\
||R||_1 &= \int a(x)dx = \int a(y)dy. && (2.67)
\end{aligned}
$$

With this assumption, we can achieve accurate results when the shadow is either vertical or horizontal. However, this formulation is not artist-friendly as mentioned in the original paper [Jimenez et al., 2015] because $R = R_d$, the diffusion profile, which is often Gaussian approximations of a dipole model measured that is fixed at approximation time. But, in real-time rendering applications, the pre-integrated separable model implementation[1] is widely adopted where the result is interpolated between the center pixel and the pre-integration. Since we have an alternative model R as diffuse reflectance profile in Burley's normalized profile, it makes the control artist-friendly when we select the diffuse mean free path parameterization Θ_3, which enables us to create a hybrid of this separable model with our proposed algorithm for better quality and performance (See Section 5.3.2).

---

[1] separable-sss, Git Repository, https://github.com/iryoku/separable-sss

Chapter 3: Heterogeneous Real-time Rendering

> All truths are easy to understand once they are discovered.
>
> —Galileo Galilei

Photo-realistic rendering relies heavily on stochastic sampling to achieve perceived realism. However, it is a great challenge to do physically-based rendering in real-time since it has high incoherent cache demands. Each interaction causes the sampling of complex materials required for rendering. Moreover, the access does not always follow the assumption of high temporal and spatial coherence that the current GPU architectures have. For example, to have more physically correct subsurface scattering in screen space, we need to sample the irradiance texture stochastically, counter to the current GPU architecture design. Meanwhile, we need to access utility textures (e.g., pixel subsurface profile and/or normal) in the same stochastic way to resolve how different subsurface appearances should be blended per pixel, which further exaggerates the cache incoherence and bandwidth demands.

What novel and scalable techniques can we use to target high-quality real-time rendering with contemporary and next-generation hardware? This research question drives us to explore and to provide novel solutions in this dissertation. We strive to develop cache and bandwidth aware algorithms to achieve the best rendering quality under a given tight bandwidth and cache budget. However, there is no taxonomy that we can refer to in the literature to help the design of our algorithms and show the significance to potential readers clearly. With this motivation, we develop such a taxonomy. We review the basic GPU cache architecture and its bandwidth, the source of incoherence in photo-realistic real-time rendering. We then create a taxonomy of techniques in real-time rendering and present our work within that framework.

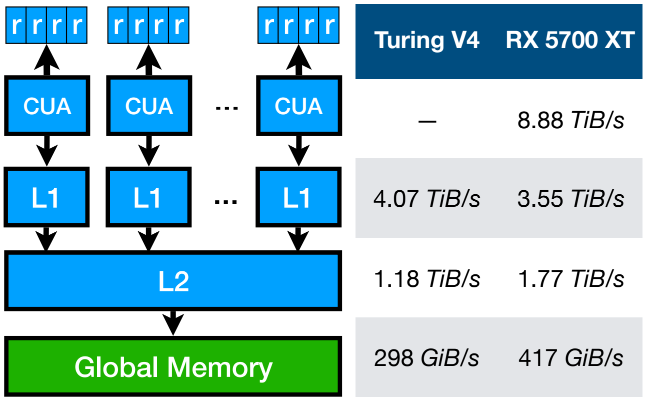
3.1 GPU Cache

In current GPU architecture design, there is an assumption of temporal and spatial coherence. 1) Spatial coherence indicates that the next address visited has a high probability of being in the cache lines already filled. For example, array elements have high spatial coherence if they are accessed in succession. 2) Temporal coherence indicates recently visited context has a high chance of being revisited again. For example, the variable to hold the sum of data has high temporal coherence. Any algorithm implementation that follows those two assumptions helps to increase the cache hit rate and reduce the traffic to caches with lower bandwidth, thus, improving the performance.

Based on this design assumption, different efficient memory architectures are proposed by NVIDIA, AMD, Intel, etc. Although not all GPU architecture information is publicly available online, we find two similar GPU cache architectures between NVIDIA Turing and AMD RDNA [Jia et al., 2019, NVIDIA, 2018, AMD, 2019].

3.1.1 Simplified GPU Data Cache Architecture

There are basically four types of data cache in a contemporary GPU (e.g., Volta V100, Turing T4 GPU, and RX 5700 XT) [Jia et al., 2019, NVIDIA, 2018, AMD, 2019]. The publicly available bandwidth is also presented as a performance reference for each type in Fig. 3.1.



Turing V4	RX 5700 XT
–	8.88 *TiB/s*
4.07 *TiB/s*	3.55 *TiB/s*
1.18 *TiB/s*	1.77 *TiB/s*
298 *GiB/s*	417 *GiB/s*

Figure 3.1: Simplified GPU data cache architecture for Turing and RDNA. It also shows a coarse bandwidth between different cache levels for Turing V4 [Jia et al., 2019] and RX 5700 XT [AMD, 2019]. The computing unit array (CUA) stands for streaming multiprocessors (SM) for Turing, and for dual computing unit array for RDNA.

- *L0/Registers.* For each of the four processing block in a streaming multiprocessor, there is 64 KiB of fast access registers [Jia et al., 2019]. For each dual computing unit, there are four SIMDs that operate independently. The known caches include vector and scalar registers, local data share (LDS), two 16 KiB L0 vectors, and one 16 KiB L0 scalar caches. They have the highest bandwidth. The RDNA whitepaper presents an L0 cache bandwidth of 8.88 TiB/s for RX 5700 XT.

- *L1/shared data cache.* L1 data cache and shared memories are combined in Turing and Volta architecture to reduce hit latency and improve the bandwidth. It is shared between multiple threads in SM. However, it is incoherent between different SM to maximize the throughput. The hit latency is 32 clock cycles on Turing T4 GPU with a load throughput of 4.07 TiB/s. For RDNA architecture, an L1 cache is first introduced for each array of dual computing units with the size of 128 KiB. L1 bandwidth is 3.55 TiB/s for RX 5700 XT.

- *L2 data cache.* The L2 data cache is unified and coherent within GPU (instruction and constant memory are the same for Turing Architecture). On Turing T4 GPU, it has a latency of about 188 clock cycles and a load throughput of 1.18 GiB/s. On RX 5700 XT, the bandwidth is 1.77 TiB/s.

- *Global Memory.* A large chunk of memory unique to a GPU. Although data fetched from global memory are automatically cached in L1 and L2, the GDDR6 memory creates a hard limit of 298 GiB/s theoretical bandwidth on Turing T4 GPU. RX 5700 XT reports a bandwidth of 417 GiB/s.

AMD has recently released AMD Infinity Cache, L3 Caches, on RX 6800 [AMD, 2020], which reduces the bandwidth demands to the global memory. Even if all data can be cached in L3, the cache incoherence still exists between different cache levels due to the demanding access from all computing units. When a rendering task is limited by bandwidth, the corresponding cache and bandwidth aware rendering algorithm is designed so that the data resides in lower cache levels and the bandwidth demands are reduced as much as possible from higher cache levels.

This is generally achievable in most current algorithms, as the data can be efficiently stored in different registers, like LDS, in L0. However, there are increasing demands of high native resolutions in the gaming industry (e.g., from 1K to 4K), and some algorithms (e.g., based on stochastic sampling) have intrinsic high bandwidth demands to higher cache levels. The current bandwidth cannot fully satisfy the needs and creates a lot of cache misses.

3.1.2 Cache Miss Type

On CPU, caches are designed to reduce latencies. The source of cache misses can be summarized as a well-known model, *3C model* [Hill and Smith, 1989]: *1) Compulsory.* It is the cold start that causes cache misses due to accessing a data block for the first time. From architecture design, it can be reduced by increasing block size but at the risks of increasing cache miss rate; *2) Capacity.* The cache is not large enough to contain all data blocks. It can be resolved by increasing cache size but at the risk of increasing access time, or by reusing the loaded data more frequently, e.g., by preprocessing discrete data into sequential data blocks; *3) Conflict.* Multiple memory addresses are mapped to the same cache location due to limited associative or replacement policy [Nugteren et al., 2014]. It can be mitigated by increasing cache size, increasing associativity, or improving the replacement policy but at the risk of increasing the access time.

GPUs are also designed with 3C model considered. However, instead of reducing latencies, GPUs are designed to hide the high latency through multi-threading, using the on-chip memory to reduce the bandwidth-limited off-chip traffic [Nugteren et al., 2014]. Making full use of on-chip memory is essential to exploit the full potential of the parallel capability of GPU. However, this architecture design is not always helpful for photo-realistic real-time rendering with per-pixel Monte Carlo sampling that accesses arbitrary rendering resources of different types and sizes without a compromise of biased sampling. This hiding mechanism will still be limited by the bandwidth bottleneck.

3.1.3 Source of Incoherence

The core of current photo-realistic rendering, especially for offline rendering, is stochastic sampling, which intrinsically does not follow this architecture design. The incoherence comes from Monte Carlo sampling, large data elements, and ray and path tracing.

- Photorealistic rendering requires many samples per pixel with Monte Carlo sampling based on some cumulative density function. It makes each access of the content intrinsically incoherent. It becomes a performance hazard when the context accessed cannot be efficiently placed on-chip. One solution is to utilize a prebuilt mipmap of target textures (e.g., pre-filtered environmental map [Křivánek and Colbert, 2008]) to increase the cache hit rate.

- However, when the scene material has complex fine details, each interaction must access and evaluate a large chunk of data that might not be able to go through the mipmap chain, at the risk of increasing the cache misses, especially

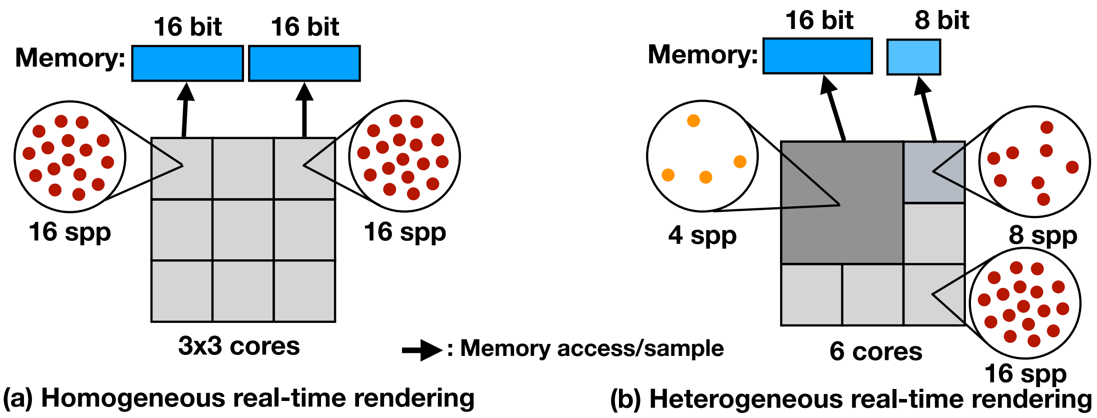


Figure 3.2: Comparison between (a) homogeneous and (b) heterogeneous real-time rendering. Heterogeneous real-time rendering can dynamically reduce the sampling, shading units, and memory demands without noticeable quality degradation.

for SIMD architecture. This data might include not only geometry information but material and utility buffer or texture overheads.

- Moreover, ray tracing recently introduced in real-time rendering fires several rays per pixel, and each can bounce into any direction several times incoherently into a complex scene.

3.2 Framework

There are different architecture and algorithm-level methods to deal with this memory incoherence. In this dissertation, we mainly focus on algorithm-level methods. Specifically, we proposed a new concept to classify the algorithms and guide our algorithm design: heterogeneous real-time rendering instead of the existing homogeneous real-time rendering. In the literature of real-time rendering, the rendering process is usually assumed to be homogeneous, where a fixed number of samples are gathered/scattered into a single or several fixed numbers of outputs. The off-chip traffic is a critical issue if the computation has to have a homogeneous use and shading of the resource. However, to achieve the same quality in physically-based rendering, not every output needs the same number of resource samples. Heterogeneous sampling is enough. Moreover, adjacent results might be so similar that less computation is sufficient (e.g., Variable Rate Shading). This observation creates new opportunities and new challenges for real-time rendering research by exploring the observed heterogeneity. Fig. 3.2 shows an example of the difference between homogeneous and heterogeneous real-time rendering from the computing, sampling, and memory demands. Note that our discussion excludes research that deals with bandwidth between CPU and GPU and focuses on internal bandwidth within GPU. Fig. 3.3 shows an overview of the taxonomy and our novelty within this framework for subsurface scattering.

3.2.1 Heterogeneous Computing Demands

For homogeneous rendering, all computing/shading units are of the same type and uniformly used for a task. A good example to improve the rendering efficiency is dynamic resolution, where the resolution and the corresponding required number of units are scaled uniformly in width and height. This has been adopted in many real-time rendering engines (e.g., Unreal Engine [Engine, 2020] and Unity3D [Unity, 2019]) and games when a constant frame time is the optimization goal. Other techniques, including *tiled rendering* [Fuchs et al., 1989, Ribble, 2008], and *checkerboard* [Wihlidal, 2017], also reduce stalling due to limited bandwidth, com-

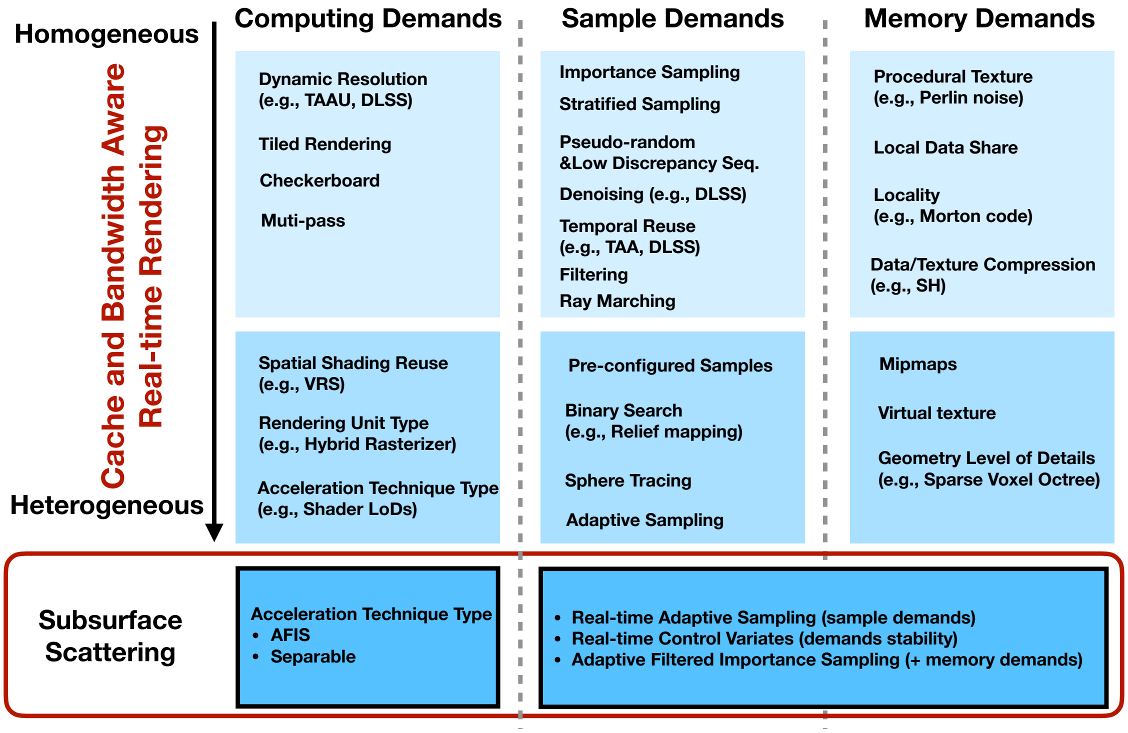

Homogeneous
Cache and Bandwidth Aware Real-time Rendering
Heterogeneous
Computing Demands
Sample Demands
Memory Demands
Dynamic Resolution (e.g., TAAU, DLSS)
Tiled Rendering
Checkerboard
Muti-pass
Importance Sampling
Stratified Sampling
Pseudo-random &Low Discrepancy Seq.
Denoising (e.g., DLSS)
Temporal Reuse (e.g., TAA, DLSS)
Filtering
Ray Marching
Procedural Texture (e.g., Perlin noise)
Local Data Share
Locality (e.g., Morton code)
Data/Texture Compression (e.g., SH)
Spatial Shading Reuse (e.g., VRS)
Rendering Unit Type (e.g., Hybrid Rasterizer)
Acceleration Technique Type (e.g., Shader LoDs)
Pre-configured Samples
Binary Search (e.g., Relief mapping)
Sphere Tracing
Adaptive Sampling
Mipmaps
Virtual texture
Geometry Level of Details (e.g., Sparse Voxel Octree)
Subsurface Scattering
Acceleration Technique Type
• AFIS
• Separable
• Real-time Adaptive Sampling (sample demands)
• Real-time Control Variates (demands stability)
• Adaptive Filtered Importance Sampling (+ memory demands)

Figure 3.3: Cache and bandwidth aware real-time rendering algorithm taxonomy based on heterogeneity and the contribution of our real-time rendering technique for subsurface scattering in the taxonomy. It classifies algorithms to minimize the rendering computing demands, sample demands, and memory demands.

puting power, and caches by reducing the number of computing units at the same frame time. The fundamental data-parallel operations in most GPGPU are *gather* and *scatter/splatting* that are limited by bandwidth. To mitigate the performance degradation due to the low locality of random access, He et al. [2007] use multi-pass to increase locality utilization by processing each chunk of data in a close region at a time. It helps to increase the cache utility for algorithms like radix sort, hashed search, and sparse matrix multiplication.

Different categories of shading units could be used or combined (e.g., compute and/or ray tracing, software and/or hardware) to exploit heterogeneity to improve the efficiency further. For example, the rasterization of millions of tiny triangles could be more efficient with a software-based rasterizer than the hardware counterpart optimal for large triangles. The combination leads to better performance [Battaglia, 2020]. The ray-tracing result can correct shading bias from rasterizer units [Marrs et al., 2018]. Moreover, the computing demands can be reduced by different real-time acceleration technique types like shader levels of detail [Olano et al., 2003]. Different acceleration algorithms are called for different computing units under different conditions. In this dissertation, we contribute another in this category, real-time acceleration for subsurface scattering. In section 5.3.2, subsurface scattering can be accelerated in the same frame with either the separable approximation or our proposed algorithm adaptive filtered importance sampling. The type can be pre-determined or by custom process on the fly.

Furthermore, spatial shading reuse has gained popularity recently. A unit can be configured to create multiple outputs to reduce the bandwidth and computing demands further when the output is predicted to be low frequency with high probability. Examples include software and hardware adaptive/variable rate shading [He et al., 2014, NVIDIA, 2018, Drobot, 2020] and the hierarchical subdivision strategy [Mallett and Yuksel, 2018]. For rendering where eye tracking is possible, or in Virtual Reality (VR), non-linear transformation can be used (e.g., kernel foveated rendering [Meng et al., 2018]) to redistribute the computing unit density.

3.2.2 Heterogeneous Sample Demands

For homogeneous rendering, all sample counts are fixed inside each computing unit for a given algorithm. Different techniques have been exploited in real-time rendering to improve the quality (or reduce the variance) given a fixed number of samples. For example, there are importance sampling, stratified sampling, pseudo-random and low discrepancy sequence, denoising, separable and low-rank approximation, filtering, and temporal reuse (e.g., TAA [Karis, 2014] and DLSS [Liu, 2020]). Usually, some subsets of these techniques are combined differently for different implementations. In real-time rendering, even some bias is acceptable if the variance can be minimized. For example, the ray-marching algorithm places each sample uniformly for uniform volume tracing. It converges to zero variance but with bias [Novák et al., 2014]. SVGF [Schied et al., 2017] applies spatial and temporal filtering to denoise the one sample per pixel (spp) noisy path tracing result.

One widely adopted method for heterogeneity to improve efficiency is to have different sample count configurations per pass or material. Then it can be configured by users or developers for quality and performance balance. Researchers have tried to pursue heterogeneity with numerical search acceleration techniques like sphere tracing [Bastos and Celes, 2008], binary search in relief mapping [Policarpo et al., 2005], and adaptive sampling techniques in real-time rendering. For example, signed distance field (SDF) tracing uses an SDF to determine the next ray intersection point, which requires fewer intersection evaluations than uniform ray marching. Adaptive sampling tries to determine the sample count per computing unit with variance metrics based on local gradients and/or neighboring regions [Boksansky et al., 2019]. Those methods run efficiently if the variance is locally bounded (e.g., anti-aliasing). However, accessing those regions could increase the memory demands per pixel when it requires more regions. Moreover, it reduces the variance at the stake of correlating neighbor pixels.

3.2.3 Heterogeneous Memory Demands

For homogeneous rendering, each memory request accesses the same sized texel or buffer. Data are preferred to be cached to reduce the number of stalls due to limited bandwidth in registers, local data share, or cache levels with high locality access pattern (e.g., Morton code [Lauterbach et al., 2009, Nocentino and Rhodes, 2010], and ray sorting [Meister et al., 2020, Keller et al., 2019]). Different formats compress or decompress (either software or hardware based) on the fly so that the number of bytes transferred is minimized. In practice, the diffuse irradiance can be characterized by the first nine spherical harmonics coefficients [Ramamoorthi and Hanrahan, 2001] or the cone representation [Zeng et al., 2021] initially used to model ambient occlusion for cone tracing [Crassin et al., 2011]. An alternative method to reduce the bandwidth is using procedural techniques where each access can be directly evaluated on the fly at the cost of increasing computing demands (e.g., Perlin noise [Perlin, 1985]). Frequently, those techniques are combined together to balance bandwidth and computing power. Moreover, the access pattern (Morton code) can also reduce the bandwidth demand.

When large physically measured textures cannot be avoided, and the output surface is large (e.g., 4K resolutions), *Mipmaps* [Williams, 1983] can be used, where if most data access can be redirected to the high level of detail data, smaller memory demands can be stored in, e.g., registers or GPU caches. They require an accurate way to determine the level of detail. Mipmap level of detail is based on local derivatives for texture sampling. For more complex usage, like importance sampling, sampling density can be used. *Virtual texture* [Mittring and GmbH, 2008] brings in the virtual memory idea to support even larger textures. Different geometry levels of detail can also be used to reduce the bandwidth demands of a scene, including adaptive signed distance field [Frisken et al., 2000], sparse voxel octree [Kämpe et al., 2013], GVDB [Hoetzlein, 2016], and SSVDAGs [Villanueva et al., 2016].

3.3 Formulation

We propose a descriptive bandwidth demand algorithm for the cache and bandwidth aware real-time rendering. For computing demands, we model the total bandwidth demand for a given rendering task as

$$T(n) = kT'_k(n/k) + O(n), \tag{3.1}$$

$$T'(n) = \sum_{i=1}^{n} \sum_{a \in \mathcal{A}, r \in \mathcal{R}} w_{i,a,r} S_{i,a,r} + O(n), \tag{3.2}$$

where n is the final number of rendering pixels, k is the number of passes, $\mathcal{A}$ denotes the technique sets, $\mathcal{R}$ denotes the rendering unit type, S denotes the sample demands, and w is the compute demanding weight. Eq. 3.1 is the sum of the bandwidth demand of k passes, plus some additional per-pixel bandwidth. Eq. 3.2 is the sum over pixels, pass techniques, and compute unit types used of the compute demand weight and number of samples for each pixel/technique/compute unit, plus some additional per-pixel bandwidth. For example, in variable rate shading, if 2×2 pixels share the same computing unit output, $w_{i,a,r} = 0.25$. It is a constant weight in dynamic resolution and checkerboard rendering. Note that this bandwidth demand function is a cost function instead of being in actual bytes.

The corresponding bandwidth demand for a given sample count budget m is

$$S(m) = \sum_{j=1}^{m'} M(d_j) + O(m') \tag{3.3}$$

$$m' = f(m, \sigma_0^2), m' \leq m \tag{3.4}$$

where m is the sample budget, and m' is the actual sample count used for the corresponding computing unit derived from a technique function f that tries to meet certain variance criteria σ_0^2 adaptively or beforehand. This technique function tries to reduce the required number of samples based on different selected techniques illustrated in Fig. 3.3 to minimize the sample demands. Usually, this function is not automatically determined but by the algorithm designer with the help of manual experiments. d is the raw memory demand for each sample while M is the actual memory demand with certain memory demand reduction technique R as

$$M(d) = R(d) + O(d) \tag{3.5}$$

This formulation gives us a good start point for reasoning how much each proposed algorithm helps minimize the total bandwidth demands. For example, DLSS uses dynamic resolution to minimize the compute demanding weight $w_{i,a,r}$ in Eq. 3.2. To minimize sample demands, it reuses temporal histories and denoising through deep learning in Eq. 3.4 to keep m' low. During inference, the weight is constant and thus can be cached efficiently. The tensor core is used to accelerate the inference step to assure that the bottleneck is not the computing unit. From here, we can find that DLSS does not provide innovations for memory demands minimization.

In this dissertation, the major work is in sample demand minimization. We first introduce our novel statistically optimal real-time adaptive sampling techniques for Eq. 3.4 to minimize m' on the fly based on a given variance σ_0^2 in Chapter 4. Chapter 6 addresses the temporal instability issue due to dynamic lighting since we have used the temporal variance to estimate the sample count per computing unit. These serve as the foundation to minimize the sample demands, thus consistently reducing the total bandwidth demands. The technique introduced in these two chapters are generic and can be used for other rendering tasks to increase the performance when Monte-Carlo sampling is used in real-time rendering. In Chapter 5, the general homogeneous sample demand minimization techniques (i.e., importance sampling and low discrepancy sequence) are introduced. Then we separate diffuse from distant subsurface scattering to track the variance of distant scattering to reduce the overestimation due to direct scattering.

For memory demands minimization, We have proposed a technique, adaptive filtered importance sampling (AFIS), to use mipmaps in section 5.2. This technique is different from the existing technique [Křivánek and Colbert, 2008]. In our algorithm, the MIP level is affected by computing unit-varying m' instead of a constant sample count for all units.

For computing demands minimization, we have proposed combining two different techniques in section 5 for subsurface scattering: Separable and AFIS because the separable approximation has constant sample count and bandwidth demand while AFIS minimizes the bandwidth based on the assumptions that some regions only need a few samples to produce a low variance smooth image. For high-frequency lighting everywhere, the Separable approximation still performs better, though with visible variance.

3.4 Summary

This chapter introduces a taxonomy of cache and bandwidth aware real-time rendering techniques used to address bandwidth problems from homogeneity to heterogeneity. We have also proposed a descriptive algorithm to model the bandwidth at the algorithm level. We analyze our new rendering techniques through the taxonomy and performance model.

Chapter 4: Real-time Adaptive Sampling

> *"Do not plan for ventures before finishing what's at hand."*
>
> —Euripides

In real-time rendering, achieving a physically correct shading result requires real-time Monte-Carlo sampling per pixel per frame, which seems prohibitive to achieve. Since offline rendering has already confirmed that there is heterogeneous demand for sample count in different regions with adaptive sampling, we propose a real-time adaptive sampling method based on temporal variance to lower the required demand of samples.

Especially in this chapter, we propose a one-phase adaptive sampling pass, which is unbiased and able to adapt to scene changes due to motion and lighting. To further improve the quality, we explore temporal reuse with a guiding pass before the final temporal accumulation phase that further improves the quality. Fig. 4.1 visualizes the sample count based on this guiding pass. Our local guiding pass does not constrain the global temporal accumulation implementation, enabling us to support different temporal accumulation algorithms, including Temporal Anti-Aliasing (TAA) and deep learning-based algorithms like Deep Learning Supper Sampling (DLSS). Moreover, it requires only one additional texture to be

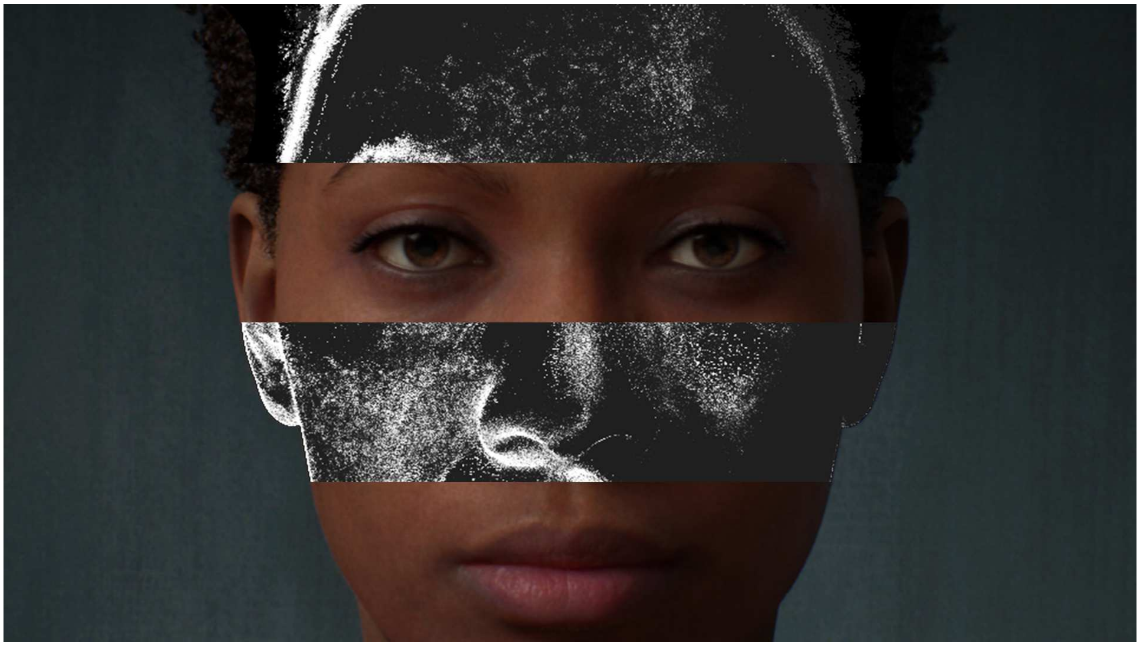


Figure 4.1: Sample count visualization in greyscale for real-time adaptive sampling on MetaHuman character *ada*.

passed between frames. Our proposed variance-guided algorithm has the potential to make a stochastic sampling algorithm effective for real-time rendering. This chapter is extended from our I3D conference (PACMCGIT) publication [Xie et al., 2020].

4.1 Introduction

A classic adaptive sampling algorithm consists of two phases. Usually, in the first phase, samples are gathered to determine the variance and then discarded to avoid bias. In the second phase, an adaptive number of samples is used to reduce the variance. However, it is inefficient for real-time application due to sample disposal and might lead to noise due to low sample counts in the first phase. Recent research tries to use all samples by posing the two sample sets as a multiple importance sampling problem [Grittmann et al., 2019]. In this chapter, we make it a single adaptive sampling guiding pass for subsurface scattering that makes use of each individual sample. Specifically, sample variances at each frame are continuously updated into a temporal buffer based on exponential moving variance. With temporal reuse, this variance at each frame reflects the pixel condition in the previous adjacent frames. It can control the sample counts for the next frame without visible noise for real-time stochastic sampling.

Since we cannot afford too many samples per frame in real-time rendering, we seek temporal reuse, which has recently been explored for improving the quality of 1 sample per pixel path tracing [Schied et al., 2017]. However, this work changes the existing temporal reuse algorithm, which does not fit well into existing rendering engines. We explore a different approach by adding a pass for guiding the subsurface scattering that assumes there is an existing temporal reuse pass like Temporal Anti-Aliasing (TAA), but without assumptions about the implementation of that temporal reuse pass, as long as it tries to make use of temporal information and to resolve artifacts like ghosting, blurring, lag, and flickering.

4.2 Basic Metrics

To estimate the minimal sample count, we need a metric to adaptively increase the number of samples when there is high perceivable noise and decrease when fewer samples are sufficient.

We estimate the minimal number of samples,

$$n_{(i)} = \max\left(\sigma^2_{M_{(i-1)}} \cdot n_{(i-1)} \Big/ \sigma_0^2,\ \beta_{(i-1)}\right), \tag{4.1}$$

where $n_{(i-1)}$ is the minimal sample count estimated in the previous frame, $\sigma^2_{M_{(i-1)}}$ is the pixel variance of the distribution mean estimated in the previous frame. The purpose of this formula is to reduce the variance of the distribution mean to a target level σ_0^2. Due to lighting and the chance that we might miss some details if we have too few samples, we always use at least $\beta_{(i-1)}$ samples.

We do not know the exact distribution per pixel, due to motion or lighting conditions, but by the central limit theorem, given a population of a finite mean and variance, the sampling distribution of the mean μ_M becomes a *normal distribution* of $(\mu, \sigma^2/N)$ as sample size increases, regardless of the shape of the original distribution. Therefore, we can estimate the variance in means as the distribution is sampled repeatedly across frames.

Using this algorithm to determine the minimal number of sample count in two phases would still be inefficient (i.e., in the first pass, collect samples to gather the variance, and estimate the sample count for rendering in the second pass). We adopt the idea of temporal accumulation from TAA to reduce the number of samples per frame and make it a one pass adaptive sampling technique.

4.3 Temporal Anti-aliasing

Temporal anti-aliasing (TAA) [Karis, 2014] is the de facto standard for anti-aliasing in real-time rendering engines. It amortizes sampling over time with an exponential moving average to accumulate consecutive frames in color space per jittered pixel. When an object moves, TAA reprojects samples from the accumulated history buffer along a per-pixel velocity vector. We summarize it as

$$\mu_i = (1-\alpha)\mathcal{C}(x_i, \Lambda) + \alpha\mathcal{S}(p_i);\ \alpha = \mathcal{M}(\alpha_0, \Lambda); p_i = \mathcal{N}(x_i, f(i)), \tag{4.2}$$

where μ_i is the estimated value at pixel $x_i \in \mathbb{R}^2$ in frame i. α is the exponential weight between the clamped history context term $\mathcal{C}$ and current-frame shading term $\mathcal{S}$. α is the per-pixel exponential moving average weight as computed by the weight update function $\mathcal{M}$ based on the user defined max weight α_0 and a context Λ that includes velocity, geometry type (e..g, transparent or not), neighbors etc. $\mathcal{C}$ is an operator to resample and reject the history projected from x_i. Both $\mathcal{C}$ and $\mathcal{M}$ are designed to minimize artifacts (i.e., ghosting, blurring, lag, and flickering) while preserving anti-aliasing results [Karis, 2014, Yang et al., 2009].

$\mathcal{S}$ is the shading function at p_i, where p_i is a jittered pixel position computed by the neighbor sampling function $\mathcal{N}$ from pixel position x_i and filter-kernel importance sampling offset f.

When used to accumulate MC results across multiple frames, the clamping and rejection performed by $\mathcal{C}$ and $\mathcal{M}$ need to either be modified to explicitly account for the MC process, or the incoming variance in $\mathcal{S}$ must be reduced to fit their rejection model. In modern game engines, TAA is used to accumulate samples from many MC processes, including glossy reflection, ambient occlusion, shadowing, and subsurface scattering. Methods that modify the clamping and rejection models for one MC method used in isolation are not effective given the multiple types of accumulation being performed by TAA. In this chapter, we do not modify the existing TAA process. Instead, we create one local variance-guided phase starting from Eq. 4.2 that outputs an adaptive sampling count per pixel per frame. The sampling result goes to the standard post-process pipeline and uses the existing TAA to further improve quality.

4.4 Metrics within Temporal Accumulation

To reduce sample count in each frame, we perform variance and sample count estimation in a local phase. With Eq. 4.2, we calculate an exponential moving average (EMA) over jittered pixel values. When α is small enough, it will eventually converge to $\mu_M = \mu$, the population mean [Karis, 2014]. Instead of recording all three channels, we record the moving average of gamma-corrected luminance to consider human perception. Knowing the sample count $n_{(i)}$ at frame i, we estimate the mean sample count, $\bar{n}_{(i)}$ as

$$\bar{n}_{(i)} = (1-\alpha)\bar{n}_{(i-1)} + \alpha n_{(i)}, \tag{4.3}$$

and the population variance σ^2 with exponential moving variance (EMV) [Finch, 2009] as

$$\sigma_i^2 = (1-\alpha)\sigma_{i-1}^2 + \alpha(1-\alpha)(\mathcal{S}(p_i) - \mathcal{C}(x_i, \Lambda))^2. \tag{4.4}$$

In this way, we can accumulate sample counts to solve Eq. 4.1. We now ignore $\beta_{(i-1)}$, it can be set after we get $n_{(i)}$. If a consecutive k frames are accumulated, the equation becomes

$$\sum_{i-k+1}^{i} n_{(j)} = \frac{\sigma^2_{M_{(i-1)}} \sum_{i-k+1}^{i} n_{(j-1)}}{\sigma_0^2}. \tag{4.5}$$

Then the estimated sample count for the current frame is

$$\begin{aligned}
\hat{n}_{(i)} &= \sum_{i-k+1}^{i} n_{(j)} - \sum_{i-k+1}^{i} n_{(j-1)} + n_{(i-k)} && (4.6)\\
&\approx \frac{(\sigma^2_{M_{(i-1)}} - \sigma_0^2)}{\sigma_0^2} \cdot \bar{n}_{(i-1)} \cdot k + \bar{n}_{(i-1)} && (4.7)\\
&= \frac{(\sigma^2_{M_{(i-1)}} - \sigma_0^2)}{\sigma_0^2} \cdot \bar{n}_{(i-1)} \cdot (k-1) + E[\bar{n}_{(i)}]. && (4.8)
\end{aligned}$$

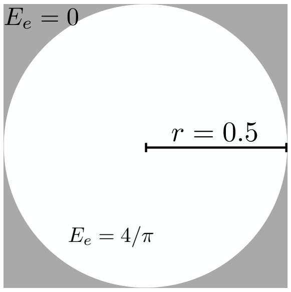
Since we are using EMA and EMV, we cannot maintain $n_{(i-k)}$, we approximate it by $\bar{n}_{(i-1)}$. Then, we estimate k as $k = 2/\alpha - 1$ based a common conversion between N-day EMA and simple moving average in analysis of financial data [Bauer and Dahlquist, 1998]. Eq. 4.8 is the expected sample mean, $E[\bar{n}_{(i)}] = \sigma^2_{M_{(i-1)}}/\sigma_0^2 \cdot \bar{n}_{(i-1)}$ at frame i derived from Eq. 4.5, plus a correction term. When $|\sigma^2_{M_{(i-1)}} - \sigma_0^2|$ is large, this correction term will be very large to aggressively reduce the variance to the target level in a single frame. We add a control factor κ to limit the per-frame correction. Then the final formula based on Eq. 4.8 is

$$\hat{n}_{(i)} = \kappa \cdot \Delta(i) + E[\bar{n}_{(i)}], \kappa \in [0, 1] \tag{4.9}$$

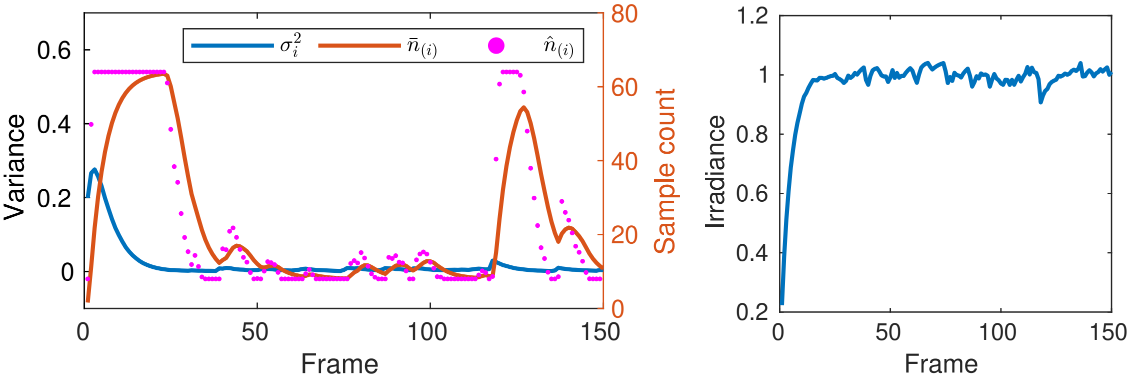
$$\Delta(i) = \frac{(\sigma^2_{M_{(i-1)}} - \sigma^2_0)}{\sigma^2_0} \cdot \bar{n}_{(i-1)} \cdot (k - 1), \tag{4.10}$$

where $\Delta(i)$ is the correction term. At $\kappa = 1$, this favors faster convergence at the cost of firing up to the maximum sample budget per frame, while at $\kappa = 0$ it uses time to accumulate enough samples.

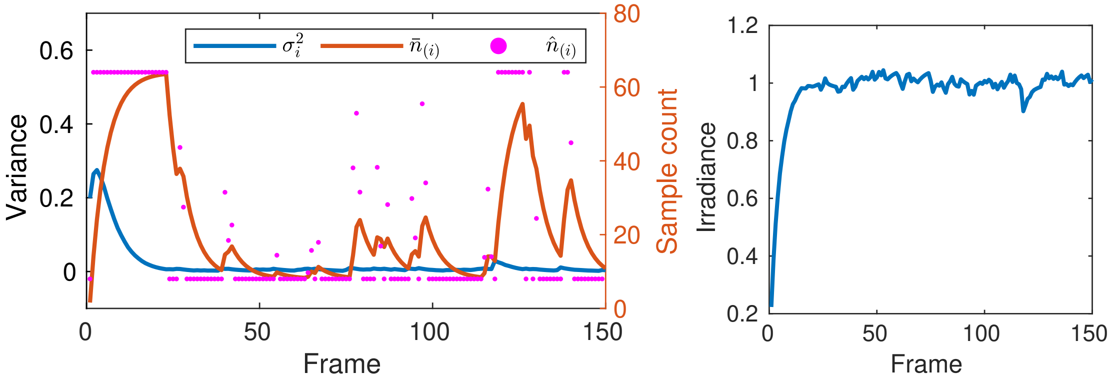
4.4.1 Circle Scenario



We create a simplified subsurface scattering scenario to help understand this estimation. We simplify the subsurface scattering scenario model to sample uniform irradiance in a circle of radius 0.5 centered at (.5, .5). We sample with a uniform 2D random number $(\xi_1, \xi_2) \in [0, 1)^2$ (instead of using Burley's subsurface scattering model). The accumulated irradiance is $4/\pi$ within the circle and 0 outside. We give the sample budget of $b_{\min} = 8$ spp and $b_{\max} = 64$ spp, use $\alpha = 0.2$ and a target quality level $\sigma^2_0 = 0.08^2$. The history tuple $\mathcal{H}_i = (\bar{n}_{(i)}, \mu_i, \sigma^2_i)$ is initialized to $\mathbf{0}$. Fig. 4.2 shows the sample count, variance and the scattering result over 150 frames for $\kappa = 0$ and $\kappa = 1$. After a *cold start* session, both methods try to converge to 1 within the given quality level. $\hat{n}_{(i)}$ in Fig. 4.2(a) and 4.2(c) shows how κ affects the actual samples used in each frame. The $\kappa = 1$ case has higher sample count peaks, usually lasting for a single frame, while $\kappa = 0$



(a) Sample count and variance ($\kappa = 0$).
(b) Result ($\kappa = 0$).



(c) Sample count and variance ($\kappa = 1$).
(d) Result ($\kappa = 1$).

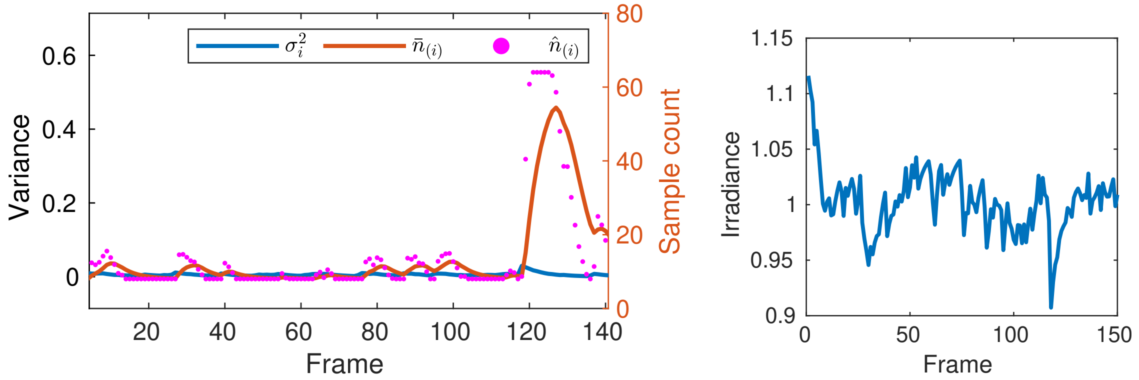
Figure 4.2: Sample count estimation. Effect of using different correction factors ($\kappa = 0$ vs $\kappa = 1$) in Eq. 4.9 in *Circle Scenario*. Disocculsion/*cold start* happens at the first frame. The spike at frame 120 is due to random sampling sequence.

uses fewer samples, but with sample count increases smoothed over several frames.

4.4.2 Disocclusion

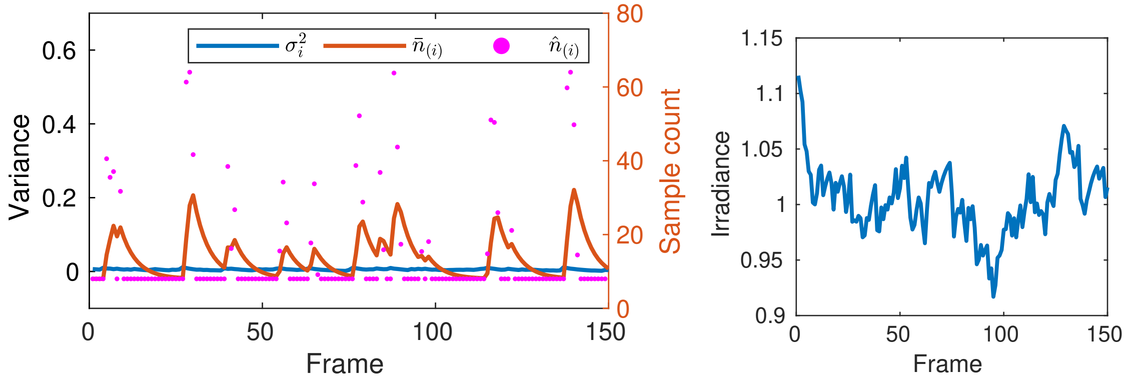
Fig. 4.2 shows the sample count per frame for a *cold start*, which is rare in typical rendering scenarios. The more common case for missing history data is disocclusion, when a previously hidden portion of an object becomes visible. We estimate the initial history when disocclusion happens.

Denote $\mathcal{C}_s(\cdot)$ as a point sampling operator without rejection on subsurface mask history. $\mathcal{C}_s(x_i, \Lambda) = 0$ when the subsurface mask history is not available in the



(a) Sample count and variance ($\kappa = 0$).

(b) Result ($\kappa = 0$).



(c) Sample count and variance ($\kappa = 1$).

(d) Result ($\kappa = 1$).

Figure 4.3: Sample count estimation considering disocclusion with the same configuration in Fig. 4.2. Disocculsion/*cold start* happens at the first frame. We initialize the history $\mathcal{H}_i$ with estimation from the initial sampling to solve the overestimation due to disocclusion (*cold start*).

previous frame. Then our new $\mathcal{M}$ has

$$\mathcal{M}'(\alpha_0, \Lambda) = \begin{cases} 1 & \mathcal{C}_s(x_i, \Lambda) = 0 \\ \alpha_0 & \text{otherwise} \end{cases}. \tag{4.11}$$

This operator enables the estimation of the initial sample value as $\mathcal{S}(p_i)$ with an initial sample count of $n_{(i)} = \max(\hat{n}_{(i)}, \beta_{(i-1)})$. We have no variance history, so estimate the variance as

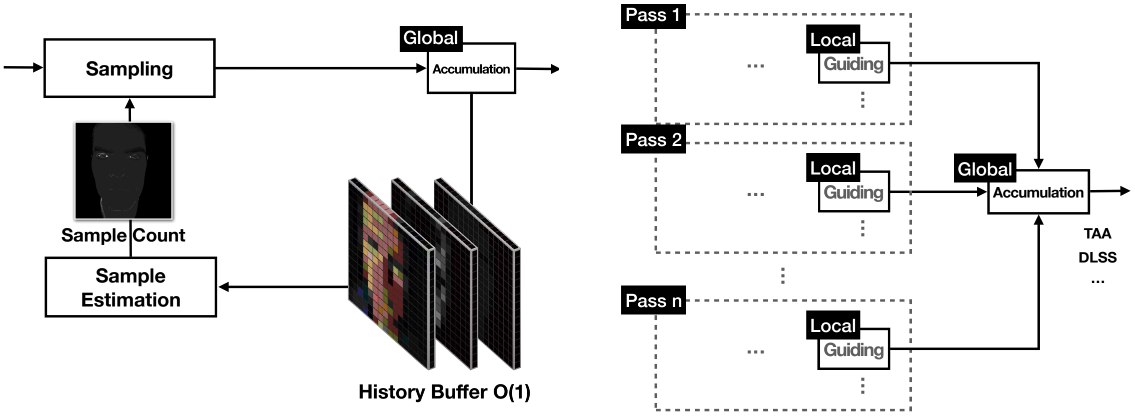
$$\hat{\sigma}_i^2(\alpha_0, \Lambda) = \begin{cases} \sigma_0^2 & \mathcal{C}_s(x_i, \Lambda) = 0 \\ \sigma_i^2 & \text{otherwise} \end{cases} . \tag{4.12}$$

Fig. 4.3 shows how the updated weight function $\mathcal{M}$ and variance estimation work under the same configuration as in the *Circle Scenario*. The initial high sample counts are eliminated.

4.5 Local Guiding Integration

The current adaptive sampling architecture is shown in Fig. 4.4(a). It relies on a global temporal accumulation pass (TAA) to update the history buffer. This design makes it depend on the existing global temporal accumulation pass, which has three drawbacks. It requires a modification of global temporal accumulation to output history. Second, the guiding pass might be affected by other passes like the overlaid transparency. At last, the global temporal accumulation parameter sets might be designed to improve the overall quality instead of a single pass.

To resolve this issue, We propose a local pass guiding for adaptive sampling in Fig. 4.4(b). We name it single pass variance guiding (SPVG). This design decouples the global temporal accumulation. No global temporal accumulation modification is required, and we have a custom local quality control, and we could have better quality input to global accumulation pass. With this design, we could even explore not only Global TAA but also other temporal accumulation algorithms. One



(a) Global accumulation framework

(b) Local guiding framework

Figure 4.4: Local guiding framework for real-time adaptive sampling. (a) Instead of relying on a global accumulation step to update the history buffer for real-time adaptive sampling, we use (b) a local guiding framework to update the history buffer for sample count estimation, then rely on the global accumulation for further quality improvement. This design enables us to use not only TAA but other accumulation designs like DLSS.

promising direction is to use Deep Learning for the accumulation. We also find that even DLSS, which is designed to have some accumulation but not targeted for a task (e.g., subsurface scattering), can help.

4.5.1 Global TAA

In the previous section, we estimated the sample counts at each frame with the assumption that local and global TAA use the same configuration (i.e., $\mathcal{C}_l = \mathcal{C}_g, \mathcal{S}_l = \mathcal{S}_g, \Lambda_l = \Lambda_g, \mathcal{M}_l = \mathcal{M}_g$ and $\mathcal{M}_l(\alpha_0, \Lambda_l) = \alpha_0$) so that the variance guided sampling guides the global TAA quality. However, this is not guaranteed in a real-time rendering engine. The TAA weight and functions are primarily tuned for artifacts outside of subsurface scattering. In addition, the same subsurface scattering pixel could have other contributions like overlaid transparent objects.

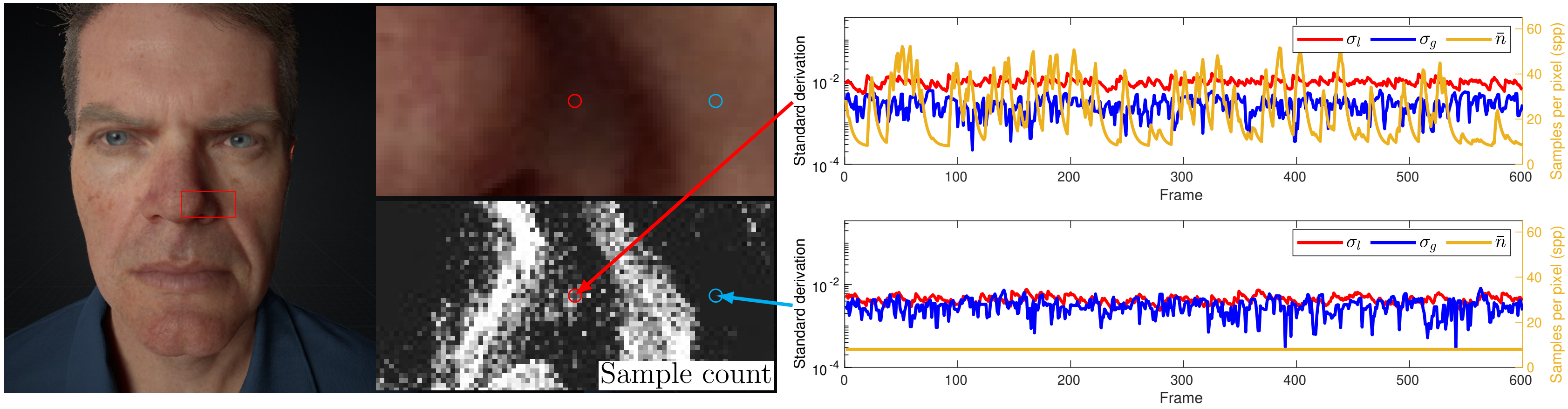


Figure 4.5: Per-frame local variance guided Screen-space Subsurface Scattering (SSS) sampling to reach a target final rendering quality. The image shows a test scene and two representative points, $P1$ (red) in shadow, and $P2$ (blue) fully lit. The graphs show per-frame local (σ_l) and global (σ_g) pixel standard derivation (SD) when $\bar{n}$ is estimated based on Eq. 4.3. Local target quality level is $\sigma_0^2 = 0.01^2$, with EMA weight $\alpha_l = 0.2$, and control factor $\kappa = 1$. The sample budget is $[b_{\min}, b_{\max}] = [8, 64]$ spp. The mean of local and global SD is $\bar{\sigma}_l = 0.0097$ and $\bar{\sigma}_g = 0.0029$ for $P1$, $\bar{\sigma}_l = 0.0044$ and $\bar{\sigma}_g = 0.0031$ for $P2$.

To make it a general technique for subsurface scattering without assumptions on the global TAA, we choose to *target a lower bound on subsurface quality* to improve the overall TAA quality. Denote $\gamma = (\mathcal{C}, \mathcal{M}, \mathcal{N})$ as a solution in the space Γ, $\gamma \in \Gamma$. The variance for a given context Λ_i at frame i of a pixel has $\sigma^2_{i,\gamma} = \varsigma^2(\gamma, \Lambda_i)$. We chose $\gamma = \gamma_0$ such that for any given sequence of length N_b:

$$\gamma_0 = \underset{\gamma_0 \neq \gamma_j, \gamma_0, \gamma_j \in \Gamma}{\operatorname{argmax}} \sum_{i=1}^{N_b} \mathbb{1}_{\sigma^2_{i,\gamma_0}} \sigma^2_{i,\gamma_j} \tag{4.13}$$

where the indicator function has $\mathbb{1}_A(B) = 1$ if $A <= B$, otherwise $\mathbb{1}_A(B) = 0$. Eq. 4.13 selects the solution that consistently produces lowest variance every frame. To achieve this, we choose $\mathcal{C}$ to be a nearest neighbor sampler without history rejection, $\mathcal{M}$ to be a max operator with $\mathcal{M}(\alpha_0, \Lambda) = \alpha_0$ and $\mathcal{N}$ as the default sample position when TAA is used (e.g., positions following a Gaussian distribution). In practice, it is sufficient to be the lower bound.

To illustrate the effect of this selection, we show a sequence of per frame local variance and global variance in Fig. 4.5 at a shadow pixel and a direct lighting pixel. We fix α_l to 0.2 (this is the maximal weight in UE4 where we implemented our algorithm) and $\sigma_0^2 = 0.01^2$. In this example, the mean global pixel standard derivation is smaller. Moreover, in high variance pixel $P1$, the average global target quality is 11.19 times better than the actual local quality. $P2$ has lower variance than the target quality, at the minimal allowed sample count of $\beta_{i-1} = 8$ spp. It results in a variance 2× better.

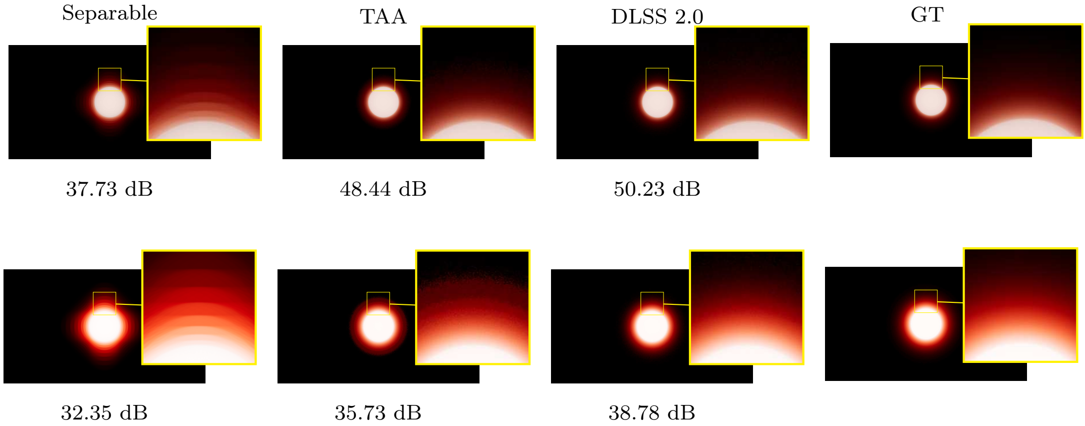


Figure 4.6: Quality of different global temporal accumulation methods (TAA and DLSS 2.0) under two different intensities. The quality of Separable is also presented. The PSNR is tested against the ground truth with 1024 spp after converting to luminance. DLSS 2.0 has higher quality. The dmfp is 10 cm.

4.5.2 Deep Learning Super Sampling (DLSS)

The ability to do local guiding allows supporting different temporal accumulation techniques other than just variants of TAA to improve the integration qualities. That accumulation algorithm can be designed for different materials as well. Recent advancement in deep learning brings in an opportunity for computer graphics and real-time rendering.

NVIDIA has recently proposed a general deep learning technique for real-time super sampling, deep learning super sampling (DLSS) [Liu, 2020]. The technique is designed to scale a low-resolution image rendered by a real-time engine to a higher resolution. In this way, GPUs with lower computing capability can play games at 4K resolution (3840×2160) with a high frame rate.

DLSS 2.0 becomes a generic technique that does not need to train for a specific game. It reuses temporal frames through an autoencoder to enhance the quality. This feature allows us to preview what could come out of specialized deep learning neural networks. Figure 4.6 shows a quality comparison between high-quality DLSS and TAA with a single spotlight perpendicularly to the surface. DLSS has a consistently higher quality than Global TAA when compared to a ground truth with 1024 spp. In the high-intensity setting, we can see the apparent energy loss due to clamping in TAA. However, the rendering from DLSS does not have this kind of energy loss. This is very interesting as DLSS is not designed with the target of temporal accumulation for subsurface scattering, and it is a general technique for upscaling. It also suggests potential quality improvement if we have a designated neural network that is targeted for subsurface scattering.

4.6 Discussion and Limitations

Our variance-guided algorithm produces the greatest speedups when sampling is most incoherent, since the bandwidth reduction matters most in those cases. Thus the speedup is best for distant sampling (close views or large mean free path), or sparse sampling. However, if the variance is very high due to the high frequency of the surrounding regions for the whole pass, there will not be any performance boost.

We also might overestimate the pixel variance, leading to more samples than needed. Because we haven't distinguished the variance due to TAA jittering from variance due to insufficient sampling. For subsurface, this might be good as everything will be diffused. Note that this biased sample count might introduce unwanted performance cost, but there is no bias in the final color.

Our final quality is bounded by the local target quality. Once met, no more samples will be added. This is reflected in the Head example err_2 in Table 7.2. Even with higher max samples, it will not improve the variance at regions that already meet the requirement. When $\kappa = 1$, the sampling contribution cannot always be correctly reflected in the final rendering. Because the weight of the final TAA limits the contribution of each frame.

In the dissertation, we choose a minimum number of 8 spp. This is because, if $b_{\min}$ is set to 1 spp, the estimated sample count series might oscillate or flicker due to noise from the low sample count and TAA history rejection. $b_{\min} = 4$ spp was observed to also reduce these flickering effects, and could be a good choice for particularly time-critical use. However, we chose $b_{\min} = 8$ spp to as sufficient to remove all TAA flickering artifacts, while still benefiting from adaptive sampling.

The target variance σ_0^2 is set by experience in our experiment. However, it can also be determined statistically from Eq. 2.28.

Chapter 5: Subsurface Scattering

> “When you are old and grey and full of sleep,
>
> And nodding by the fire, take down this book,
>
> And slowly read, and dream of the soft look
>
> Your eyes had once, and of their shadows deep;”
>
> — William Butler Yeats

In real-time applications, it is not easy to simulate realistic subsurface scattering with differing degrees of translucency. Burley’s reflectance approximation, empirically fits the diffusion profile as a whole, makes it possible to achieve realistic looking subsurface scattering for different translucent materials in screen space. However, achieving a physically correct result requires real-time Monte Carlo sampling of the analytic importance function per pixel per frame, which seems prohibitive to achieve. This chapter proposes an approximation of the importance sampling function that is suitable to be evaluated in real-time. Since subsurface scattering is more pronounced in certain regions (e.g., with light gradient change), we propose a new importance sampling technique for real-time rendering that places more samples in demanding regions, adaptive filtered importance sampling (AFIS). This algorithm minimizes sample demands with real-time adaptive sampling. It also reduces the bandwidth demands by utilizing the online generated irradiance

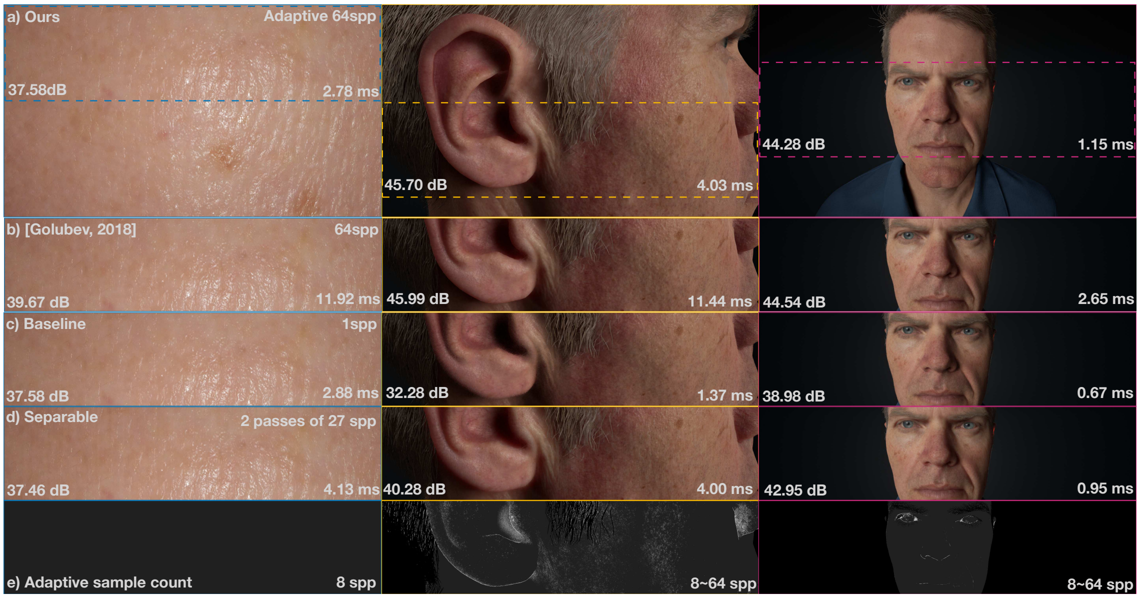


Figure 5.1: Subsurface rendering comparison from close to far at 1920 × 1080 on NVIDIA Quadro P4000 (implemented in UE4). (a) our adaptive sampling algorithm ($\sigma_0^2 = 0.001$, $\kappa = 0.2$, $b_{\mathrm{min}} = 8$ spp, $b_{\mathrm{max}} = 64$ spp), (b) Golubev [2018]'s sampling model in our framework with 64 spp, (c) a *Baseline* fixed 64-sample implementation without our proposed acceleration techniques, (d) *Separable* screen-space diffusion. (e) Visualization of our adaptive sample count for each view. Our quality is higher than Baseline in all three scenarios (close skin patch, ear, and front). Moreover, our algorithm runs faster on the close skin patch with an acceleration of up to 91.07× (2.78 ms vs 253.18 ms). In addition, our algorithm enables better quality with run time comparable or even better than Separable. Error measurements are PSNR for the subsurface, as compared to a reference image at $2K$ samples per pixel. *Digital Mike* ©Epic Games, Inc.

texture mipmaps every frame. Moreover, to enable running on GPUs with different capabilities, we propose a framework that allows users to switch different subsurface scattering acceleration techniques (i.e., AFIS and Separable) online to balance computing demands and quality. Fig. 5.1 shows the subsurface scattering result under different distance with a total acceleration up to 91.07× with AFIS. Separable can be switched on for consistent performance.

5.1 Efficient Sample Generation

In this section, we provide our approximation of the sampling function and the sampling sequence generation procedures.

5.1.1 Sampling Function

The subsurface scattering exitance radiance Eq. 2.38 can be expressed in polar coordinates as

$$\begin{aligned} L_o(p,\omega) = & \int_{\partial\mathbf{\Omega}} 2\pi r_q R(r_q) \cdot \\ & \int_{\mathbf{S^2}} CF_t(p,\omega) F_t(q,\omega_i) L_i(q,\omega_i) \langle \omega_i, n_q \rangle d\omega_i dr_q \end{aligned} \tag{5.1}$$

with the corresponding Monte Carlo estimator at p

$$L_o = \frac{1}{n} \sum_{j=1}^{n} \frac{2\pi r_{q_j} R(r_{q_j}) \cdot L_{q_j}}{pdf_{q_j}}, \tag{5.2}$$

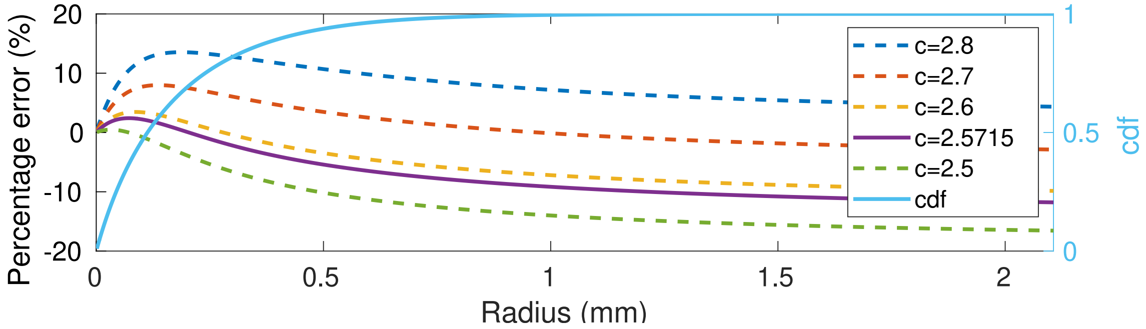


Figure 5.2: Approximation error for $cdf^{-1}(\xi)$. *Left axis)* Percentage error of $g(\xi)$ compared to $cdf^{-1}(\xi)$ for different c. $c = 2.5715$ has the minimal cost. *Right axis / light blue*) The CDF in Eq. 5.3. d is derived with $A = 0.9$ and $\ell = 1.0$.

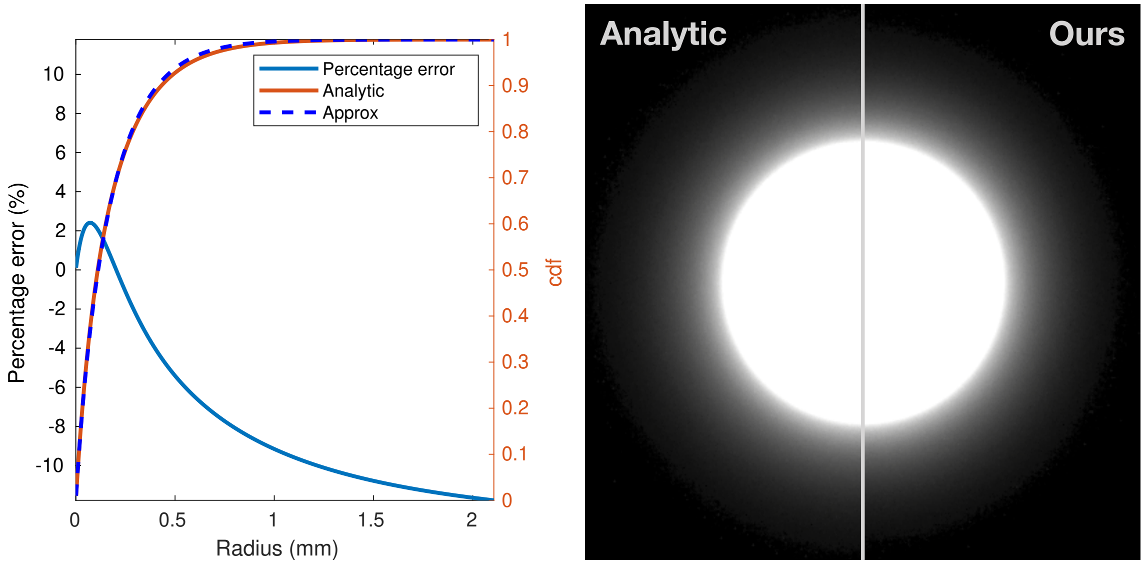
where L_o is the scattering result, r_{q_j} and pdf_{q_j} are the radius to center p and PDF of the jth sample. L_{q_j} is the accumulated diffuse irradiance at q_j. To solve this equation, apart from an efficient 2D sampling sequence (ξ_1, ξ_2), we need to importance sample the density function based on the CDF for radius sampling:

$$cdf(r) = 1 - \frac{1}{4}e^{-r/d} - \frac{3}{4}e^{-r/(3d)}. \tag{5.3}$$

Christensen [2015] suggests to use multiple importance sampling (MIS) of the two exponents, Newton iterations, or a look up table. Golubev [2019] derived an analytic inverse solution as

$$cdf^{-1}(\xi) = 3d \cdot \log(\frac{1 + G(\xi)^{-1/3} + G(\xi)^{1/3}}{4\xi}), \tag{5.4}$$

$$G(\xi) = 1 + 4\xi(2\xi + \sqrt{1 + 4\xi^2}). \tag{5.5}$$



(a) The CDF

(b) 2D profile

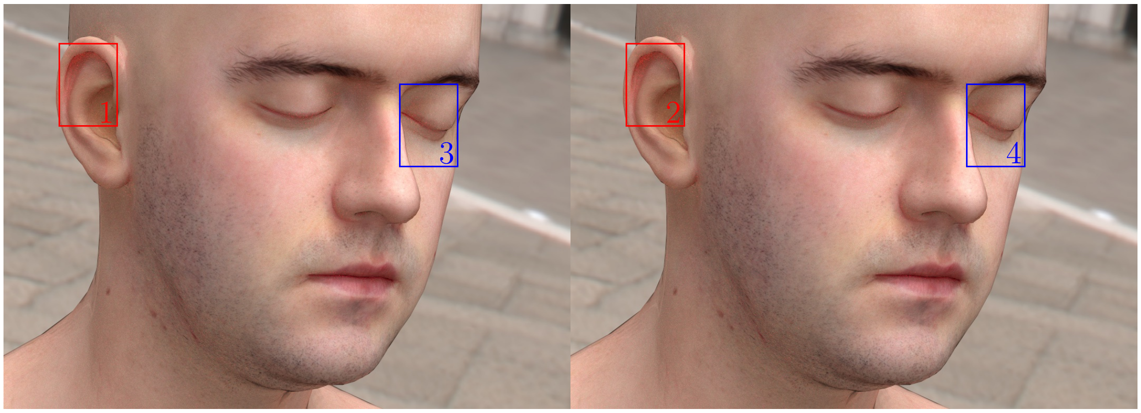
Figure 5.3: Analytic inverse vs. our approximation. Graphs of the analytic (orange) and approximate (dashed blue) are shown, along with the percentage error. The image shows that the errors do not produce significant visual differences between the analytic and our approximation for the configuration $c = 2.5715$ in Fig. 5.2 for the CDF, and 2D profile rendered in UE4.

Instead, we propose to use a simpler function $g(\xi)$ that approximates $cdf^{-1}(\xi)$ [Xie et al., 2020] as:

$$g(\xi) = d((2-c)\xi - 2)\log(1-\xi); \tag{5.6}$$

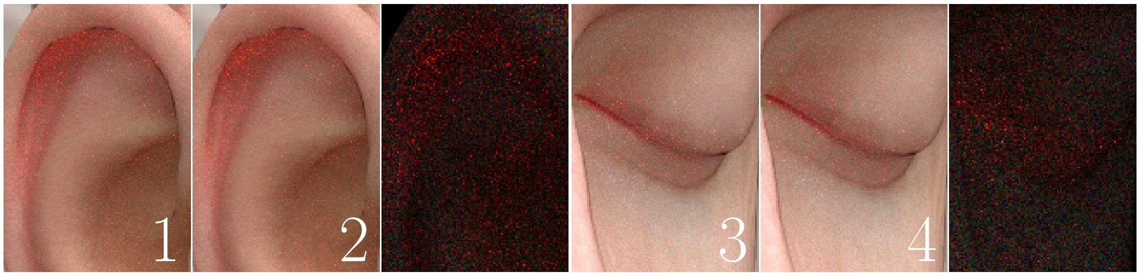
where c is a parameter to control the damping of the function. We find that when $c = 2.5715$, we have the minimal mean squared error of $cdf(g(\xi|c))$ vs. ξ as optimized using least squares nonlinear curve-fitting [Coleman and Li, 1996] at the tolerance of 10^{-6}. Fig. 5.2 shows the CDF function and the approximation error.

Since this is an approximation to the inverse CDF, samples generated with it are only a (close) approximation to the Burley PDF. To evaluate its quality, Fig. 5.3 illustrates how close the analytic inverse and our approximation are. In the beam



(a) MIS

(b) Our approximation



(c) Ear

(d) Eye

Figure 5.4: Infinite head (©Lee Perry-Smith) rendered with (a) MIS and (b) our approximation from Eq. 5.6 in PBRT. RMS error is 1.87%. We show a close comparison of high scattering region (c) ear and (d) eye with their pixel differences.

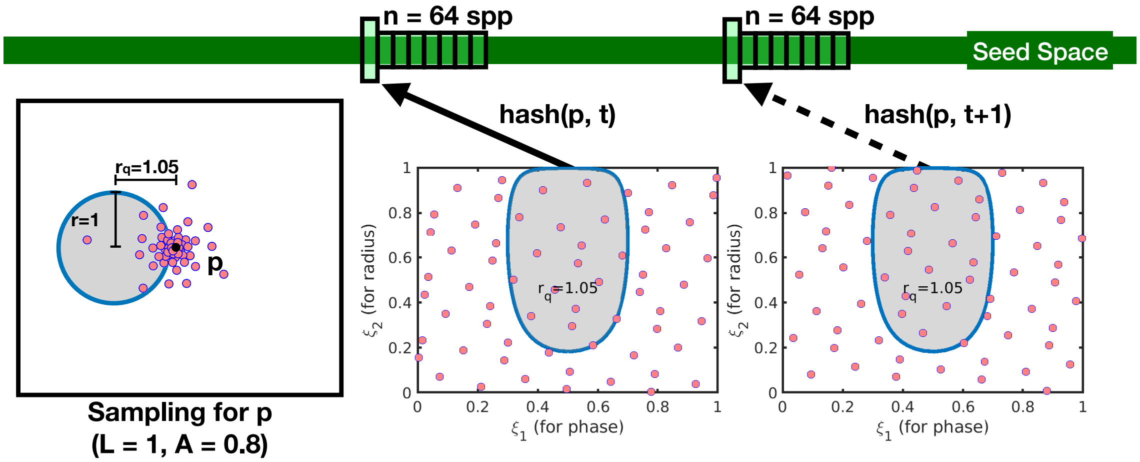


Figure 5.5: Online sampling sequence generation for real-time subsurface scattering with R_2 sequence.

light scenario (Fig. 5.3(b)), the circle light (10 flux) radius is 1 *cm*. Zooming into the figure, a tiny shrink in scattering distance can be observed, but is negligible at normal viewing distances. Further more, we implemented our approximation into PBRT [Pharr et al., 2016] and compared it with the built-in MIS method. Fig. 5.4 shows the rendering comparison within PBRT using the *head* scene. The *dmfp* parameter is derived from the *skin1* configuration in Jensen et al. [2001]. Our approximation has a small RMS error of 1.87% compared to multi-importance sampling for $2K$ spp.

5.1.2 Sampling Sequence

Fig. 5.5 shows how the sampling sequence is generated and used for subsurface scattering sampling, where the diffuse mean free path is 1.0, and the surface albedo is 0.8. We use the pixel location p and the frame time t as a hash to index into the seed space during each frame. We select the best hash function *pcg3d16* for

16-bit format from Jarzynski and Olano [2020] to hash the 3-dimensional vector to a one-dimensional seed index. With this index during each frame, we generate n continuous 2D samples by incrementing the index by 1 for each sample. We find that the 2D R_2 sequence is best for the purpose as it is generated online efficiently with a simple formula. The quality is comparable to the real-time Sobol sequence. The R_2 sequence [Roberts, 2018] is

$$t_m = m\psi, m = 1, 2, 3, ... \tag{5.7}$$

$$\psi = (\frac{1}{\phi_d}, \frac{1}{\phi_d^2}, ...\frac{1}{\phi_d^2}) \tag{5.8}$$

where ϕ_d is the unique positive root for $x^{d+1} = x + 1$, and d is the dimension of each sample.

The blue noise sampling sequence has the best-perceived quality when the sample count is low. However, the utilization in a real-time engine either requires high generation time or a large chunk of lookup table, which prevents the utilization in our application where bandwidth is critical. It might be a good direction to explore. However, it is outside the scope of this dissertation.

5.2 Adaptive Filtered Importance Sampling

In subsurface scattering, we need to efficiently get samples, e.g., at each $\mathbf{q}$, on GPU for Monte Carlo integration of the following formula

$$L_o(p) = \int R(q)\mathscr{b}(p, q)dq. \tag{5.9}$$

Instead of solving the equation with a brute force Monte Carlo estimator, we explore increasing the sampling information with pre-filtered sampling. Prefiltering has been successfully used in environment map importance sampling [Křivánek and Colbert, 2008] for efficient sampling on GPU. Its efficient sample count is higher due to the filtering steps. For example, if a sample is spent to sample a higher level of a mipmap, it is equivalent to averaging four samples from the lower MIP. We use it for screen-space scattering irradiance sampling in our application to reduce the sample demands.

From Eq. 5.9, we derived the filtered sample at $\mathbf{u}_i$ by filtering the screen-space irradiance cache $\mathscr{b}$ for subsurface scattering with respect to the center at $\mathbf{p}$ using a spatially variant pre-filter ($\mathbf{p}$ is removed for simplicity as it is not changing within the integral).

$$\begin{aligned} \mathcal{S}_{ss}(\mathbf{u}_i) &= \int \frac{R(\mathbf{q})}{p(\mathbf{q})}\mathscr{b}(\mathbf{q})[h(\mathbf{u}_i - P(\mathbf{q}))p(\mathbf{q})w(s)]ds && (5.10) \\ &\approx \frac{R(P^{-1}(\mathbf{u_i}))}{p(P^{-1}(\mathbf{u}_i))} \underbrace{\int \mathscr{b}(\mathbf{q}')[h(\mathbf{u}_i - P(\mathbf{q}'))p(\mathbf{q}')w(s)]ds}_{\text{Pre-filtered mipmap } \bar{\mathscr{b}}(\mathbf{q},l)} && (5.11) \end{aligned}$$

where $\mathbf{q}' = W(s)$. $s = W^{-1}(\mathbf{q})$ maps the point $\mathbf{q}$ in world space (mm) to screen space texture coordinate $\mathbf{s}$. The Jacobian determinant is $w(\mathbf{s})$. $\mathbf{q} = P^{-1}(\mathbf{u})$ maps the coordinate $\mathbf{u}$ in uniform sampling space to world space (mm) coordinate $\mathbf{q}$. The Jacobian determinant is $p(\mathbf{q})$. $p(\cdot)$ is the probability density function. $h(u)$ is a low-pass filter (e.g., 2×2 mean filter when creating mipmaps). The integration term can be pre-filtered using mipmaps and retrieved by a MIP level l as $\bar{b}(\mathbf{q}, l)$. By moving R/p out, it is assumed constant over the filtered region, which is a good approximation as the filtered region is small and p proportional to the PDF.

MIP level. Prefiltering has been successfully used in environmental map importance sampling [Křivánek and Colbert, 2008]. We use it for screen-space scattering irradiance sampling. Specifically, we generate the MIP level with

$$l = \frac{1}{2} \cdot \max\left(-\log_2\left(\frac{a \cdot p \cdot \hat{n}}{\ell_{\max}^2 \cdot t}\right), 0\right) \tag{5.12}$$

$$t = \frac{w}{D} \frac{w \cdot \mathit{AspectRatio}}{D} = \left(\frac{w}{D}\right)^2 \cdot \mathit{AspectRatio} \tag{5.13}$$

where $\ell_{\max}$ is the max diffuse mean free path of the three-channel subsurface profile. a is a constant factor to scale the MIP level. t is the texel size in screen space considering world unit scale w, D is the scene depth at the center sample. We find $a = \frac{1}{16}$ gives a good balance between quality and performance.

Adaptive Algorithm. This MIP level formulation enables us to perform adaptive filtered importance sampling for subsurface scattering. Namely, we perform filtered importance sampling with real-time adaptive sampling shown in Algorithm 2. Note that this algorithm is the basic form. It can be improved to handle artifacts like

Algorithm 2 Adaptive Filtered Importance Sampling

```
Require: $\mathbf{p}$(the target pixel in world coordinate for Monte Carlo integration)
$\hat{n}$ = Eq. 4.9 (Sample Count Estimation)
$L_o = 0$
for $i = 1, 2, ..., \hat{n}$ do
    $\mathbf{u}_i$ =Uniform2DSequence(i)
    $\mathbf{q}_i$ =ProjectToWorldCoord($\mathbf{u}_i$)        ▷ Base unit = $1mm$
    pdf = SamplePDF($\mathbf{q}_i$)
    $lod$ = Eq. 5.12 (Compute Mip Level)
    $B = \text{tex2Dlod}(\bar{b}, W^{-1}(\mathbf{q}_i), lod)$
    $L_o + = B \cdot \text{SampleProfile}(\mathbf{q}_i)/\text{pdf}$
end for
return $L_o/\hat{n}$
```

bleeding with bilateral filtering. The main difference AFIS has to the prefiltered importance sampling [Křivánek and Colbert, 2008] is that the sample count $\hat{n}$ is adaptively determined and can affect the MIP level selection. For example, when the surrounding lighting is smooth, AFIS leads to fewer samples and higher MIP levels, thus significantly reducing the bandwidth demands than adaptive sampling or filtered importance sampling alone.

5.3 Advanced Design

In this section, two different rendering acceleration techniques are explored for subsurface scattering. One is to further optimize the efficiency of the adaptive filtered importance sampling to monitor variance change of distant scattering only without diffuse because diffuse is already known, and the TAA algorithm (a temporal filter of neighbor pixels) will create a variance due to subpixel jittering.

Another acceleration is based on our heterogeneous real-time rendering framework to explore the heterogeneous computing space. We combine two acceleration types to perform hybrid subsurface scattering. The idea is similar to combining rasterization with a ray tracing pipeline, a hot topic in real-time rendering [Barré-Brisebois et al., 2019, Haines and Akenine-Möller, 2019]. In this way, the type of subsurface scattering can be changed at run time by the users to give important regions higher quality through AFIS or stable performance through the separable filter.

5.3.1 Unification of Scattering

Scattering decomposition. Due to the resolution of $B(p, q, \omega)$ in practice, the sampling resolution is bounded by texel size $\mathbf{t} = (w, h)$. Denote the diffuse scattering sampling radius as $r_0 = Z\sqrt{w^2 + h^2}/2$, where Z is the depth of p, we divide the region into a direct scattering (or diffuse) region $\partial\mathbf{\Omega}_d$, and distant scattering region $\partial\mathbf{\Omega}_s$, $\partial\mathbf{\Omega} = \partial\mathbf{\Omega}_d \cup \partial\mathbf{\Omega}_s$ as shown in Fig. 5.6. From Eq. 2.38 we have:

$$
\begin{aligned}
L_o(p,\omega) =& L_d(p,\omega) + L_s(p,\omega) && (5.14)\\
=& \int_{\partial\mathbf{\Omega}_d} R(r_q)\boldsymbol{b}(p,q,\omega)dq + \int_{\partial\mathbf{\Omega}_s} R(r_q)\boldsymbol{b}(p,q,\omega)dq && (5.15)\\
=& B(p,p,\omega)\cdot cdf(r_0) + \int_{\partial\mathbf{\Omega}_s} R(r_q)\boldsymbol{b}(p,q,\omega)dq && (5.16)
\end{aligned}
$$

where the first term in Eq. 5.16 is the analytic result after a change of variables from the Cartesian coordinate system to the polar coordinate system with the assumption that the covered region of direct scattering has constant lighting and it can be approximated by the discrete representation $B(p, p, \omega_o)$. Because we have

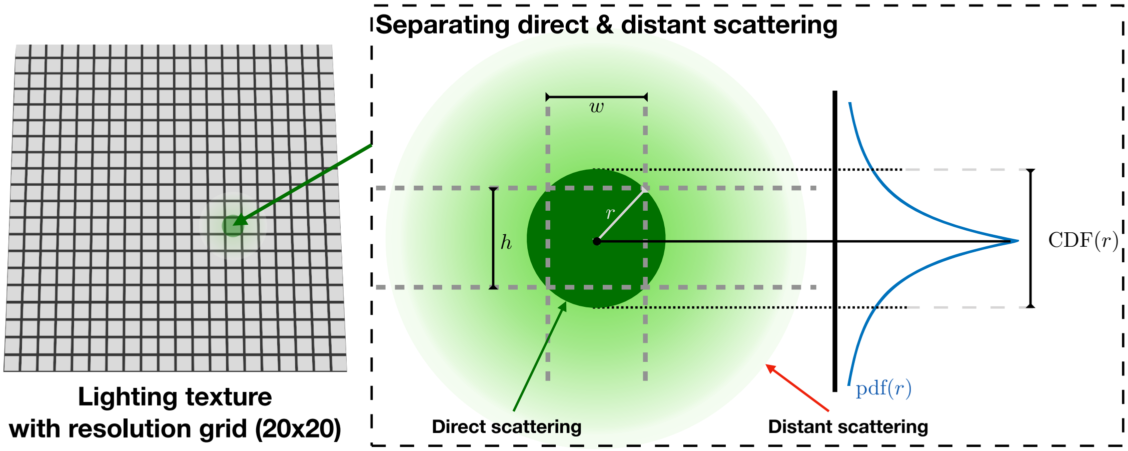


Figure 5.6: Subsurface scattering with sampling resolution.

$\int_0^{r_0} 2\pi r R(r) dr = A \cdot \mathit{cdf}(r_0)$ by Christensen and Burley [2015], where A is the surface albedo. We ignore this constant for simplicity. It can be introduced back with a direct multiplication after subsurface scattering [Xie et al., 2020]. If we deploy importance sampling for this formula with $r_q \sim \mathit{pdf}(r)$, and the corresponding cumulative density function (cdf) inverse $g(\xi) = \mathit{cdf}^{-1}(\xi)$, the numerical approximation is

$$\begin{aligned} E[L_o(p,\omega)] \approx {} & B(p,p,\omega) \cdot \mathit{cdf}(r_0) + \\ & \frac{1}{m} \sum_{i=n-m+1}^{n} \frac{2\pi r_{q_i} R(r_{q_i}) B(p,q_i,\omega)}{\mathit{pdf}(r_{q_i})} \Big(1 - \mathit{cdf}(r_0)\Big). \end{aligned} \tag{5.17}$$

where $r_{q_i} = g((1-\xi_i) \cdot \mathit{cdf}(r_0) + \xi_i)$, n is the sample count for subsurface scattering, and $m = n(1 - \mathit{cdf}(r_0))$ is the effective sample count for distant scattering. In this formulation, we directly use the pre-integrated lighting for diffuse, and weight between diffuse and scattering based on cdf.

Generalization. With Eq. 5.17, we summarize the real-time subsurface scattering model for a given ω as:

$$L_o(p) = (1-\gamma) \cdot B(p,p) + \gamma \cdot \mathcal{L}_s(p, R_F, \partial\Omega_s) \tag{5.18}$$

where γ is distant scattering energy ratio to blend the pre-integrated direct lighting and un-normalized distant subsurface scattering $\mathcal{L}_s(p, S_F, \partial\Omega_s)$, R_F is the corresponding subsurface scattering profile. Table 5.1 shows the realization of γ and R_F for different models.

Table 5.1: The realization of different subsurface scattering models.

	γ	R_F	Reference
Normalized Burley	$1 - cdf_b(r_0)$	R_b	[Burley, 2015]
Dipole	$1 - cdf_d(r_0)$	$R_{d,ss+ms}$	[Jensen et al., 2001, Mertens et al., 2003]
Pre-Integrated Separable	c	$R_{d,ms} - R_{G_0}$	[Jimenez et al., 2015]

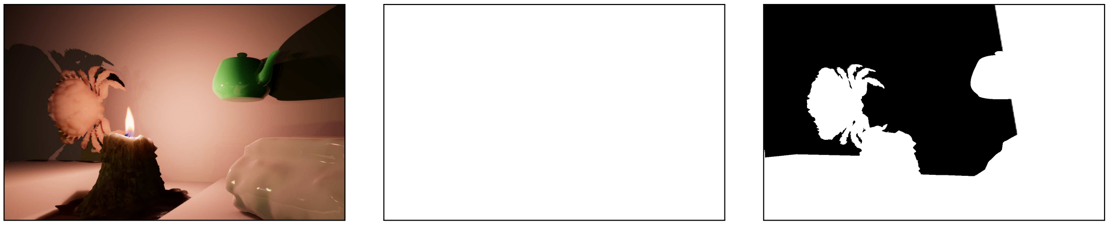
Specialization. The generalization provides us a way to reason about existing subsurface scattering models and whether they are applied in the same way for real-time rendering. For example, the artist friendly separable subsurface scattering [Jimenez et al., 2015] uses a constant blending factor c (strength) and a modified profile $\tilde{R}_d = R_{d,ms} - R_{G_0}$ where R_{G_0} is the Gaussian approximations with the minimal variance. The result is consistent with offline dipole models, only when:

$$c = \frac{(1 - cdf_d(r_0)) \cdot (B(p,p,\omega) - \mathcal{L}_s(p, R_{d,ss+ms}, \partial\Omega_s))}{B(p,p,\omega) - \mathcal{L}_s(p, \tilde{R}_d, \partial\Omega_s)} \tag{5.19}$$

Unified representation. Since γ is dependent on parameters of the CDF (e.g., diffuse mean free path for Burley), resolution and depth, we are able to enable 0 spp for distant scattering when most scattering is less or equal to one pixel as

$$\hat{m}_{(i)} = \begin{cases} 0 & \gamma < \epsilon_u \\ \\ \hat{n}_i \cdot \gamma & \text{otherwise} \end{cases} \tag{5.20}$$

where ϵ_u is a small constant to determine when distant subsurface scattering is not performed. Fig. 5.7 shows an example of scattering regions with different ϵ_u. Most of the walls do not need distant scattering when $\epsilon_u = 0.05$. Note that the estimator becomes biased due to energy loss when $\epsilon_u > 0$. As with other uses of biased estimators in rendering, this can remove unnecessary samples for distant scattering and variance tracking, but needs to be used carefully.



(a) Scene (b) $\epsilon_u = 0.0$ (c) $\epsilon_u = 0.05$

Figure 5.7: Direct/diffuse region (black) and direct+distant (white) (b,c) for scene (a). The vertical line on the wall (c) is the boundary where only 5% of scattering energy is from distant scattering.

With this formulation, instead of estimating $\hat{n}_i$ first and then calculating the sample count m_i for distant scattering, we can directly estimate $\hat{m}_i$ with Eq. 4.9 with the history tuple $\mathcal{H}_i = (\mu_i, \sigma_i^2, \bar{m}_i)$. Then the target quality σ_0^2 is set for distant scattering. This switch also implies that the variance contribution of the direct scattering due to temporal change (e.g., jittering and lighting change) has been removed. It will not affect the sample count estimation.

5.3.2 Importance-Guided Acceleration

Real-time subsurface scattering has already found its way in current generation real-time and game rendering engines before our techniques came into being. Different acceleration techniques have their own pros and cons. For example, the separable approximation has banding artifacts for close views but runs consistently fast due to coherent memory access no matter how complex the subsurface scattering material is. MC sampling has the best quality for movie presentation but is a little expensive even with our proposed adaptive filtered importance sampling. For example, the separable approximation runs slightly faster in the middle and right image in Fig. 5.1. For fullscreen subsurface scattering, even when we have temporally stable adaptive sampling through Control Variates in Fig. 6.1, the Separable approximation runs a little faster. By choosing different acceleration techniques based on quality importance concurrently, subsurface scattering would run faster and with higher perceived quality. For example, the primary character model could use adaptive filtered importance sampling, while non-focused other monsters or characters can use the Separable approximation technique. This performance and quality balance motivate us to create a coherent rendering system that uses all acceleration techniques.

This section proposes a system framework to support importance-guided acceleration, namely, to support multiple acceleration techniques (i.e., Separable approximation and adaptive filtered importance sampling) in the same frame. The users only need to switch the model based on their design requirements, which is easily achievable.

Before showing the proposed approach, the pass view with only adaptive filtered importance sampling (see Section 5.2) for subsurface scattering is illustrated in Figure 5.8. Different techniques can be combined after prefiltering and combined before the multiplication of base color. Figure 5.9 shows the overview of the new subsurface scattering pass. The sub-passes enable our flexible subsurface scattering in high quality and performance based on importance. A detailed description of different passes are as below:

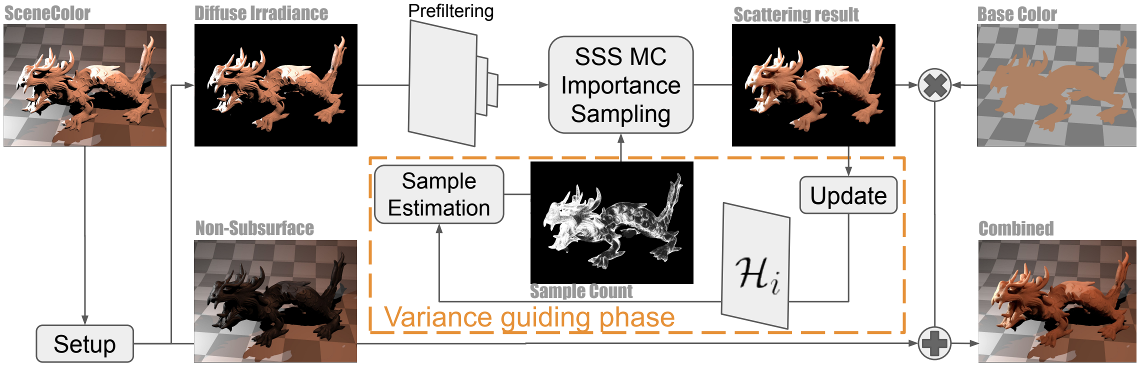


Figure 5.8: **Subsurface pass overview**. The variance guiding phase enables one pass adaptive sampling. In each frame, this pass 1) estimates the number of samples per pixel per frame and 2) updates the history $\mathcal{H}_i$. During sampling, adaptive filtered importance sampling algorithm 2 is performed in *SSS MC Importance Sampling*.

- **Setup pass**. We mask out all non-subsurface colors using a profile id and combine the diffuse with depth into a single texture to save bandwidth for primary pass sampling. Since different subsurface scattering profiles have a unique id, they are extracted separately from the specular and non-subsurface components. Those ids are gathered into different GPU buffers to support importance-guided acceleration. Specifically, two technique ids are included: Separable approximation and adaptive filtered importance sampling (AFIS).

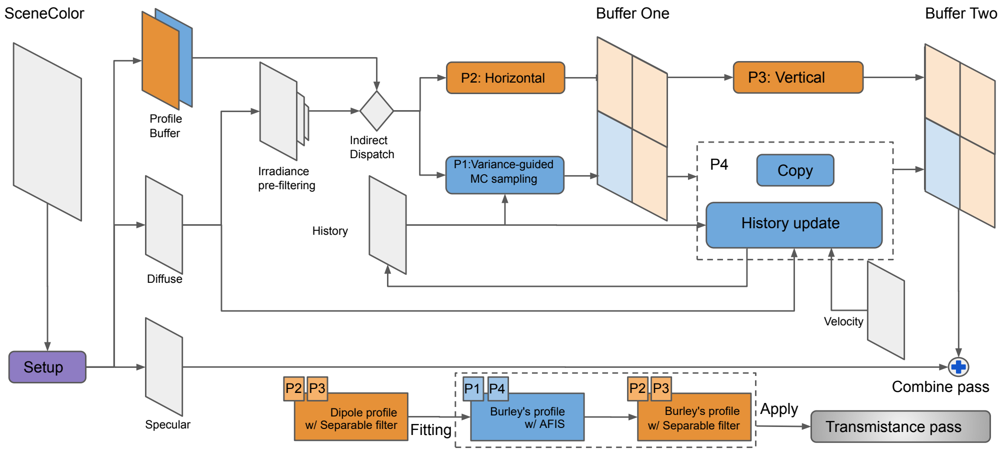


Figure 5.9: **Design overview of the new subsurface scattering pass to support importance-guided acceleration**. 1). During setup, we extract subsurface diffuse irradiance, specular, and create the profile indirect dispatch buffer. For AFIS, it runs through P1: variance-guided MC sampling and P4 that copies AFIS buffer and updates a history texture for stable variance estimation. For Separable approximation, it runs through P2 and P3 for horizontal and vertical sampling. The pass sequence is specially designed in this order to have both acceleration techniques rendered correctly. 2). Separable dipole model can be automatically upgraded to Burley with a fitting profile (shown in Appendix A).

At last, we register the technique id for each thread group and record it into two technique buffers. We then dispatch different subsurface acceleration passes. A naive method for N acceleration technique implementation is to concatenate the N technique passes. However, suppose AFIS and Separable techniques are independently implemented in three passes (i.e., one pass for AFIS and two passes for Separable filtering) and concatenated together, there will be no performance benefits supporting both techniques at the same time. Instead, we can use indirect dispatch to schedule different techniques; each computing unit only needs to call the corresponding code for different techniques. However, our case is more complicated. Separable requires a full texture integration with horizontal and vertical passes. Because of this, we cannot leave some regions untouched after the first pass (as those regions are reserved for AFIS). Otherwise, there would be dim borders between the two acceleration techniques. Moreover, AFIS requires two passes. The first pass is to perform sampling, and the second pass requires at least a copy. First, we use the AFIS result as an approximation for the first pass in the Separable filter to handle those issues. Since two passes are inevitable, we can shift some operations for AFIS to the second pass to reduce potential bandwidth. Then the four passes are executed in the order of P1 to P4, as shown in Figure 5.9.

- **Irradiance prefiltering pass**. Since we have limited budgets for MC sampling, we build a mipmap chain to prefilter the diffuse irradiance so that each MC sample can pick up more sample information. This pass is inevitable because of the filtered sampling used in AFIS to accelerate the rendering. Building mipmaps every frame seems expensive. However, it takes at most 0.17ms for 5 MIP level generation for 1080p on NVIDIA Quadro P4000, which is a low overhead. Please refer to Table 7.1 for more overhead details it takes under different configurations.

- **P1: AFIS pass**. In this pass, we call the AFIS algorithm 2 for those pixels whose technique id is marked as AFIS. We first estimate the number of samples required at each pixel to reduce the sample variance to a target noise level based on our proposed real-time adaptive sampling algorithm. We use the proposed inverse CDF approximation function to enable fast sample generation on the fly to sample from the diffuse irradiance texture. To further reduce the variance, we estimate a MIP level based on the profile configuration and retrieve diffuse irradiance. Note that the history update for real-time adaptive sampling has been shifted to P4 for the concern of potential bandwidth demands.

- **P2: Horizontal and P3: Vertical sampling pass**.The shading code for Separable filtering is called sequentially for horizontal and then vertical in these two passes. In the horizontal and vertical sampling pass, the pre-integrated weighted Gaussian kernel for the subsurface profile is used to sample the diffuse irradiance for the best performance.

- **P4: multi-functional pass**. In the multi-functional pass, We copy the AFIS sampling result to the next buffer. As a free ride, we update the history buffer of the current pixel. Since the scene will constantly change and the camera moves arbitrarily, reprojection is performed with velocity texture to get the history of the current pixel in the previous frame for the history update. Since the adaptive sampling algorithm might suffer from performance drop due to dynamic lighting, specifically, the algorithm 4 and 5 are performed for more stable adaptive sampling. Please refer to Chapter 6 for more details.

- **Combine pass**. At last, we modulate the subsurface scattering diffuse from buffer two on the base color, which is served as the surface albedo as illustrated in the pass overview for AFIS technique only in Figure 5.8. Next, we combine the result with the specular based on the subsurface profile setting.

Since the Separable approximation technique is previously for the Dipole model only, we provide a fit between Dipole and Burley's model. To achieve this, we approximated the relationship between new and old parameters with a quadratic equation. The fitting process is in Appendix A. With this fitting, we can also upgrade the transmittance to use Burley's model. In this way, all two real-time acceleration techniques can be supported at the same time. The user can switch different acceleration techniques in the game thread.

5.4 Summary

This chapter proposes a novel efficient sampling approximation function that is efficient for online evaluation, and introduced how the sample sequence is generated. We also proposed a novel algorithm, adaptive filtered importance sampling (AFIS), for photorealistic subsurface scattering. This technique not only minimizes the sample count but also increases the sampling efficiency for each sample.

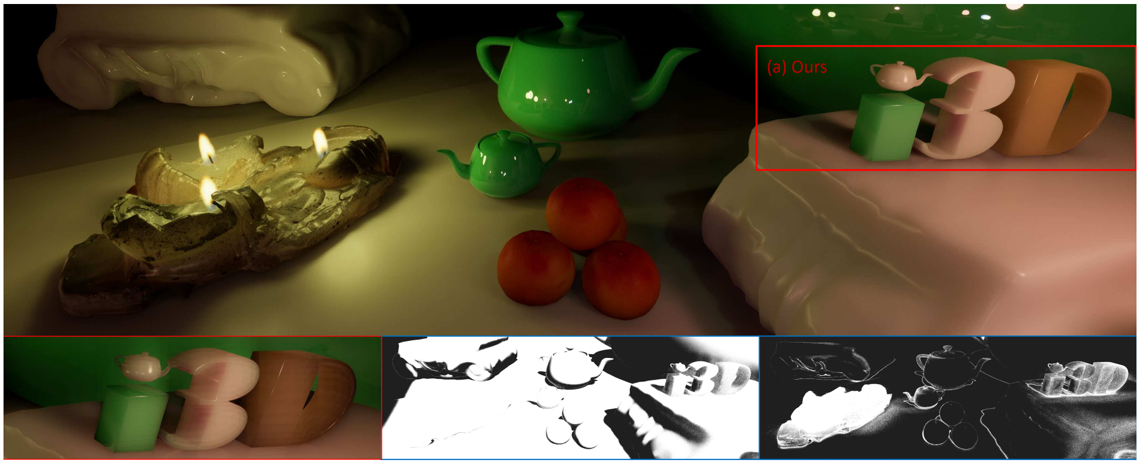
In the advanced design, we separate the diffuse and distant scattering. The variance of distant scattering is tracked, which removes the effect of diffuse variance to sample count estimation and allows 0 samples per pixel for distant scattering. At last, an acceleration framework that can be used to balance quality and performance is proposed.

Chapter 6: Real-time Control Variates

> "*Nothing puzzles me more than time and space; and yet nothing puzzles me less, for I never think about them.*"
>
> —Charles Lamb

Real-time adaptive sampling adaptively places samples based on temporal variance tracking. However, if the temporal signal has high variance, it will lead to high variance that should not be used for adaptive sampling to reduce spatial variance. The occasional performance drop due to temporal lighting dynamics (e.g., gunshots, explosions or lights turning on and off) could hinder adoption in games or other applications where a smooth high frame rate is preferred.

This chapter explores a novel usage of Control Variates (CV) in the sample domain instead of the shading domain to maintain a consistent low pass time. To achieve this, a novel joint-optimization algorithm for sample count and CV coefficient estimation is proposed. The major enabler is our novel time-variant covariance updating method that helps remove the effect of recent temporal dynamics from variance tracking. Since bandwidth is critical in real-time rendering, a solution without adding any extra textures in the adaptive sampling framework is also provided. This chapter is extended from our I3D conference (PACMCGIT) publication [Xie



(b) Separable [Jimenez et al. 2015] (c) SPVG [Xie et al. 2020] (d) Ours

Figure 6.1: Dynamic subsurface scene just after light has been turned off. Our method has consistently lower sample count (d) than SPVG [Xie et al., 2020] (c) at this frame. It leads to lower sampling pass time in dynamic lighting from 12.9 ms to 5.2 ms at 3360 × 1440 (×2.5), while maintaining good quality (47.5 dB) vs SPVG (48.6 dB). Separable (b) runs fastest for the whole subsurface pass at 4.0 ms, however, with visible banding artifacts.

and Olano, 2021].

6.1 Introduction

Real-time adaptive sampling proposed in Chapter 4 is a general technique, and has been used for subsurface scattering with importance sampling of Burley's normalized diffusion profile [Xie et al., 2020]. It has been adopted in real-time rendering engines (e.g., Unreal Engine 4). Its low time and space complexity $O(1)$ increases the sample efficiency of Monte Carlo algorithms for real-time rendering. The basic idea is to use sample histories in the shading domain to estimate the spatial Monte Carlo sampling variance and adjust the sample count to minimize the sample count for a target variance level in real time.

However, if high temporal variance is present after history projection, it will also increase the sample count and pass time to achieve convergence. Real-time applications like games are intrinsically dynamic, including i) gunshots/lightning creating rapid incoming radiance changes, ii) dynamic light particles changing intensities, and iii) moving objects creating dynamic shadowing.

In this chapter, we propose a novel technique, *real-time subsurface control variates*, to address the issue. Control variates is a variance reduction technique for Monte Carlo sampling. If we can find a good approximation of the sampling function, with the optimal CV coefficient, the variance after MC sampling can be reduced given the same number of samples. In other words, given a target variance, the sample count can be minimized. Note that the application of CV in this paper is a little different from the typical use. We only reduce the monitored change in variance caused by the lighting changes as shown in Fig. 6.1, but keep the same variance as the MC estimator when the lighting is static because the monitored variance controls the sample count estimation [Xie et al., 2020]. In this dissertation, we focus on real-time subsurface scattering. To have real-time subsurface scattering anytime and anywhere efficiently using control variates, the following three challenges are addressed:

1) Evaluate the time-variant covariance matrix. Calculating the optimal CV coefficient online requires a covariance matrix that captures the recent spatial correlations between the sampling function and the control for time-variant scenarios like dynamic lighting, yet without temporal variance. We propose a novel covariance estimation based on the CV residual, the *exponential moving covariance matrix*. We compare this to the widely adopted exponential weighted moving average (EWMA) covariance estimator [Guldimann et al., 1995].

2) Compute the optimal CV coefficient online. We provide online joint estimation algorithms to find the sample count and the CV coefficient numerically. Since the estimated coefficient is dependent on the control variable with unknown distribution, the rendering result based on the control variates might be biased [Lavenberg et al., 1982]. To avoid this issue, we shift the application of CV from the shading domain to the sample domain, where the CV guides sample count estimation. In this way, we also shift potential bias from the shading domain to the sample domain as biased sample count. Since our online covariance computation assures sample count overestimation, the final subsurface scattering shading result is still unbiased, though possibly using more than the absolute minimal number of samples. Nevertheless, the rendering time for subsurface scattering is reduced up to 3.11× during dynamic lighting tests.

3) Lightweight CV coefficient. Since online optimal CV coefficient estimation still requires an additional texture and real-time computation, this might violate the memory budget for time critical applications. Under such scenarios, we also provide a lightweight offline approximation for the optimal CV coefficient without adding any additional textures.

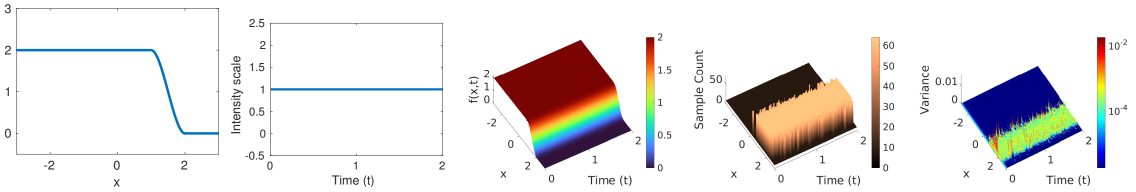
The major contribution is summarized as below:

- Derived exponential moving covariance matrix as an extension to the largely adopted EWMA covariance estimator.

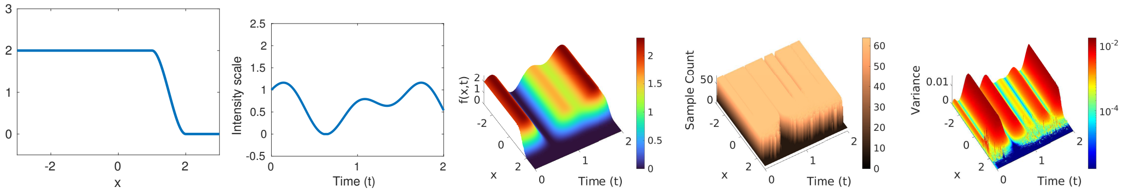
- An online sample count estimation algorithm jointly estimated with optimal

CV coefficient estimation to remove temporal influence.

- A lightweight CV without adding any extra textures.



(a) Lighting spatial profile (b) Lighting temporal profile (c) Spatial & temporal result (d) Spatial & Sample count (e) Spatial & Variance



(f) Lighting spatial profile (g) Lighting temporal profile (h) Spatial & temporal result (i) Sample count (j) Variance

Figure 6.2: Temporal instability leads to sample count over-estimation with real-time adaptive sampling. Diffuse lighting on a 1D surface is shown in (a) and (f). During subsurface scattering sampling, when lighting is temporally stable (b), the estimated sample count (d) leads to lighting (c) with a variance close to target variance 10^{-4} (e). When lighting is not stable (g), although we can get high quality lighting (h), $rmse = 0.0028$, the sample count(i) is over estimated because of temporal variance in (j). Temporal instability induces 242% (i) as many samples as that shown in (d) on average in this example. The induced calculation and bandwidth demands might threaten the performance with real-time adaptive sampling.

6.2 Motivating Example

In real-time adaptive sampling, there are two major contributions to the variance: spatial variance due to MC sampling, and temporal variance due to lighting changes.

Spatial variance. Fig. 6.2(a)–(e) shows a 1D example of spatial variance with stable lighting over 2 seconds. For the lighting gradient change region around $x \in [1, 2]$, the sample count is estimated to the max budget allowed at 64 spp to minimize the variance to $\sigma_0^2 = 10^{-4}$ as shown in Fig. 6.2(e).

Temporal variance. Dynamic lighting introduces *temporal variance*. For example, it could be caused by gunshots, dynamic lighting particles, or moving objects that occlude/dis-occlude lighting randomly for a shading point. This variance is present as we use temporal histories. It leads to sample count overestimation. We demonstrate it in the same setup in Fig. 6.2(f)–(j), lighting intensity changes over time is shown in Fig. 6.2(g). It leads to high sample count Fig. 6.2(i) across space to reduce the high variance Fig. 6.2(j) induced by temporal lighting changes. It leads to sample count increase by 242% in this example. It is critical to mitigate the effect of dynamic lighting changes for real-time performance.

6.3 Control Variates

In the rendering literature, control variates (CV) has been used to increase Monte Carlo (MC) rendering efficiency. The basic idea is to modify the original function with a known integral corrected by an coefficient over a spatial domain (e.g., 2D surface or 3D Volume)

$$F(x) = a(x) \cdot G(x) + \int_{y \in \mathcal{D}} f(x,y) - a(x) \cdot g(x,y) dy, \tag{6.1}$$

where $G(x) = \int_{y \in \mathcal{D}} g(x,y)dy$ and the optimal CV coefficient has $a = Cov(f,g)/Var(g)$ [Lavenberg et al., 1982]. In this chapter, we deal with temporal change for sample estimation. Instead of working in the space domain, we extend the concept to the time domain to address time-variant instability. This is slightly different from prior techniques that minimize MC estimator variance in static scenes. The modified formula is:

$$F(x,t) = a(x,t) \cdot G(x,t) + \int_{y \in \mathcal{D}} f(x,y,t) - a(x,t) \cdot g(x,y,t) dy. \tag{6.2}$$

In non-real-time rendering, time can be regarded as another dimension. If $x \in \mathbb{R}^n$, then $(x,t) \in \mathbb{R}^{n+1}$. However, in real-time rendering only limited history can be accessed at t for performance and storage, and no forward time evaluation is possible. In real-time adaptive sampling, only one history texture is utilized, we explore in the same spirit to make control variates possible in real-time rendering.

We use CV to reduce the influence of temporal variance. Starting from Eq. 6.2 and 5.16, we introduce the control variable at time t for position p as $g(q,t)$ where p is neglected for simplicity. Then we have time dimension in distant subsurface scattering as

$$L_s(t) = a(t) \cdot G(t) + \underbrace{\int_{\partial \boldsymbol{\Omega}_s} R(r_q) \boldsymbol{b}(q,t) - a(t) \cdot g(q,t) dq}_{Res(t)} . \tag{6.3}$$

Denote $f(q,t) = R(r_q)\boldsymbol{b}(q,t)$, and suppose we have a realization of $g(q,t)$ whose known integral is $G(t)$, which captures most of the temporal varying component. The optimal coefficient can be estimated with $a^* = Cov(f,g)/Var(g)$ after sampling using the m_t samples (Table 6.1 shows some selected symbols). However, we might not have enough samples in one frame to get a good estimation. Instead, we use the covariance of the two batch means to estimate the CV coefficient and use history to improve the estimation, because we have $Cov(X,Y) = nCov(\bar{X},\bar{Y})$ (see Appendix B.2). It leads to $a^* = Cov(\bar{X}, \bar{Y})/Var(\bar{Y})$. This enables us to estimate the coefficient with temporal observations over time with online covariance estimation instead of just in a single frame. Then, the actual sample count could be estimated based on the residual component $Res(t)$ with minimal temporal variance.

6.4 Theory

Before running into the detail of the online updating algorithm, we provide a simplified theory to guide the algorithm design. We assume that the temporal change component can be independently separated out from the function to integrate $f(q,t)$ and the control variable $g(q,t)$.

Namely, we have three random variables T, F, and G. TF is the function to estimate, TG is the control variable. Then the general optimal CV coefficient is

$$a^* = \frac{Cov(TF, TG)}{Var(TG)}. \tag{6.4}$$

When T is independent from F and G, the variance for $\langle TF \rangle$ and the residual variance are

$$Var(\langle TF \rangle) = Var(T(F - aG)) + Var(aTG) + 2Cov(T(F - aG), aTG), \tag{6.5}$$

$$Var(T(F - aG)) = E[T]^2 \, Var(F - aG) + Var(T) \, Var(F - aG) + Var(T) E[F - aG]^2 \tag{6.6}$$

where $\langle TF \rangle$ is an unbiased estimator. The goal in this paper is to reduce the variance contribution of T to $Var(\langle TF \rangle)$ during variance tracking. For distant subsurface scattering as illustrated in Eq. 6.3 when the lighting intensity $T = I(t)$ is independent from the scattering function F and the control variable G, we have

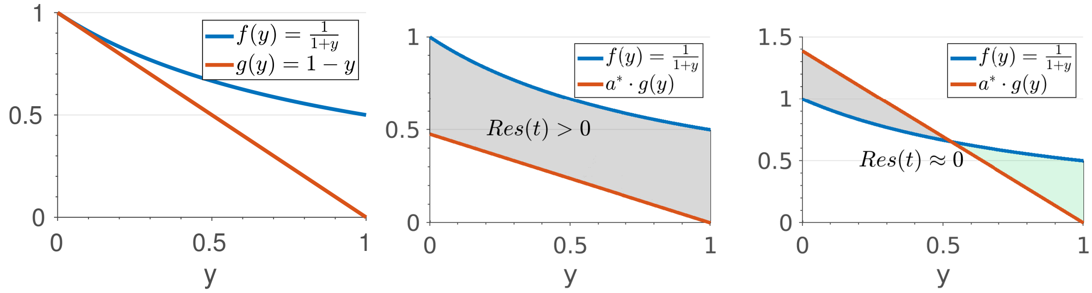
$$F = \frac{L_s(t_r)}{I(t_r)} = \int_{\partial \mathbf{\Omega}_s} R(r_q) \frac{\text{б}(q, t_r)}{I(t_r)} dq, \qquad G = \frac{G(t_r)}{I(t_r)} = \int_{\partial \mathbf{\Omega}_s} \frac{g(q, t_r)}{I(t_r)} dq, \tag{6.7}$$

$$TF = L_s(t) = I(t) \frac{L_s(t_r)}{I(t_r)}, \qquad TG = G(t) = I(t) \frac{G(t_r)}{I(t_r)}, \tag{6.8}$$

where $I(t_r)$ is a reference intensity $I(t_r) \neq 0$ at time t_r. This assumption holds when the intensity of all lights (e.g., point light, directional light and spotlight) are controlled by a single intensity parameter $I(t)$. Because we have a linear relationship between the intensity of incoming lights and the exitance radiance as shown in Eq. 2.38 and Eq. 2.39. Although $g(q, t)$ is unknown, it does not affect the reasoning in this section. Please refer to Section 6.5 for a concrete realization.

6.4.1 In-frame Standard Control Variable

In standard CV, CV requires Monte Carlo sampling and $E[TG]$ is expected to be known. Since T is independent, we have $E[TG] = E[T]E[G]$. However, $E[T]$ is unknown. Even if our proposed algorithm can estimate $E[T]$ to some extent, the MC sampling of G brings in variance, which makes the estimation more vulnerable. Fig. 6.3 illustrates the basic idea by using our CV coefficient updating algorithm. The temporal dynamics is introduced by a sine function . It demonstrates an ability to adjust a under both static and dynamic lighting (see Appendix B.3).



(a) f and g (b) $a^* \approx 0.477$ (c) $a^* \approx 1.388$

Figure 6.3: Illustration of our novel application of CV. For function f and g (a), a standard CV coefficient estimation results in (b) with a residual vulnerable to temporal variance. However our method can find the coefficient that leads to (b) during time-invariant scenarios, but to (c) during time-variant scenarios, where the residual is near zero. The residual variance is less vulnerable to temporal change.

Even worse, as TG and TF need to be sampled in a correlated way. It often means the bandwidth demand doubles in real-time rendering. Monte Carlo sampling leads to further cache incoherence. This is against the idea of real-time adaptive sampling, where bandwidth demand is minimized by reducing sample count with extremely low overhead. Because of this, applying standard CV to maintain stability for real-time adaptive sampling does not seem to be an attractable feature even when we have demonstrated some capability in Fig. 6.3.

6.4.2 In-frame Constant Control Variable

To deal with the bandwidth demand hazard, a known in-frame constant can be used as the control variable. Namely, $E[G]$ is constant with $Var(G) = 0$. This only adds a low overhead as one texture fetch. Under this condition, however, we do have a valid best CV coefficient derived from Eq. 6.4 as:

$$a^* = \frac{E[F]}{E[G]} = \frac{E[TF]}{E[TG]}. \tag{6.9}$$

With this formula, the optimal CV coefficient for the example in Fig. 6.3 has $a^* = \frac{\int_0^1 \frac{1}{1+y} dy}{\int_0^1 1-y dy} = ln(4) \approx 1.386$, and $E[F - a^*G] = 0$. Then Eq. 6.5 and Eq. 6.6 simplify to

$$Var(\langle TF \rangle) = Var(T(F - a^*G)) + Var(T)E[F]^2, \tag{6.10}$$

$$Var(T(F - a^*G)) = (Var(T) + E[T]^2)\, Var(F). \tag{6.11}$$

This analytic CV coefficient is the reason why the CV coefficient in Fig. 6.3 under dynamic lighting is approximately 1.386. When $(F - aG) \to 0$, the right term diminishes in both Eq. 6.5 and Eq. 6.6, leaving only one controllable variance $Var(F)$ in the residual variance that should be considered for variance tracking. More specifically for subsurface scattering with Eq. 6.7 and Eq. 6.8, this optimal CV coefficient is

$$a^*(t) = \frac{E[L_s(t_r)]}{E[G(t_r)]} = \frac{E[I(t)L_s(t_r)]}{E[I(t)G(t_r)]} = \frac{E[L_s(t)]}{G(t)} \tag{6.12}$$

where $G(t)$ is constant at frame t and $I(t) \neq 0$. Then, our online algorithm for subsurface scattering is designed with the following guidance:

1. Use analytic optimal CV from Eq. 6.9 for regions where intensity dynamics T

Table 6.1: Selected Symbols

Symbol	Description
$\mathcal{H}$	Real-time adaptive sampling history
α	Exponential moving coefficient
σ_0^2	Target variance level for adaptive sampling
n	Sample count for total scattering
m	Sample count for distant scattering
$\mathcal{J}$	Control Variates history
a	Control Variates coefficient
$g(t)$	Control variable at time t
$G(t)$	Integration of $g(t)$, a known constant value at time t that varies according to time.
$R(r)$	Diffuse reflectance profile
$\mathscr{b}(p, q, \omega_o)$	Lighting contribution at p from q for direction ω_o
$\mathscr{b}(t)$	Lighting contribution at p from p at time t
$\mathscr{B}(p, q, \omega_o)$	Discrete pre-integrated lighting texture for $\mathscr{b}(p, q, \omega_o)$
$\mathscr{B}(t)$	Pre-integrated lighting at texture point p at time t
γ	Distant scattering energy ratio

is independent from F and G.

2. Adaptively switch to use online estimation of Eq. 6.4 to estimate the CV coefficient when the assumption does not hold.

6.5 Online Solution

In this section, a novel online covariance estimation method is first introduced. We then provide our online CV coefficient and sample count joint estimation algorithm. To provide a concrete example for a^* estimation that separates out time-variant signal, we select one reasonable realization of f, g as

$$f(q,t) = R(r_q)\mathscr{b}(q,t), \qquad \bar{f} \approx \gamma \mathcal{L}_s(t), \tag{6.13}$$

$$g(q,t) = R(r_q)\mathscr{b}(t), \qquad \bar{g} = G(t) = \gamma B(t). \tag{6.14}$$

Note that $\bar{f}$ and $\bar{g}$ are the MC sampling result of the integration at time t. They are not necessarily equal to the analytic integration if the control variable requires Monte Carlo sampling. However, we use in-frame constant control variable to maintain temporal stability and deal with the bandwidth hazard. Therefore, $\bar{g} = G(t)$. $\mathscr{b}(t)$ is the lighting contribution at p itself at time t. It leads to a constant integration of $\gamma B(t)$, where $B(t)$ is a texture fetch directly at texel p in the pre-integrated lighting texture. Since the algorithm will run for each p, we ignore p for simplicity. Note that this fetch can also be queried from different level of details to add in the correlation of temporal intensity change of surrounding lighting for further optimization. However, it is outside the scope in this paper. With this realization in large flat lighting region, Eq. 6.12 leads to

$$a^* = 1, \tag{6.15}$$

the *main part separation*, and we have zero variance during dynamic lighting.

6.5.1 Online Covariance

Covariance is a fundamental concept in computational statistics and has great applications in many fields. When memory is limited, it is critical to have a single pass online algorithm. Welford [1962] proposed an online single pass algorithm to calculate the overall covariance numerically when each value is equally weighted. In a real-time time-variant system, weighted covariance that favors latest results are used to track temporal changes. Two notable examples are the prediction error covariance matrix update in the Kalman filter [Welch et al., 1995] with varying weights, which is frequently used in control systems, and the EWMA covariance estimator [Guldimann et al., 1995, Tsay, 2005] in finance with constant weights. In this chapter, we exploit the boundary of second case. We propose an online exponential moving covariance matrix that is mathematically derived from the weighted covariance matrix with inspiration from exponential moving variance [Finch, 2009]. We also provide its relationship to the well-known EWMA covariance estimator. Moreover, we demonstrate how temporal covariance can be removed during the monitoring.

6.5.2 Exponential Moving Covariance (EMC)

For two random variable X, Y that are incrementally observed according to time as $\{x_0, x_1 ..., x_t\}$ and $\{y_0, y_1, ..., y_t\}$ with a constant weight of α, the exponential moving covariance between them at time t $(t \geq 1)$ is:

$$\begin{aligned} Cov_t(X,Y) =&(1-\alpha)Cov_{t-1}(X,Y)+ \\ &\alpha(1-\alpha)(x_t-\mu_{t-1})(y_t-\upsilon_{t-1}) \end{aligned} \tag{6.16}$$

where x_t, y_t are the current observation, $\mu_{t-1}, \upsilon_{t-1}$ are the corresponding exponential moving average at $t-1$, and $Cov_0(X,Y) = 0$. Since there is no reference in the scientific literature, we provide a detailed proof in Appendix B.3. The new covariance formulation enables direct calculation of CV coefficient as $a_t(X,Y) = Cov_t(X,Y)/Var_t(Y)$. Then, we can derive a more generalized matrix form.

6.5.3 Exponential Moving Covariance Matrix (EMCM)

If we have a time series vector $\mathbf{Z}_t \in \mathbb{R}^{n\times 1}$ with the exponential moving average at time t as $\boldsymbol{\zeta}_t \in \mathbb{R}^{n\times 1}$, then the covariance matrix $\mathbf{\Sigma}_t (t \geq 1)$ is

$$\mathbf{\Sigma}_t = (1-\alpha)\mathbf{\Sigma}_{t-1} + \alpha(1-\alpha)(\mathbf{Z}_t - \boldsymbol{\zeta}_{t-1})(\mathbf{Z}_t - \boldsymbol{\zeta}_{t-1})^T \tag{6.17}$$

where $\mathbf{\Sigma}_0 = \mathbf{0}^{n\times n}$. This is different from EWMA covariance estimator used in stock analysis [Guldimann et al., 1995] as

$$\tilde{\mathbf{\Sigma}}_t = (1-\alpha)\tilde{\mathbf{\Sigma}}_{t-1} + \alpha(\mathbf{Z}_t - \boldsymbol{\zeta}_{t-1})(\mathbf{Z}_t - \boldsymbol{\zeta}_{t-1})^T. \tag{6.18}$$

The estimator uses the previous history at $t-1$ to predict the value at t, solving for the estimated covariance between X, Y:

$$\widetilde{Cov}_t(X,Y) \approx E_t((X-\mu_{t-1})(Y-\upsilon_{t-1})) \tag{6.19}$$

where $E_t(\cdot)$ calculates the exponential weighted average at t. While the equation we resolve is to calculate the covariance as

$$Cov_t(X,Y) = E_t((X-\mu_t)(Y-\upsilon_t)) \tag{6.20}$$

(see Appendix B.1.3). For rapid changing frames with dynamic lighting, it is preferable to have a direct calculation instead of prediction to get the CV coefficient. Then, the coefficient matrix is $\mathbf{A}_t = diag(\sum_1^n(\mathbf{e}_i^T\mathbf{\Sigma}_t\mathbf{e}_i)\mathbf{e}_i)^{-1}\mathbf{\Sigma}_t$ where $\mathbf{e}_i$ is the ith matrix basis, and $diag(\cdot)$ creates the diagonal matrix from a vector.

6.5.4 Coefficient Boundary

Since the numerical estimation might lead to instability (e.g., oscillating larger or being NaN), we bound the range of CV coefficient for subsurface scattering with two considerations:

1. Both diffuse, B, and distant scattering results are non-negative, the maximum residual cannot be larger than the distant scattering result $\bar{f}$, thus $Res_{\max} \leq \bar{f}$.
2. Residual can be negative, however, we anticipate that the minimal residual can be raised to non-negative by $G(t)$ as $Res_{\min} + a_0 \cdot \bar{g} \geq 0$ where a_0 is a constant

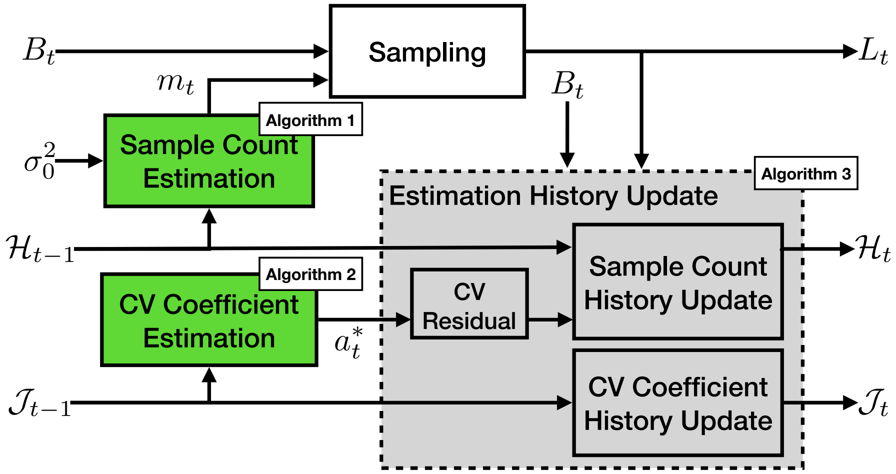


Figure 6.4: The online control variates based adaptive sampling diagram at frame time t. With sample count history $\mathcal{H}_{t-1}$ and CV history $\mathcal{J}_{t-1}$, we estimate the sample count required at time t based on the control variates residual, trying to meet the target variance level σ_0^2.

positive clamping coefficient. Thus, we have $Res_{\min} \geq -a_0 \cdot \bar{g}$.

With $Res = \bar{f} - a \cdot \bar{g}$ we have the bound $D \in [0, a_0 + \bar{f}/\bar{g}]$. If a variable a is clamped by D, we denote it as $a|_D$. Since $Cov_t(\bar{f}, \bar{g})$ and $Var_t(\bar{g})$ can be zero for hard shadows. To deal with this issue, we added a small constant factor ϵ_a as

$$a_t = \frac{Cov_t(\bar{f}, \bar{g}) + \epsilon_a}{Var_t(\bar{g}) + \epsilon_a}. \tag{6.21}$$

In this way, when both variables become zero, it could simplify to *major part separation* ($a_t = 1$).

6.5.5 Online Joint Estimation Algorithm

With the ability to calculate covariance matrix online, the online algorithm to estimate sample count m_t as well as the optimal CV coefficient a_t is provided in this section. Fig. 6.4 shows an overview of the online control variates based adaptive sampling

sampling diagram. It composes of three major parts: i) Sample count estimation, ii) CV coefficient estimation, and iii) Estimation history update.

Algorithm 3 Sample Count Estimation

```
Require: $\mathcal{H}_{t-1}, \gamma, \epsilon_u, \beta_{\min}, \beta_{\max}$
1: if $\gamma < \epsilon_u$ then
2:     $m_t = 0$
3: else
4:     Update $\hat{m}_t$ with Eq. 4.9
5:     $m_t = \hat{m}_t|_{[\beta_{\min}, \beta_{\max}]}$
6: end if
7: return $m_t$
```

Sample count estimation. As shown in Algorithm 3. if most contributions come from direct scattering (Eq. 5.20), there is no need to perform distant scattering (Line 1-2). Otherwise, it will use the real-time adaptive sampling algorithm to estimate sample count in frame t (Line 4). To have adequate observations and also consider the computing capability, the estimated sample count is restricted within $[\beta_{\min}, \beta_{\max}]$ (Line 5).

Algorithm 4 CV Coefficient a_t^* Estimation

```
Require: $\mathbf{\Sigma}_{t-1}, \epsilon_a, D$
1: $a_{t-1} = \frac{\mathbf{\Sigma}_{t-1}.xy+\epsilon_a}{\mathbf{\Sigma}_{t-1}.yy+\epsilon_a}|_D$
2: return $a_{t-1}$
```

CV coefficient estimation. With the proposed exponential moving covariance matrix, $\mathbf{\Sigma}_t = \begin{pmatrix} Var_t(\bar{f}) & Cov_t(\bar{f},\bar{g}) \\ Cov_t(\bar{g},\bar{f}) & Var_t(\bar{g}) \end{pmatrix}$. We can easily calculate the CV coefficient at $t-1$ as $a_{t-1} = (\Sigma_{t-1}.yx + \epsilon_a)/(\Sigma_{t-1}.yy + \epsilon_a)$ (Line 1 in Algorithm 4) and make it bounded by D to deal with instability. At last, the coefficient solution is approximated by the coefficient at $t-1$ as a_{t-1} (Line 2). Note that if we use EWMA covariance estimator (Eq. 6.18), the coefficient would be approximated by covariance estimator at $t-1$ using history from $t-2$. Another potential solution for CV coefficient estimation is to calculate per-frame CV coefficient directly and use exponential moving average

(EMA) for a good estimation as Fig. 6.3(b). However, it cannot lead to Fig. 6.3(c) to remove the temporal variance.

Algorithm 5 Estimation History Update

```
Require: $\mathcal{H}_{t-1}, m_t, \mathcal{J}_{t-1} = (\boldsymbol{\Sigma}_{t-1}, \boldsymbol{\zeta}_{n-1}), a_t^*, t$
1: // Sample count history update
2:     $\mathcal{S}(t) = Res(t)$
3:     Update $\mathcal{H}_t = (\mu_t, \sigma_t^2, \bar{m}_t)$ based on Eq. 4.2–4.4
4: // CV coefficient history update
5:     $\mathbf{Z}_t = [\bar{f}, \bar{g}]$
6:     Update $\boldsymbol{\Sigma}_t$with Eq. 6.17
7:     $\boldsymbol{\zeta}_t = (1 - \alpha)\boldsymbol{\zeta}_{t-1} + \alpha \mathbf{Z}_t$
8:     $\mathcal{J}_t = (\boldsymbol{\Sigma}_t, \zeta_t)$
9: return $(\mathcal{H}_t, \mathcal{J}_t)$
```

Estimation history update. After sampling, we need to update the history buffer for both sample estimation and VC coefficient estimation: *1) Sample count estimation.* After applying the control variates, the value we monitor is $Res(t)$ instead of $L_o(t)$. Because $L_o(t)$ contains both spatial and temporal variance, what represents the spatial variance most is $Res(t)$. So $Res(t)$ is set as the shading result for history update (Line 1). For storage efficiency, only luminance is used for $Res(t)$. *2) CV coefficient estimation.* The exponential moving covariance matrix and the exponential moving average for $\bar{f}$ and $\bar{g}$ are updated between line 5-7. Note that the exponential moving coefficient for CV can be different from adaptive sampling to reduce the variance caused by a_t^* estimation. There are two considerations during implementation:

1. *Number of textures.* Line 8 indicates that we need to store 9 parameters, which would require three textures of *floatRGBA16*. But $\boldsymbol{\Sigma}_t$ is symmetric, and $\boldsymbol{\Sigma}_t.xx$ is not used. Thus only 7 parameters (two textures) are actually needed to keep the history.

2. *Sampling value.* Since rendering solves $L_o(p, t)$, to efficient compute the monitoring value $\mathcal{S}(p_t)$ and $\mathbf{Z}_t$, we can derive another formulation without calculating intermediate values as

$$\bar{f} = L_o(t) - (1 - \gamma) \cdot B(t), \tag{6.22}$$

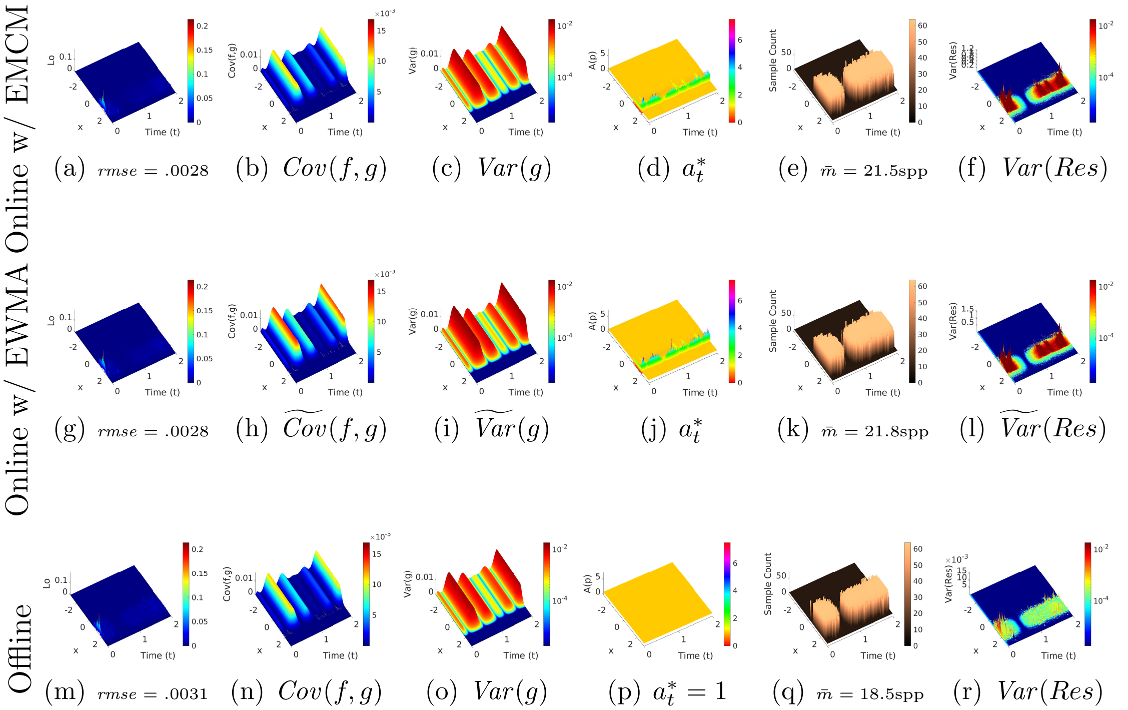
$$Res(t) = \bar{f} - a_t^* \cdot G(t). \tag{6.23}$$

Fig. 6.5(a)–(l) shows an example of applying the algorithm with EWMA covariance estimator and EMCM. The sample count and quality are close. However, we have a slightly smaller sample count with EMCM. Moreover, the approximation of EWMA brought higher variance and covariance changes shown between Fig. 6.5 (c) and (i), (b) and (h). Therefore, we select EMCM. Note that Eq. 6.23 uses $G(t)$ instead of $\bar{g}$ (no matter whether the standard or constant control variable is used) to remove temporal variance while still keeping spatial variance to avoid under-sampling in both static and dynamic lighting scenarios (see Appendix B.4.3).

6.6 Offline CV Coefficient Estimation

From the online solution, we find an approximation, $a_t^* \approx 1$, leading to an even more efficient implementation for real-time applications.

Fig. 6.5(d) shows that for most regions in time, even with time variant lighting condition, a_t^* remains 1. Fig. 6.5(m)–(r) show the corresponding states if we set $a_t^* = 1$. The quality is a little worse ($.0031 > .0028$) with a reduced average sample



(a) $rmse = .0028$ (b) $Cov(f,g)$ (c) $Var(g)$ (d) a_t^* (e) $\bar{m} = 21.5$spp (f) $Var(Res)$

(g) $rmse = .0028$ (h) $\widetilde{Cov}(f,g)$ (i) $\widetilde{Var}(g)$ (j) a_t^* (k) $\bar{m} = 21.8$spp (l) $\widetilde{Var}(Res)$

(m) $rmse = .0031$ (n) $Cov(f,g)$ (o) $Var(g)$ (p) $a_t^* = 1$ (q) $\bar{m} = 18.5$spp (r) $Var(Res)$

Figure 6.5: Sample count m_t estimation with online and offline CV coefficients a_t^* estimation.

count (-3 spp), but we can save one texture and the corresponding calculation.

Our online solution requires two textures. When the memory capacity or bandwidth budgets are tight, one more texture per pass might be too demanding.

6.7 Static Lighting

In the previous sections for control variates, we assume that the static scene has all lighting constant, namely $Var(T) = 0$. It leads to a CV coefficient $a = 1$ based on Eq. 6.21. However, in a real-time rendering engine, static lighting might still have dynamics due to sub-pixel jittering for temporal anti-aliasing [Yang et al., 2020]. More specifically, the platform in our research (i.e., Unreal Engine 4) uses 8 temporal samples in sample space based on Halton sequence, as shown in Fig. 6.6b. If the sub-pixel has a high frequency like on the orange (Fig. 6.6a), there are

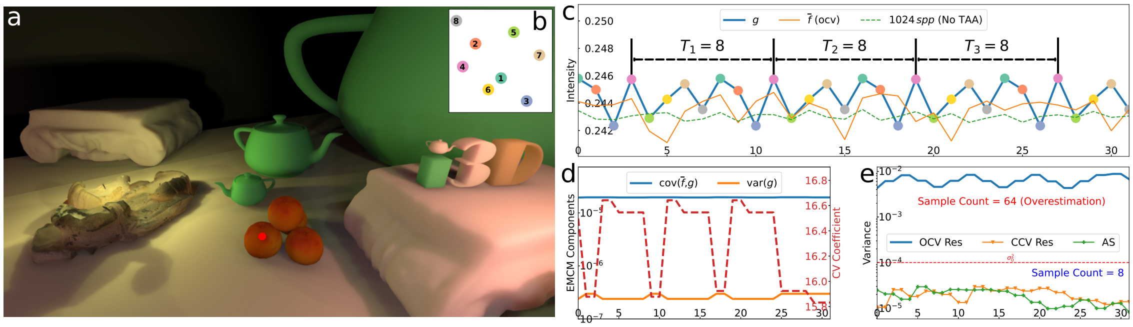


Figure 6.6: Online control variates on the static scene in real-time rendering engine leads to sample count over-estimation due to subpixel jittering. However, it leads to an equivalent quality compared to fixed sampling with 64 spp.

still temporal periodic dynamics (the control variable g shown in Fig. 6.6c). These periodic dynamics make the online CV coefficient find the traditional CV coefficient to minimize the periodic dynamics, instead of making $E[F - aG] \rightarrow 0$, thus creating large CV coefficients shown in Fig. 6.6d when not clamped. This operation leads to variance over-estimation shown in Fig. 6.6e, yet we can achieve an equivalent quality when compared to 64 spp due to this over-estimation. To avoid this over-estimation, we can increase ϵ_a to a larger value like 10^{-4} to mask this pixel jittering (See Section 7.3). Please note that even with this over-estimation, it still runs faster than the constant sampling rate of 64 spp. Although with this sub-pixel jittering, the EMCM components for sub-pixel jittering are small. It is smaller than 10^{-6} in the example shown in Fig. 6.6d. During dynamic lighting, this contribution can be ignored. It has a similar performance to CCV (constant Control Variates) and runs faster than AS (adaptive sampling), see Section 7.4. Refer to Appendix B.5 for the temporal insights of more pixels.

6.8 Discussion and Limitation

Since the main purpose of control variates is to reduce the variance during dynamic lighting conditions, the control variates does not reduce the variance when the lighting is stable as how control variates is used generally in offline rendering. The reason is that we use a constant control variable in each frame. It does not reduce the variance. However, since it is dynamic according to time, it can be used to minimize temporal variance.

To reduce variance during stable lighting, we need to find a control variable that is not constant in a frame. For the application of subsurface scattering, we need an efficient control variable that samples the texture or caches efficiently during the correlated sampling with the function to integration. The complexity should be $O(1)$ instead of $O(n)$, where n indicates the number of samples. Otherwise, it is not suitable for real-time subsurface scattering. However, it is still viable with a constant in-frame control variable for stable lighting in real-time rendering engines where per-frame jittering has been applied. Because frame jittering introduces temporal dynamics in real-time rendering when TAA is enabled, our algorithm can capture this temporal information to derive the CV coefficient. However, it performs over-estimation, yet leading to equivalent quality compared to fixed sample count of 64 spp with better performance.

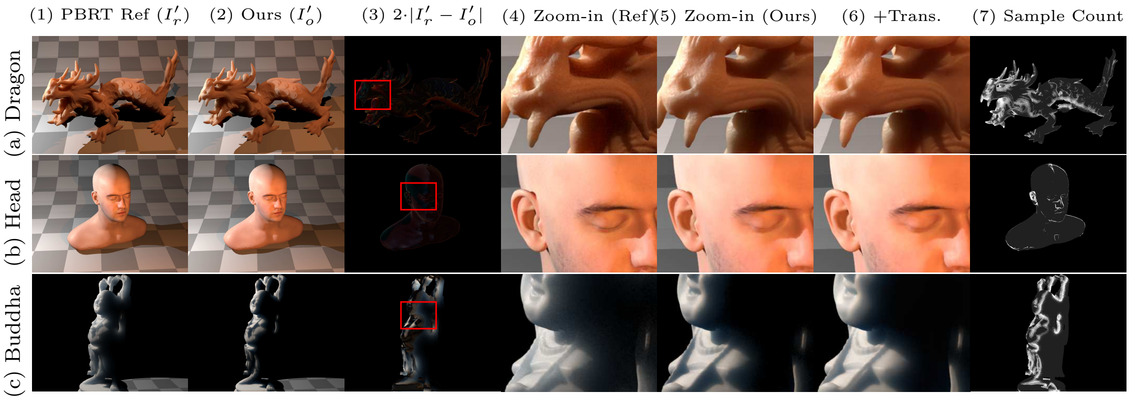


Figure 7.1: Subsurface ground truth comparison (without transmission). (a) Stanford Asian Dragon, (b) Infinite-Realities head, and (c) Stanford Happy Buddha at 1366×1024. In each row, we show (1) PBRT reference, (2) our scattering, and (3) difference from PBRT. We also zoom into a high difference region for (4) PBRT and (5) ours (6) with transmission. The (7) sample count from our one pass adaptive sampling (white = 64 spp, black = 0 spp). For our algorithm, $\kappa = 0.2$, $\sigma_0^2 = 0.001$, $[b_{\text{min}}, b_{\text{max}}] = [8, 64]$ spp.

7.1 Implementation

We implemented our subsurface scattering as a single screen-space adaptive sampling post-processing pass in UE4, without modifying the final TAA pass. Fig. 5.8 summarizes the subsurface scattering pass. The scene color is broken into diffuse irradiance and non-subsurface irradiance. Then we prefilter the diffuse irradiance map to accelerate cache hits when incoherently sampling the irradiance. During screen-space subsurface scattering, we use our importance approximation Eq. 5.6

to sample in the subsurface plane that is perpendicular to our view direction. The number of samples is estimated in the variance guiding phase. Since we could have different adjacent subsurfaces, we cache an 8-bit profile ID texture to resolve how bleeding color between profiles is mixed. After sampling, the result is used to update the history texture that stores $\mathcal{H}_i$ per pixel with $\alpha_0 = 0.2$ (the max weight for the non-transparent object in UE4). Finally, the scattering result is combined with surface albedo and the non-subsurface part to form the final output.

Bilateral filtering. We adopt depth-based bilateral filtering [Golubev, 2018] to solve the bleeding problem between distinct scattering surfaces. We extended Eq. 5.2 as:

$$L_o = \frac{\sum_{j=1}^{n_{(i)}} \mathbb{1}_s(q_j) \cdot r'_{q_j} R(r'_{q_j})/pdf_{q_j} \cdot L_{q_j}}{\sum_{j=1}^{n_{(i)}} \mathbb{1}_s(q_j) \cdot r'_{q_j} R(r'_{q_j})/pdf_{q_j}} \tag{7.1}$$

where $r'_{q_j} = \sqrt{r^2_{q_j} + \Delta D^2_{q_j}}$, ΔD_{q_j} is the depth difference between q_j and the center sampling point, and the indicator function $\mathbb{1}_s(q_j)$ is 1 if there is subsurface and 0 otherwise at q_j.

7.2 Static Scene

We include comparisons to evaluate the quality and speed of our adaptive sampling algorithm. For quality, we compare root mean square error (RMSE) and gray-scale peak signal-to-noise ratio (PSNR). For real-time performance, we compare speed and quality to Burley's method without the adaptive sampling, and to the separable screen-space method that is the standard implementation in UE4. We modified it to approximate Burley's model instead of Dipole. Please refer to

Appendix A for more information about our approach and validation of this approximation. Unless otherwise specified, all performance numbers are measured on NVIDIA Quadro P4000 with a resolution of 1366x1024. We measure time in *ms* for just the subsurface work.

7.2.1 Quality Comparisons

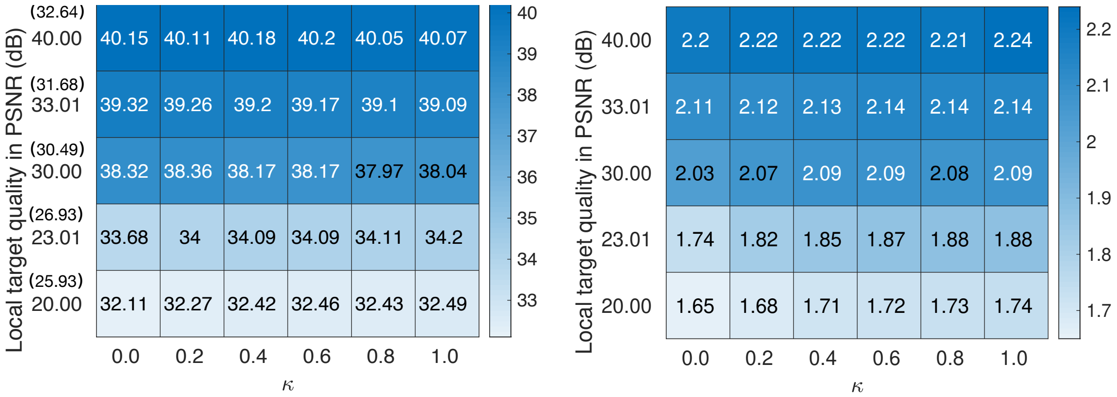
We compare against PBRT ground truth in Fig. 7.1 for three scenes: Dragon, Infinite-Realities head, and Happy Buddha. We compare our screen-space subsurface scattering (with no transmission) to the PBRT *path* integrator with *Disney* material with $maxdepth = 1$. To focus on subsurface and minimize the difference caused by different light and tone mapping implementations, only point lights are used, with maximum shadow resolution in the UE4 rendering. Tone mapping is not applied in either renderer for ground truth comparison. The results show that, while some differences are visible, they remain qualitatively low. We hypothesize the most significant differences are due to the PBRT renderings including transmission paths, while diffusion models cannot. UE4 does include a separate translucent object transmission model. To confirm this as the major source of observed errors, we replaced the UE4 transmission profile with Burley's model as shown in Fig. 7.1(6). This produces a closer (though still not exact) match. We believe a better real-time transmission model could further reduce the difference.

7.2.2 Adaptive Sampling Quality

For real-time timing evaluation, we compare our adaptive algorithm to a fixed sample-count interactive implementation of the Burley model, and the UE4's separable screen-space diffusion model, tuning each for approximately equal quality as compared to a 2K sample per pixel ground truth within UE4. Performance for the sampling models is worst at a close viewing distance since the texture accesses are least coherent then, with cache misses causing significant performance degradation. Therefore, we evaluate the quality and performance of adaptive sampling at both a normal viewing distance, and at a view close to the surface.

Regular distance. We compare a fixed 2K-sample rendering of the Buddha model to our adaptive algorithm, with a max sample count of 64 and varying σ_0^2 and κ. For timing, both the Buddha and checkerboard base plane use the subsurface material, though the PSNR is only calculated for the Buddha pixels. Fig. 7.2 shows the PSNR of the Buddha and the subsurface time. Our algorithm runs faster (1.97×) with negligible quality difference (40.51-40.2=.31 dB) when local target quality is $\sigma_0^2 = 0.0001$ (PSNR=40.00 dB). Moreover, our algorithm can make use of TAA to boost the performance with small quality degradation. For examples, when the local target quality is $\sigma_0^2 = 0.001$ (PSNR $= 30.00dB$), the final quality reaches 38.36 dB with a speedup by 2.12× in 2.07 *ms*. The last column of Fig. 7.1(c) shows the sample count of the Buddha in this configuration.

Close distance. We use a forehead skin patch from Digital Mike model that has high scattering distance in UV space to investigate this case. Fig. 7.3 shows the



Local target quality in PSNR (dB)	0.0	0.2	0.4	0.6	0.8	1.0
(32.64) 40.00	40.15	40.11	40.18	40.2	40.05	40.07
(31.68) 33.01	39.32	39.26	39.2	39.17	39.1	39.09
(30.49) 30.00	38.32	38.36	38.17	38.17	37.97	38.04
(26.93) 23.01	33.68	34	34.09	34.09	34.11	34.2
(25.93) 20.00	32.11	32.27	32.42	32.46	32.43	32.49

(a) PSNR for Buddha region

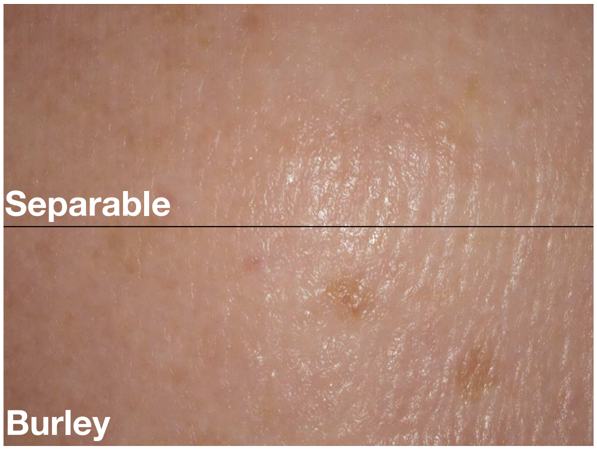
Local target quality in PSNR (dB)	0.0	0.2	0.4	0.6	0.8	1.0
40.00	2.2	2.22	2.22	2.22	2.21	2.24
33.01	2.11	2.12	2.13	2.14	2.14	2.14
30.00	2.03	2.07	2.09	2.09	2.08	2.09
23.01	1.74	1.82	1.85	1.87	1.88	1.88
20.00	1.65	1.68	1.71	1.72	1.73	1.74

(b) Pass time (ms)

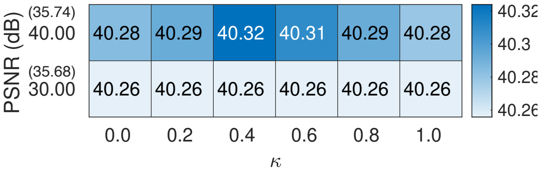
Figure 7.2: Varying PSNR and κ for the Buddha scene with $b_{\max} = 64$ spp. Color shows pass time in ms. PSNR numbers in parentheses are single-frame without TAA. Fixed 64 spp runs in 4.38 ms with final PSNR$=$ 40.51 (32.91) dB.

final PSNR and subsurface time. Especially, when local target quality is $\sigma_0^2 = 0.001$ (PSNR $= 30$ dB), our algorithm adaptively reduced sample count to $b_{\min} = 8$ spp for all κ. We almost achieved equal quality when compared to fixed 64 spp (40.26 dB vs. 40.39 dB) with a speedup of 4.15× (from 7.51 ms to 1.81 ms). We expect better performance for adaptive sampling in these cases vs. fixed sampling, since adaptive sampling reduces the number of incoherent non-cached texture accesses.

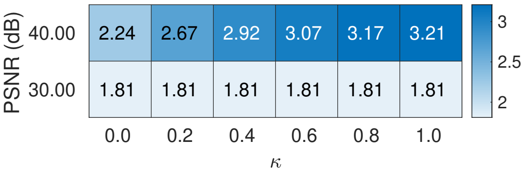
With the sampling bandwidth bottleneck of close views, the 8-bit cached profile ID texture is critical for performance. Though computed each frame, profile ID texture generation is coherent, and it reduces the incoherent accesses from 16 bytes per sample to one byte per sample. Without the texture, quality level 0.001 took 4.23 ms (6.13×) while fixed 64 spp took 25.94 ms. When compared to Separable, the rendering time of this case is faster than our model, at 3.06 ms, but the separable approximation shows significant banding artifacts in close views, resulting in a worse PSNR $= 39.95$ dB. We show a better comparison of the artifacts to ours in Fig. 7.4.



(a) Close skin patch

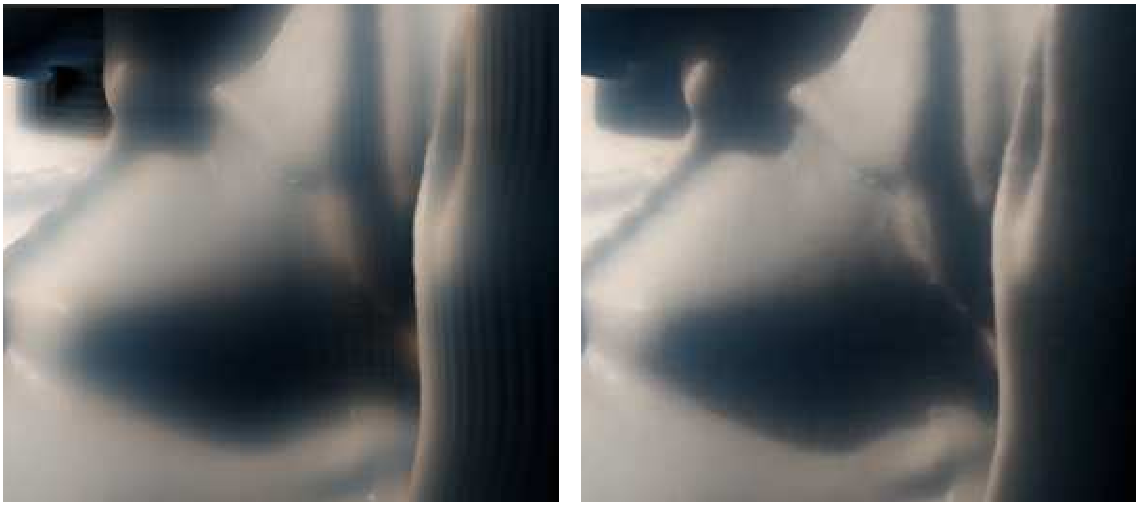


(b) Final PSNR



(c) Final PSNR

Figure 7.3: Varying PSNR and κ for the close skin patch with $b_{\max} = 64$ spp. Color shows pass time in ms. PSNR numbers in parentheses are single-frame without TAA. Fixed 64 spp runs in 7.51 ms with final PSNR= 40.39 (36.02) dB.



(a) Separable (b) Ours

Figure 7.4: Quality comparison of high scattering marble material. Observable vertical banding artifacts in (a) Separable.

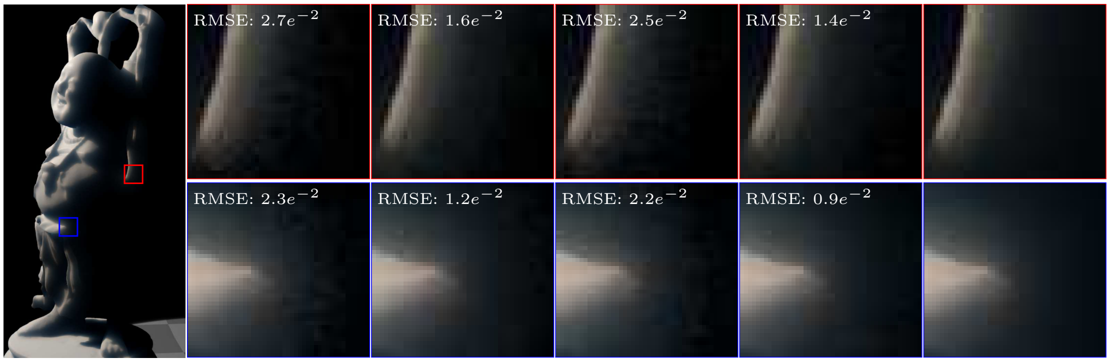
7.2.3 Equal Quality Comparison

Fig. 7.5 compares a single frame Buddha scene quality between our adaptive sampling algorithm and fixed sampling, given approximately the same execution time. In the Buddha scene, we fixed $b_{\min} = 8, b_{\max} = 64$ with two different σ_0^2, κ settings that lead to the best quality ($\sigma_0^2 = 0.001, \kappa = 0.2$, and $\sigma_0^2 = 0.0001, \kappa = 0.6$). The quality compared to 2K-spp ground-truth is better using our adaptive sampling algorithm.

Fig. 5.1(a) and (c)/left compares quality on the Digital Mike model for equal time comparison between our adaptive sampling algorithm and the fixed sampling algorithm without pre-filtering and profile caching as a baseline. We also show two other views with the same settings (middle and right). Fig. 5.1(d) shows the separable screen-space filtering algorithm for the same views. Fig. 5.1(e) visualizes the sample counts for our adaptive algorithm in each view. The result shows that our algorithm targets best quality with a single setting that runs comparable or even better than Separable.

7.2.4 Real-time Counterpart Comparison

We also implemented the state-of-art screenspace subsurface scattering by Golubev [2018] into our framework as a fixed 64-sample ground truth with the Burley diffuse profile instead of our approximation. The result is shown in Fig. 5.1(b). Based on the approximation comparison in Fig. 5.3, the bias should be small. Differences between Burley's model and our approximation are small in the ear (0.29



	fixed / 2.08ms	adap. / 2.08ms	fixed / 2.20ms	adap. / 2.21ms	ground truth
Config:	26 spp	$\sigma_0^2 = .001$	28 spp	$\sigma_0^2 = .0001$	$2k$ spp
Image PSNR:	36.61 dB	38.51 dB	37.14 dB	40.39 dB	

Figure 7.5: Equal time comparison: fixed vs. adaptive.

dB) and front (0.26 dB) scenarios. However, those differences are relatively large in the skin patch scenario (2.09 dB). Further inspection of the adaptive sampling count in Fig. 5.1(e), shows that only 8 samples are used per pixel to meet the target quality. Although there is a small quality degradation, our sampling algorithm is a good mechanism to prevent oversampling when the quality is already met for real-time rendering. Despite the minor quality drop, we have a significant performance gain ranging from 2.3× to 4.3×.

Table 7.1: Adaptive vs fixed phase breakdown for Digital Mike in Fig. 5.1 (ms).

Scenario	Setup	Pre-filtering	Sampling	Update	Combine	Total
(L)+fixed	0.38	0.16	10.73	N/A	0.20	11.47
(L)+adt.	0.38	0.16	1.50	0.54	0.20	2.78
(C)+fixed	0.41	0.17	9.72	N/A	0.27	10.22
(C)+adt.	0.41	0.17	2.72	0.46	0.27	4.03
(R)+fixed	0.35	0.17	1.45	N/A	0.11	2.08
(R)+adt.	0.35	0.17	0.38	0.14	0.11	1.15

7.2.5 Performance Breakdown

To help understand the cost of the variance guiding phase, the update pass in Fig. 5.8 is separated from the sample estimation and sampling process for time measurement. We also compared fixed and adaptive 64 spp cost to illustrate how techniques utilizing our method might perform on different hardware. The result is presented in Table 7.1. The performance is measured as the median time (in ms) for the left (L) close patch, center (C) ear, and right (R) front image in Fig. 5.1.

Table 7.2: The pass time (ms) and PSNR for rendering the scattering pass under different sample configurations: *fix* the sample per pixel is fixed, *adt.* using our single pass adaptive sampling, and *SRatio* the total sample ratio between *adt* and *fix*. The reference image is *fixed* 2K spp. err_1 is the RMSE of pixels that already meet the target quality $\sigma_0^2 = 0.0001$ with $\hat{n} = b_{\min}$ at $b_{\max} = 64spp$, err_2 the RMSE of pixels that need $n > b_{\min}$.

		a) 64 spp		b) 128 spp		c) 512 spp	
		fix	adt.	fix	adt.	fix	adt.
Dragon	time	2.29	2.10	4.1	3.14	14.77	8.6
	PSNR	46.93	46.61	48.34	47.24	48.64	48.01
	err_1	.006	.006	.006	.006	.006	.006
	err_2	.004	.005	.004	.004	.004	.004
	SRatio	78.66%		64.92%		36.70%	
Head	time	1.33	0.82	1.99	0.93	6.44	1.62
	PSNR	51.29	48.54	51.57	49.71	52.03	49.86
	err_1	.003	.003	.003	.002	.002	.002
	err_2	.005	.010	.005	.010	.004	.010
	SRatio	16.00 %		9.54%		4.41%	
Buddha	time	1.68	1.35	3.34	2.01	9.63	5.74
	PSNR	41.04	40.75	43.90	43.59	48.85	47.98
	err_1	.006	.006	.005	.006	.003	.003
	err_2	.011	.011	.007	.007	.004	.004
	SRatio	59.84%		54.29%		39.19%	

7.2.6 Effect of Sample Budget on Time and Quality

We observe that adaptive 64 spp still differs from 2K sample ground truth for high scattering material, marble. The local target quality is not met (Figure 7.2(a)) for $\sigma_0^2 > 0.001$. Since the run-time is still low, we explore the effect of adding more adaptive samples for all three test scenes in Figure 7.1 with $\sigma_0^2 = 0.0001, \kappa = 0.2$. The checkerboard is not subsurface. The result is shown in Table 7.2. For small *dmfp* (Head) with high sample count sparsity, we could use only 4.41% of 512 spp to

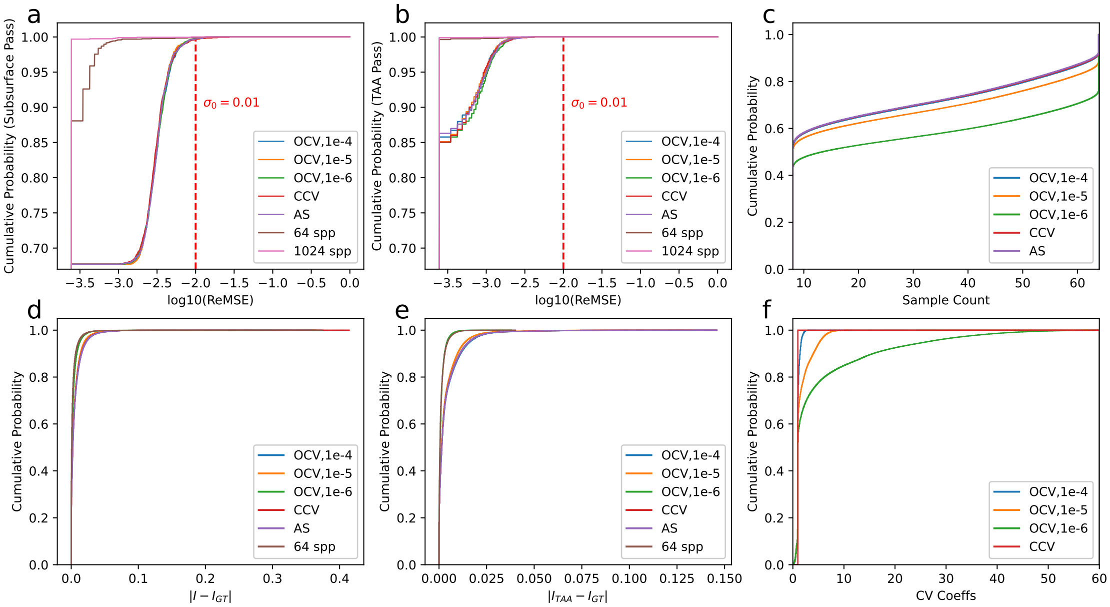


Figure 7.6: Adaptive sampling history and quality analysis for teaser in Fig. 7.7 with static lighting.

achieve 49.86 dB in 1.62 ms. The PSNR is lower than fixed 64 spp because we have set a target quality level, where most sparse regions will not increase the sample count and remain as $\hat{n} < b_{\min}$ if the quality has already been met (the err_2 of adt. sampling for Head).

7.3 Control Variates on Static Scene

In this section, we analyze the quality with real-time control variates in static scenes. In adaptive sampling, we try to achieve a target variance. Specifically, we are interested in how adaptive sampling and CV affect the detailed quality. We use the empirical cumulative distribution function (ECDF) to present the square root of moving variance of the subsurface scattering pass within a frame for adaptive sampling and the square root of the moving CV residual variance. The final image quality is presented with absolute error in ECDF compared to the ground truth with 1024 spp. This experiment fixed the coefficient boundary as $a_0 = 1000$ and the exponential coefficient $a_{cv} = 0.005$ for CV histories. Note that although we test in static lighting, the jittering of sampling still creates temporal dynamics in real-time rendering engines.

In the static scene, we find that our adaptive sampling with online CV provides *equivalent quality* compared to rendering with fixed sample count if TAA is applied. Fig. 7.6(a) shows the empirical cumulative distribution function (ECDF) of the square root of the exponential moving variance (ReMSE) in the subsurface scattering pass for adaptive sampling (AS), constant CV (CCV), and Online CV (OCV) with different CV coefficient estimation factor ϵ_a. σ_0 is the target error for real-time adaptive sampling control. For fixed sampling, it just monitors the ReMSE without the loop to control the sample count. This figure shows that nearly all pixels have quality better than the target quality level. To show the corresponding perceivable error, we use absolute luminance error. Fig. 7.6(d) shows the absolute luminance error between the ground truth (I_{GT}, 1024 spp with TAA) and the direct subsurface scattering pass result without TAA (I). We could observe that OCV with $\epsilon_a = 1 \times 10^{-6}$ has an equivalent error distribution compared to 64 spp. After the application of TAA, the error distribution has been reduced, as shown in Fig. 7.6 (b) compared to Fig. 7.6(a). We use EMV in the TAA pass to monitor the variance with the TAA history weight instead of the constant weight used in our subsurface scattering guiding pass ($\alpha = 0.2$). Although this error is larger than the fixed 64 spp configuration, we have equivalent quality in pixel comparison as shown in Fig.

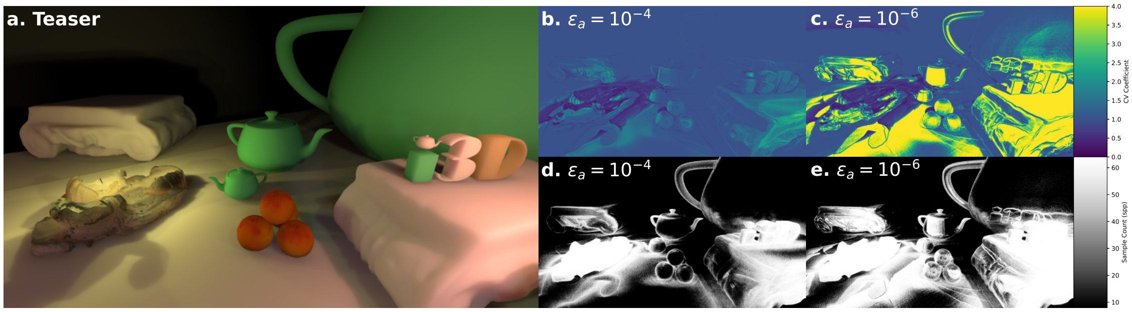

a. Teaser
b. $\epsilon_a = 10^{-4}$
c. $\epsilon_a = 10^{-6}$
d. $\epsilon_a = 10^{-4}$
e. $\epsilon_a = 10^{-6}$
CV Coefficient
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Sample Count (spp)
60
50
40
30
20
10

Figure 7.7: CV Coefficient and sample count for Teaser in a static scene. Dynamic lighting and fire transparency are disabled for subsurface scattering quality comparison.

7.6(e) with $\epsilon_a = 10^{-6}$. The quality is also better than AS and CCV. The smaller the constant ϵ_a, the closer the CV coefficient is to the ground truth while avoiding zero dividings. The CV coefficient distribution is shown in Fig. 7.6(f). A direct view is also presented in Fig. 7.7(b) and (c). It then leads to higher variance. The sample count has been increased as shown in Fig. 7.6(c) to reduce the variance lower to the target quality level. This increased sample count is also illustrated in the sample map view in Fig. 7.7(e) compared to (d) with $\epsilon_a = 10^{-4}$.

7.4 Dynamic Scene

Table 7.3: The mean metrics across all frames and the max performance ratio for adaptive sampling (AS), adaptive sampling with Online CV coefficient (+OCV), and with constant CV coefficient (+CCV).

	Sample Count (spp)				Sampling Pass Time (ms)				PSNR (dB)			
	AS	+OCV	+CCV	Max ratio	AS	+OCV	+CCV	Max ratio	AS	+OCV	+CCV	Max Ratio
Flash lighting	61.97	29.58	26.88	x2.75	1.47	0.80	0.75	x2.79	51.19	49.55	49.43	x0.94
Regular lighting	26.51	18.32	17.82	x4.00	3.39	2.55	2.58	x3.11	47.85	47.58	47.55	x1.01
Dynamic scene	22.01	14.70	14.13	x2.02	2.02	1.54	1.52	x1.98	50.62	49.11	49.11	x0.91

In dynamic scene, the target quality has $\sigma_0^2 = 0.0001$, $\alpha = 0.2$, $a_{cv} = 10^{-6}$, $\kappa = 0.2$, $\epsilon_u = 0.01$ and $\epsilon_a = 10^{-6}$, the sample count is clamped between 8 and 64 spp. a_{cv} is the exponential moving coefficient for CV history update. We created three scenes to capture typical dynamic lighting changes to show how our proposed

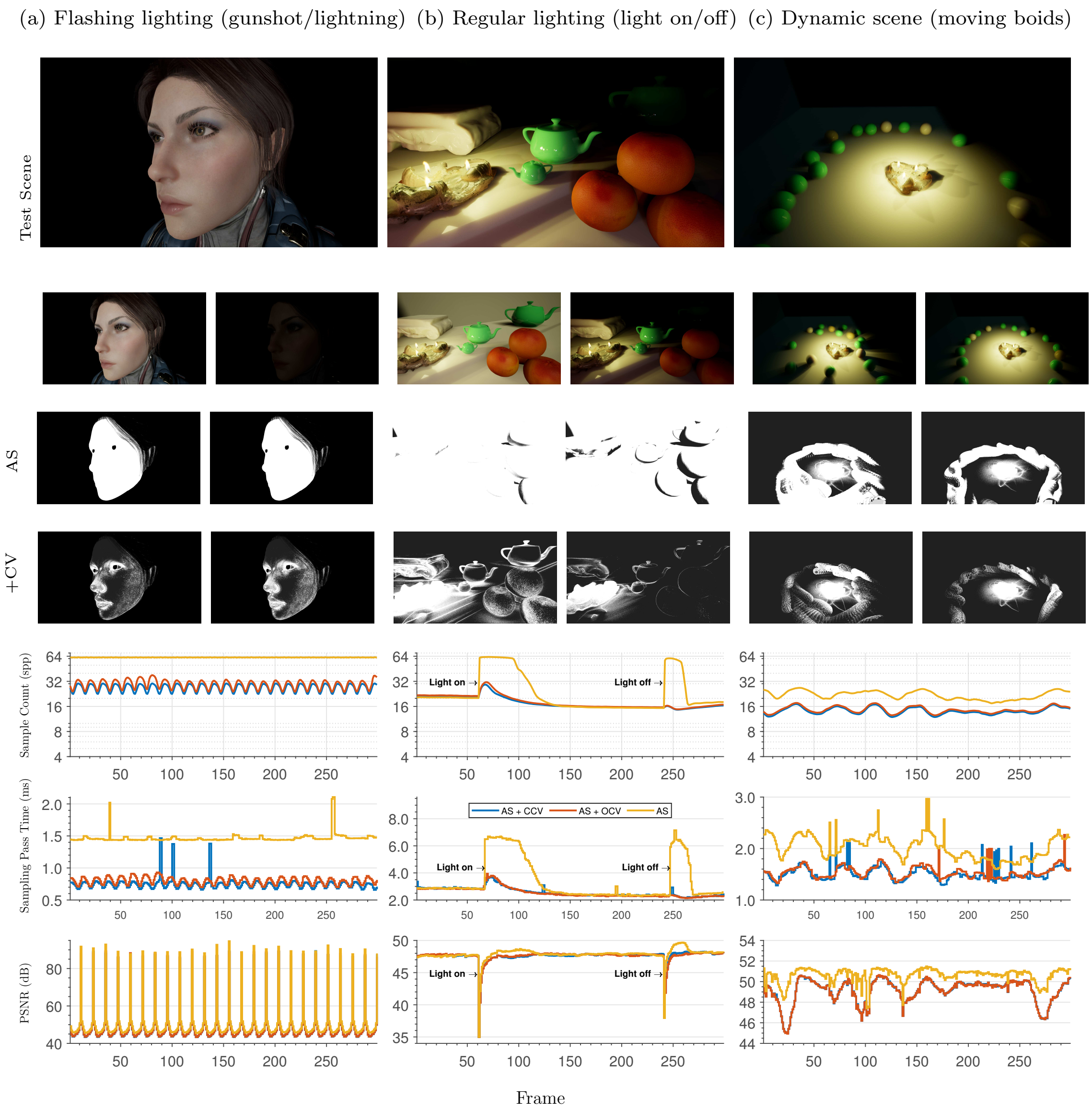

(a) Flashing lighting (gunshot/lightning) (b) Regular lighting (light on/off) (c) Dynamic scene (moving boids)

Figure 7.8: Real-time CV performance and quality test at resolution 2560x1440 for the three test scenes: (a) Flashing lighting with additional snapshots at frame 101 and 108, (b) Regular lighting at 71 and 251, and (c) Dynamic scene at 25 and 200.

algorithm works in whole scene subsurface scattering:

1. Flashing lighting. Flashing light is used to simulate gunshots or lightning on human face, using a directional light with intensity $I(t) = \sin(2\pi f t) + 1$ with $f = 5\ Hz$.

2. Regular lighting. We simulate regular lighting changes with a light switching on

and off, and flickering candle lights. The directional light is turned on/off every 3 seconds.

3. Dynamic scene. Dynamic moving boids are added to the scene creating dynamic shadowing when they move around a candle.

We choose to use three metrics for the evaluation.

1. Sample count (spp). The per-frame average samples per pixel. It is calculated on subsurface regions only.
2. Sampling pass time (ms). The sampling pass time is only measured for the MC sampling process. A 3rd-order median filter is applied to remove unstable measurements.
3. PSNR (dB). The PSNR is measured on the luminance of the subsurface scattering region. The ground truth image is captured with 1024 spp per frame.

To make the results reproducible, we use fixed random seed and UE4 *Sequencer* to render a standard dynamic range (SDR) 60 frame-per-second (FPS) avi, with sample count and PSNR measured per frame. To get the sampling pass time, the built-in csv log is used during sequencer recording. The performance is measured with a resolution of $2560 \times 1440p$ on an NVIDIA RTX 2080Ti.

Fig. 7.8 shows the performance and quality test results for the selected three scenes over 300 frames. For each test scene, we show the scene, two captures to demonstrate temporal dynamics, the sample count texture for adaptive sampling, the texture with online CV, the sample count, sampling pass time, and PSNR.

Table 7.3 shows the corresponding average over time. Since it is very important to always maintain high performance in real-time rendering, the max performance ratios before and after adding CV are also shown.

Max performance ratio. Fig. 7.8(b) shows a good example when light is switching on/off. Without control variates, the sampling pass time increases up to 3.11× at frame 252 (7.15 ms vs. 2.33 ms) compared to with-CV even when quality is not increased. With CV, the pass time is more stable and friendly to real-time applications. Note that, in the sampling pass time of Fig. 7.8, some sudden increase in pass time can be observed. We believe it is due to memory incoherence since the samples are based on importance sampling.

Sample count reduction. The reason we can have better performance is that CV leads to lower sample counts, thus less pass rendering time. During continuously frequent dynamic lighting change (Fig. 7.8(a)), temporal variance leads to high sample counts (61.97 spp) and an average of 1.47 ms for the sampling pass. CV leads to consistently lower average sample count (17.82 spp) and pass time (0.75 ms) with quality drops of 1.76 dB.

Quality effects. We observe a slight quality drop after applying CV. the quality drops the highest during high dynamic scenes (Fig. 7.8(c)) with fast moving subsurface objects, down 0.91× when compared with adaptive sampling alone at frame 25. However, we think the absolute PSNR is high enough for the introduced performance.

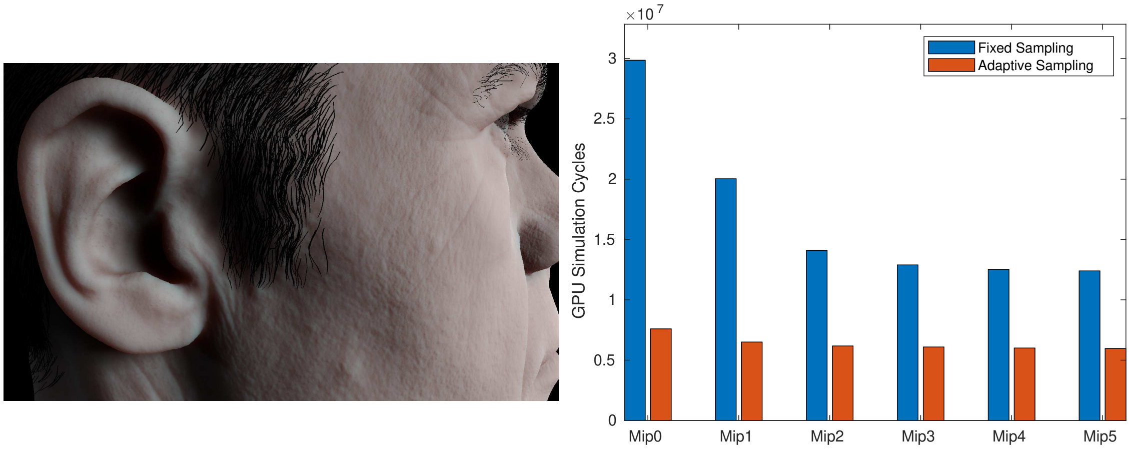
Online vs. constant CV. The average quality is expected to be better with online CV. We tend to have more samples during our 1D illustration example in

Fig. 6.5 as the algorithm tends to perform oversampling. In Fig. 7.8, a similar small increase in sample count is detected during our 3D scene tests. The constant CV seems to be more efficient regarding the memory usage of the online CV in real-time rendering.

7.5 Cache Analysis

Our adaptive sampling algorithm helps to reduce the bandwidth demands. However, it is still unknown how much cache access has been reduced without cycle level simulation and how the algorithm affects the cache hit rate on different cache levels. To answer these questions, we use GPGPU-Sim to provide a detailed simulation of the contemporary GPU from a close view.

7.5.1 GPGPU-Sim



(a) Sampling result without combine

(b) GPU simulation cycles

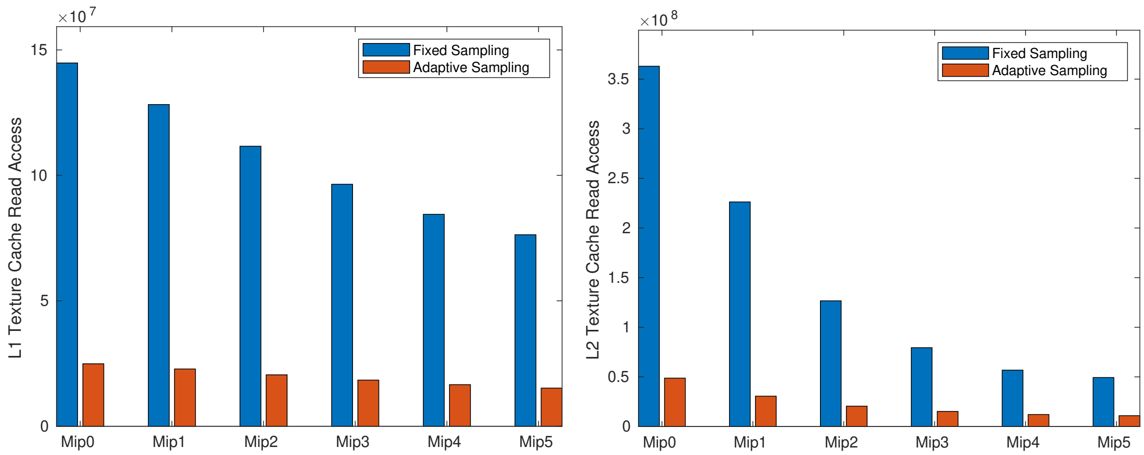
Figure 7.9: The cycle level performance for (a) the sampling pass under (b) fixed and adaptive sampling schema. Due to adaptive sampling, the simulation cycles are reduced by 3.9 × (Mip0), 3.1×(Mip1), 2.3×(Mip2), 2.1× (Mip3), 2.1×(Mip4), and 2.1×(Mip5).

GPGPU-Sim 3.x [Khairy et al., 2020, Bakhoda et al., 2009] is a general framework for performing cycle-level analysis of parallel computing code that primarily supports CUDA (Compute Unified Device Architecture). CUDA is designed by NVIDIA and is a parallel computing platform and programming model that uses a GPU for general-purpose computing. The corresponding coding language is similar to C/C++, and different shading languages like High-Level Shading Language (HLSL) used to implement real-time rendering passes. This platform enables us to perform analysis once the subsurface scattering code has been manually ported to CUDA.

Specifically for a given task (Fig. 7.9(a)) in a frame, we first capture all frame buffers with RenderDoc. Then all buffers required for sampling are exported for a CUDA version of the subsurface scattering to import for the simulation. Since the MIP level is not supported in GPGPU-Sim, only adaptive sampling is analyzed without the filtered importance sampling part of AFIS. The GPGPU-Sim is configured to use NVIDIA TITAN X Pascal. The scattering result is stored in the data cache not to affect the texture access analysis.

7.5.2 Adaptive vs. Fixed Sampling Simulation

Fig. 7.9(a) shows the selected rendering task. We perform the detailed cycle level analysis through GPGPU-Sim. Fig. 7.9(b) shows the detailed GPU simulation cycles required to sample the irradiance texture for subsurface scattering in a single frame under different MIP levels for both fixed and adaptive sampling. We further



(a) Total L1 Cache Read Demand

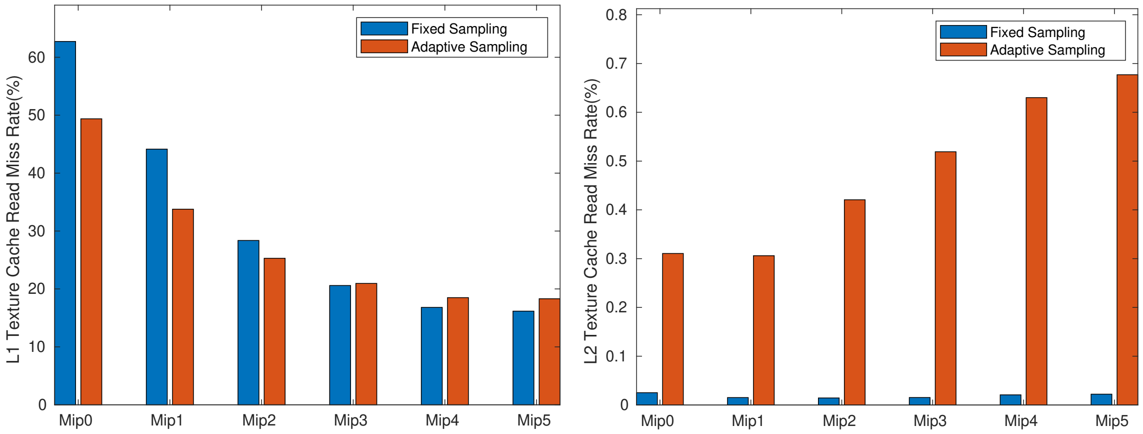
(b) Total L2 Cache Read Demand

Figure 7.10: Total L1 and L2 cache read demand for near field rendering in a single frame with fixed and adaptive sampling when the irradiance texture is sampled at different MIP levels. With adaptive sampling L1 cache read is reduced by 5×, L2 cache read by 7× to 4×.

demonstrate the cache access demand and the cache hit rate to understand why adaptive sampling uses fewer cycles.

As expected, the total cache read in L1 and L2 for adaptive sampling is much less than fixed sampling (Fig. 7.10). However, what is interesting is the L1 and L2 cache miss rate shown in Fig. 7.11. L1 cache is highly incoherent and up to 62.78% for fixed and 49.38% for adaptive sampling. This high cache miss rate reflects the incoherent cache access pattern. Since adaptive sampling has a lower sample count demand, it tends to cause fewer conflicts, thus has a better cache hit rate in L1 cache.

Interestingly at Mip 0, although adaptive sampling has much less total texture read access demand (48,733,014 vs. 363,015,023) at L2 cache, the L2 texture cache read miss count is slightly higher than fixed sampling (151,356 vs. 91,541). This leads to a higher cache read miss in Fig. 7.11(b).



(a) L1 Cache Read Miss

(b) Total L2 Cache Read Miss

Figure 7.11: L1 and L2 cache read miss rate (%) for near field rendering in a single frame with fixed and adaptive sampling when the irradiance texture is sampled at different MIP levels.

With this detail, we find that if we can have an alternative algorithm to solve a problem that requires much less bandwidth demand, the performance can be increased even when the cache hit rate is reduced at a certain cache level, like the L2 cache in Fig. 7.11(b).

Chapter 8: Conclusion

Photo-realistic subsurface scattering is an appealing feature in real-time applications like games. However, subsurface scattering based on Monte-Carlo sampling is expensive in real-time rendering because of the incoherent cache access in the contemporary GPU cache architecture and the high bandwidth demands within a frame time of several milliseconds or sub-millisecond. In this dissertation, we proposed a taxonomy of heterogeneous real-time rendering techniques in terms of cache and bandwidth cost minimization at the algorithm level. The mathematically sound adaptive sampling acceleration technique is designed to minimize the sample count and memory demands. To further reduce the computing demands for scalability, a hybrid combination of our adaptive sampling and the separable technique is proposed to achieve high frame-rate subsurface scattering with high quality (Chapter 1).

Chapter 2 briefly reviews Monte Carlo integration, subsurface scattering models, and the acceleration techniques used in the literature and industrial. Chapter 3 introduced the taxonomy of heterogeneous real-time rendering to evaluate cache and bandwidth cost to improve the performance and detailed what techniques have been proposed in the framework. Meanwhile, the cost is described by a mathematical description. For subsurface scattering, we have characterized our algorithm novelty within the taxonomy. Namely, we 1) reduce the computing demands with hybrid acceleration techniques (AFIS and Separable), 2) propose adaptive sampling local guiding to support different temporal reuse passes (e.g., TAA and DLSS) to minimize sample count, and 3) propose adaptive filtered importance sampling to increase the memory access efficiency, thus reducing bandwidth demands.

Chapter 4 presents a single-pass adaptive sampling algorithm, single-pass variance guiding (SPVG), that guides the MC sampling of subsurface scattering in a local pass to enable real-time performance even with incoherent resource access in uv space. Moreover, the guiding algorithm decouples from the final temporal accumulation pass, enabling different temporal accumulation algorithms. We demonstrate that both TAA and DLSS are capable of performing this task. We believe this algorithm is general to be applied to other rendering passes and can be used for offline rendering to guide sample distribution.

Chapter 5 details the application of adaptive sampling for subsurface scattering and further optimizations. A simplified importance sampling function for Burley's Diffuse Reflectance Profile is provided, suitable for online sample generation. We have also combined adaptive sampling with filtered importance sampling to reduce the memory demands further. The direct scattering variance has been removed from the variance monitoring to make adaptive sampling more efficient. Since high-resolution gaming is also crucial for the next generation game, we have proposed a hybrid subsurface scattering framework to enable content-importance based rendering of subsurface scattering.

Chapter 6 proposes a *real-time control variates* algorithm to reduce sample count demands in real-time adaptive sampling in dynamic lighting and scenes. It minimizes both regions and sample demands to perform stable subsurface scattering with a unified representation. The CV coefficient is based on our novel exponential moving covariance matrix, which should help in all research and application domains that rely on an accurate online exponential moving covariance matrix. A bandwidth-friendly CV coefficient approximation is also provided for real-time rendering engine integration.

Chapter 7 provides the implementation and additional results. Apart from rendering quality and time, cycle level analysis is also performed to show the actual cycles saved and the effect on cache miss of adaptive sampling.

8.1 Future Works

In the future, we can explore deep learning methods for global subsurface scattering quality improvement other than the standard DLSS that we have already demonstrated. We can also perform multiple importance sampling to sample both profile and lighting for subsurface scattering. This could potentially further reduce the variance. Moreover, we can explore more complex in-frame standard control variates with low memory demand instead of the in-frame constant control variates.

Since the adaptive sampling technique is generic, we can explore the application of the adaptive sampling technique to other rendering passes (e.g., PCSS, glossy, ambient occlusion), and even path tracing.

To explore further heterogeneity. One potential method is to extend the level of details for memory demands to bit representation. For example, pixels newly dis-occluded tend to require much over-sampling in the real-time adaptive sampling framework. We can use textures in low bit representation to deal with the bandwidth demand increase. After the code start, we can switch back to the standard level of details with high bit representation. Moreover, we might use this work as a GPU performance test case where an adaptive sampling algorithm is used.

Furthermore, there is a need to study the under-estimation in Real-time Adaptive Sampling. Given a target quality, we expect the sample count for each pixel should be large enough to raise the variance higher than the quality level. However, an under-estimation of sample count could happen in real-time adaptive sampling, leading to the perceivable noise part visible to the human eye. Throughout the experiment, two potential sources of under-estimation are found during the informal study: 1) parameter constraints and 2) sample count filtering. The first cause happens before sample count estimation during the history update process. A good example is the residual lower bound limit $Res_{\min} \leq -a_0 \cdot \bar{g}$ for the CV coefficient boundary estimation. When a_0 is small, like 1, it can lead to variance under-estimation. Suppose the random sequence has a short period, e.g., based on local hash during dynamic lighting. A visible boundary would be perceivable between scattering and shadowed regions. The second case happens after the sample count map generation. Because the sample count map is not temporally constant, a common thought is to apply a spatial filter to make it more stable. We find that if a 3x3 median filter is applied, the sample count can be under-estimated, leading to visible noise. This noise is visible during static scenes. Another research direction is to answer whether adding a sample count filter helps to improve the efficiency.

In the implementation, we have used an 8-bit subsurface profile ID. It can only support 256 different profiles. A naive solution is just to increase the bits used. However, if we still want the current acceleration, we might need a clustering system to cluster profile IDs when there are more than 256 profiles.

8.2 Contributions

This dissertation makes the following contributions:

1. Propose a real-time adaptive sampling algorithm in a single pass with time and space complexity of O(1).

2. Provide the idea of local guiding that can use different global temporal reuse algorithms (e.g., TAA and DLSS) to improve the rendering quality. Although we do not have a contribution to create a new deep learning algorithm for the temporal accumulation, we demonstrate the high potential of using deep learning neural network for this task through a pre-existing deep learning technique DLSS designed for generic super sampling with the capability of temporal accumulation. It has higher quality than TAA under different lighting intensities.

3. Propose adaptive filtered importance sampling that further improves the rendering speed of the filtered importance sampling with little overhead. For subsurface scattering, adding the adaptive sampling part alone is shown to

provide additional 2× to 6× acceleration in addition to existing acceleration techniques.

4. We have also improved our proposed adaptive method under the dynamic changing environment with online control variates updating algorithm. The major enabler is the proposed exponential moving covariance matrix.
5. Introduce an importance-guided acceleration framework that allows all users to change different acceleration techniques (Separable and AFIS) by their game logic.
6. Fit a separable approximation to on Burley's normalized profile.
7. Accelerate the radius sampling with a well-fitted approximationto the inverse CDF.
8. Provide a taxonomy in terms of cache and bandwidth for heterogeneous real-time rendering. This dissertation has contributed to all sub-categories. We deal with the cache incoherence crisis of Monte Carlo sampling to enable high performance across current and next-gen commercial off-the-shelf GPU platforms. For bandwidth-limited tasks, if we can have an alternative algorithm to solve a problem that requires much less bandwidth demand, the performance can be increased even when the cache hit rate reduces.
9. Focus on and explore temporal sequence information per pixel to advance photo-realistic real-time rendering further. This dissertation provides a taxonomic and mathematical way of thinking to improve the quality and perfor-

mance given limited bandwidth and cache. It might bring up more possible research opportunities in real-time rendering from temporal sequence information.

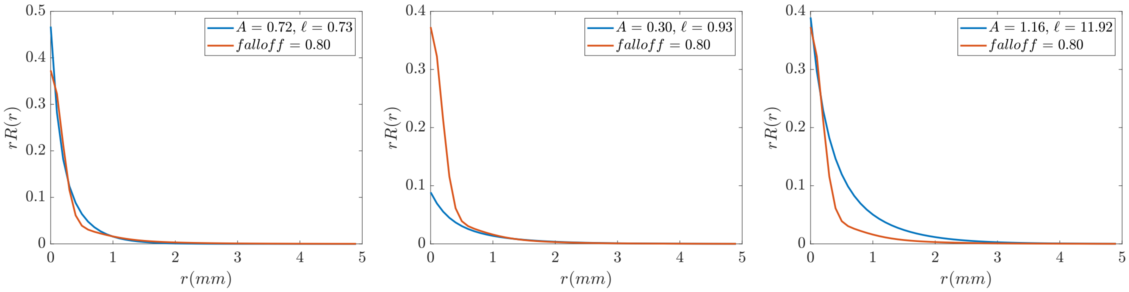
Appendix A: Separable Approximation to Burley's Model

To make sure that the separable filter implementation is valid and can be compared to our method, we provide our fitting from Gaussian kernel parameterization [Jimenez et al., 2015] to Burley based on non-linear least squares fitting, and also a comparison between these different techniques. Having a separable fit to the Burley model eases rendering upgrades where the separable model is already deployed, but also provides insight into how existing Gaussian kernel parameters interact with Burley's.

Note that this fitting is to make the separable and Burley models appear as similar as possible visually. This process itself is not physically meaningful. The original separable kernel is an approximation of the dipole model, which is already an approximation of the actual profile. We are trying to fit it to Burley's profile, which is a direct approximation of the actual profile.

A.1 Fitting

The fitting problem is to find the best parameter fitting between the Burley $R_B(r, \theta_B)$ and separable $R_S(r, \theta_S)$ diffuse profile, where $\theta_B = \{A, \ell\}$, and $\theta_S = \{\mathit{falloff}\}$ (Note that both model has already included the r term inside for the



(a) By Difference (S_Δ). (b) By Ratio (S_R). (c) By Derivative (S_D).

Figure A.1: Diffuse profile fitting result with three different methods.

fitting).To achieve this, we explore three error functions:

1. **Difference**. The direct difference between the Burley and separable curves. The minimization error function is

$$S_\Delta = \sum (R_B(r, \theta_B) - R_S(r, \theta_S))^2 \tag{A.1}$$

2. **Ratio**. The ratio between the Burley and separable curves should be close to one. The minimization error function is

$$S_R = \sum (R_B(r, \theta_B)/R_S(r, \theta_S) - 1)^2 \tag{A.2}$$

3. **Derivative**. The first derivative of both curves in terms of r should have high similarity. The minimization error function is

$$S_D = \sum (R'_B(r, \theta_B) - R'_S(r, \theta_S))^2 \tag{A.3}$$

We observed that S_Δ is the best error function for non-linear least squares

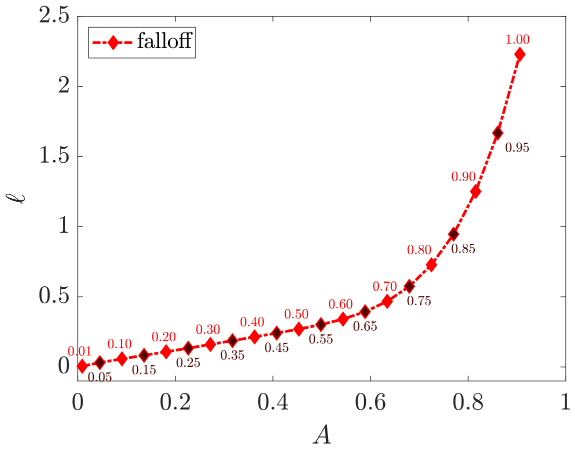
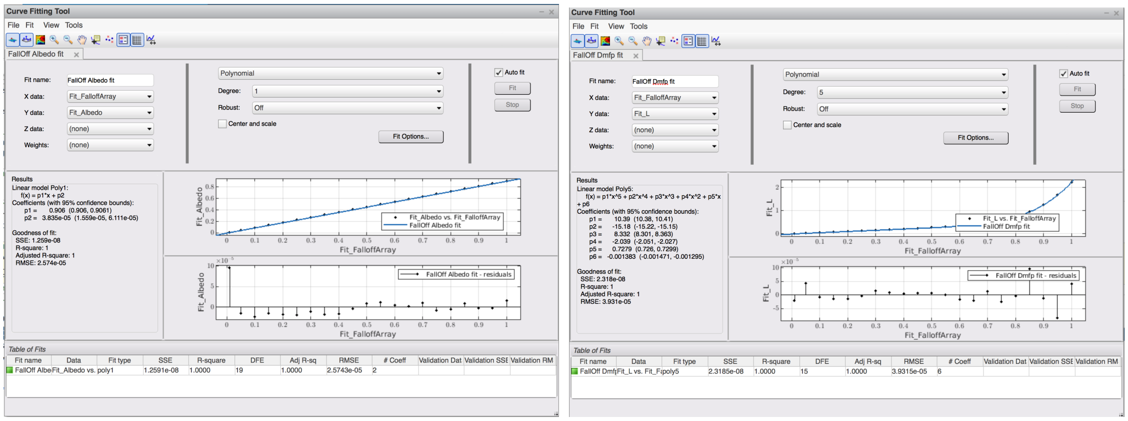


Figure A.2: The fitting from *falloff* color to Albedo (A) and DMFP (ℓ) with S_Δ.



```
Fit name:  FallOff Albedo fit
X data:    Fit_FalloffArray
Y data:    Fit_Albedo
Z data:    (none)
Weights:   (none)

Polynomial
Degree:  1
Robust:  Off
☐ Center and scale
☑ Auto fit

Results
Linear model Poly1:
    f(x) = p1*x + p2
Coefficients (with 95% confidence bounds):
      p1 =       0.906  (0.906, 0.9061)
      p2 =   3.835e-05  (1.559e-05, 6.111e-05)

Goodness of fit:
  SSE: 1.259e-08
  R-square: 1
  Adjusted R-square: 1
  RMSE: 2.574e-05
```

Fit name	Data	Fit type	SSE	R-square	DFE	Adj R-sq	RMSE	# Coeff	Validation Dat	Validation SSE	Validation RM
🟩 FallOff Albe	Fit_Albedo vs.	poly1	1.2591e-08	1.0000	19	1.0000	2.5743e-05	2			

(a) Albedo.

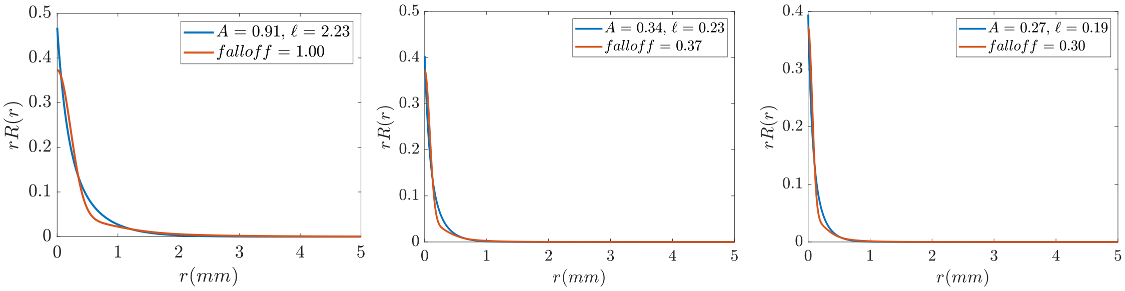
```
Fit name:  FallOff Dmfp fit
X data:    Fit_FalloffArray
Y data:    Fit_L
Z data:    (none)
Weights:   (none)

Polynomial
Degree:  5
Robust:  Off
☐ Center and scale
☑ Auto fit

Results
Linear model Poly5:
    f(x) = p1*x^5 + p2*x^4 + p3*x^3 + p4*x^2 + p5*x
+ p6
Coefficients (with 95% confidence bounds):
      p1 =       10.39  (10.38, 10.41)
      p2 =      -15.18  (-15.22, -15.15)
      p3 =       8.332  (8.301, 8.363)
      p4 =      -2.039  (-2.051, -2.027)
      p5 =      0.7279  (0.726, 0.7299)
      p6 =   -0.001383  (-0.001471, -0.001295)

Goodness of fit:
  SSE: 2.318e-08
  R-square: 1
  Adjusted R-square: 1
  RMSE: 3.931e-05
```

Fit name	Data	Fit type	SSE	R-square	DFE	Adj R-sq	RMSE	# Coeff	Validation Dat	Validation SSE	Validation RM
🟩 FallOff Dmf	Fit_L vs. Fit_F	poly5	2.3185e-08	1.0000	15	1.0000	3.9315e-05	6			

(b) DMFP

Figure A.3: Fitting details in matlab.

fitting visually. Fig. A.1 shows an example of the result with *falloff* = 0.8 for different error functions. Fig. A.2 shows the corresponding mapping between θ_B and θ_S. The corresponding fitting performed in matlab is illustrated in Figure A.3. Fig. A.2 clearly shows the range of values that can be expressed by the default separable parameterization and Burley's. To have the fitting on the fly in an engine, a LUT or a linear fitting function can be deployed.



(a) Red channel.
(b) Green channel.
(c) Blue channel.

Figure A.4: Diffuse profile fitting for the default Separable configuration in UE4.

A.2 Validation

We created a beam light scene where a 1 cm radius circle surface receives 10 flux. The subsurface profile configuration is based on the fitting shown in Fig. A.4. Subsurface color for the separable model [Jimenez et al., 2015] is set to 1. Non-subsurface processes like tone mapping, bloom, eye adaptation, specular, and auto exposure are turned off. The result is shown in Fig. A.5. We compared the separable and Burley model in different configurations and with different diffuse mean free paths (ℓ). Due to the low sampling count, the separable and Burley models have different artifacts when ℓ increases. The separable model has banding artifacts while Burley has energy loss due to TAA clamping. We believe better clamping in TAA could further reduce the energy loss.

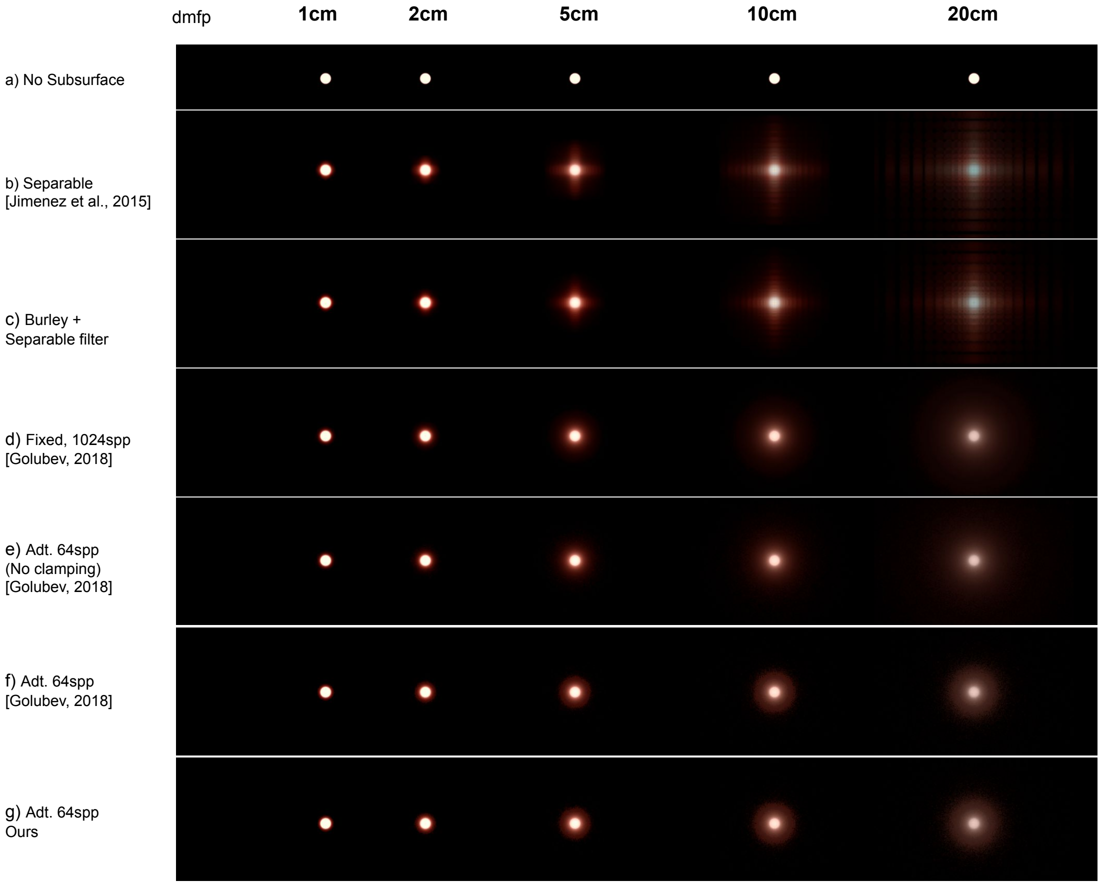


Figure A.5: Subsurface test for the beam light scene with a 1 *cm* circle receiving 10 flux. a) No subsurface. b) Separable [Jimenez et al., 2015]. c) Burley profile fit with a separable filter. d) Burley with fixed 1024 spp. e) Burley with adaptive 64 spp + no history clamping. f) Burley with adaptive 64 spp. g) Burley with adaptive 64 spp using our approximation. Halley's method introduced in [Golubev, 2018] is used in d)-f) for sampling. TAA is on for all tests.

Appendix B: Control Variates

B.1 Exponential moving covariance

To assure that the exponential moving covariance matrix is correct, we first prove the basic building block that exponential moving covariance is correct.

B.1.1 Variable-weight covariance

The derivation is based on [Finch, 2009]. For exponential moving average and variance, please refer to the original paper. Denote n observations $x_1, ..., x_n$ and $y_1, ..., y_n$ from two random variable x, y. The weighted means for x and y are:

$$\mu_n = \frac{\sum_{i=1}^n w_{n,i} x_i}{\sum_{i=1}^n w_{n,i}}, \upsilon_n = \frac{\sum_{i=1}^n w_{n,i} y_i}{\sum_{i=1}^n w_{n,i}} \tag{B.1}$$

where the weights sum as

$$W_n = \sum_{i=1}^n w_{n,i}. \tag{B.2}$$

To keep

$$W_n\mu_n - w_{n,n}x_n = (W_n - w_{n,n})\mu_{n-1}, \tag{B.3}$$

$$W_n\upsilon_n - w_{n,n}y_n = (W_n - w_{n,n})\upsilon_{n-1} \tag{B.4}$$

we have the following constraint between W_n and W_{n-1}:

$$\frac{w_{n,j}}{\sum_{i=1}^{n-1} w_{n,i}} = \frac{w_{n-1,j}}{\sum_{i=1}^{n-1} w_{n-1,i}}, 1 \leq j \leq n-1. \tag{B.5}$$

Then we can derive from the covariance between x and y

$$\begin{aligned}Cov_n(x,y) =& E_n((x-\mu_n)(y-\upsilon_n))\\ =& \frac{1}{W_n}\sum_{i=1}^{n} w_{n,i}(x_i-\mu_n)(y_i-\upsilon_n)\end{aligned}$$

with $f_n(x,y) = (x-\mu_n)(y-\upsilon_n)$ and Eq. B.5 that

$$W_nE_n(f_{n-1}(x,y)) = \sum_{i=1}^{n} w_{n,i}f_{n-1}(x_i,y_i) \tag{B.6}$$

$$= w_{n,n}f_{n-1}(x_n,y_n)+(W_n - w_{n,n})\frac{\sum_{i=1}^{n-1} w_{n-1,i}f_{n-1}(x_i,y_i)}{W_{n-1}} \tag{B.7}$$

$$= w_{n,n}f_{n-1}(x_n,y_n)+(W_n - w_{n,n})E_{n-1}(f_{n-1}(x,y)). \tag{B.8}$$

With $S_n = W_n Cov(x, y)$ we can have:

$$S_n = W_n E_n([x - \mu_n][y - \upsilon_n]) \tag{B.9}$$

$$= W_n E_n(([x - \mu_{n-1}] - [\mu_n - \mu_{n-1}])([y - \upsilon_{n-1}] - [\upsilon_n - \upsilon_{n-1}])) \tag{B.10}$$

$$= W_n E_n([x - \mu_{n-1}][y - \upsilon_{n-1}]) + W_n E_n([\mu_n - \mu_{n-1}][\upsilon_n - \upsilon_{n-1}])$$

$$- W_n E_n([\mu_n - \mu_{n-1}][y - \upsilon_{n-1}]) - W_n E_n([x - \mu_{n-1}][\upsilon_n - \upsilon_{n-1}]) \tag{B.11}$$

where the 3rd and 4th term can be simplified as

$$W_n E_n([\mu_n - \mu_{n-1}][y - \upsilon_{n-1}]) = W_n[\mu_n - \mu_{n-1}][\upsilon_n - \upsilon_{n-1}] \tag{B.12}$$

$$W_n E_n([x - \mu_{n-1}][\upsilon_n - \upsilon_{n-1}]) = W_n[\mu_n - \mu_{n-1}][\upsilon_n - \upsilon_{n-1}]. \tag{B.13}$$

The first term can be simplified with Eq. B.8 as

$$W_n E_n([x - \mu_{n-1}][y - \upsilon_{n-1}]) = \tag{B.14}$$

$$= w_{n,n}[x_n - \mu_{n-1}][y_n - \upsilon_{n-1}] + \frac{W_n - w_{n,n}}{W_{n-1}} S_{n-1} \tag{B.15}$$

$$= \frac{W_n^2}{w_{n,n}}[\mu_n - \mu_{n-1}][\upsilon_n - \upsilon_{n-1}] + \frac{W_n - w_{n,n}}{W_{n-1}} S_{n-1} \tag{B.16}$$

After summing up the four terms, it leads to

$$S_n = \frac{W_n - w_{n,n}}{W_{n-1}} S_{n-1} + w_{n,n}(y_n - \upsilon_n)(x_n - \mu_{n-1}). \tag{B.17}$$

Then, the variable-weight covariance has $Cov_n(x, y) = S_n / W_n$

B.1.2 Exponential moving covariance

In exponential moving average, we have the general form

$$\mu_n = (1-\alpha)^n x_0 + \sum_{i=1}^{n} (1-\alpha)^{n-i} \alpha x_i. \tag{B.18}$$

Therefore, we have $w_{n,i} = (1-\alpha)^{n-i}\alpha, 1 \leq i \leq n$, $w_{n,n} = \alpha$ and $W_n = W_{n-1} = 1$. Then, we have

$$\begin{aligned} S_n =& \frac{W_n - w_{n,n}}{W_{n-1}} (S_{n-1} + \frac{W_{n-1} w_{n,n}}{W_n - w_{n,n}} (y_n - \upsilon_n)(x_n - \mu_{n-1})) \\ =& (W_n - w_{n,n})(S_{n-1} + w_{n,n}(y_n - \upsilon_{n-1})(x_n - \mu_{n-1})). \end{aligned} \tag{B.19}$$

Since we have

$$Cov_n(x, y) = \frac{S_n}{W_n} = S_n, \tag{B.20}$$

the final simplified exponential moving covariance is

$$Cov_n(x, y) = (1-\alpha) Cov_{n-1}(x, y) + \alpha(1-\alpha)(x_n - \mu_{n-1})(y_n - \upsilon_{n-1}). \tag{B.21}$$

When $n = 0$, we have $Cov_n(x, y) = 0$.

B.1.3 EWMA covariance estimator

Note that Eq. B.15 leads to the EWMA covariance estimator as

$$\widetilde{Cov}_n(x,y) = (1-\alpha)\widetilde{Cov}_{n-1}(x,y) + \alpha(x_n - \mu_{n-1})(y_n - \upsilon_{n-1}) \tag{B.22}$$

where the prediction is done with

$$\widetilde{Cov}_n(x,y) \approx W_n E_n((x - \mu_{n-1})(y - \upsilon_{n-1})) \tag{B.23}$$

while ignoring the contribution of all other three terms in Eq. B.11. The formula we derived Eq.B.21 is exactly

$$Cov_n(x,y) = W_n E_n((x - \mu_n)(y - \upsilon_n)) \tag{B.24}$$

B.2 Covariance of Two Batch Means

Let n pair of random variables X_i,Y_j sampled from two different distribution correlated only when $i = j$. And $\bar{X} = \frac{1}{n}\sum X_i$, and $\bar{Y} = \frac{1}{n}\sum Y_j$. Then:

$$\begin{aligned} Cov(\bar{X},\bar{Y}) &= Cov(\frac{1}{n}\sum X_i, \frac{1}{n}\sum Y_j) && \text{(B.25)} \\ &= \frac{1}{n^2}\sum_i \sum_j Cov(X_i, Y_j) && \text{(B.26)} \end{aligned}$$

Since X_i, Y_j are correlated only when $i = j$,

$$Cov(\bar{X}, \bar{Y}) = \frac{1}{n^{\cancel{2}}} \cdot \cancel{n} Cov(X, Y) \tag{B.27}$$

B.3 CV Coefficient and Residual Function Selection

In this section we provide an example of how we arrived at the CV coefficient, the updating function, and the residual function for variance tracking if CV requires Monte Carlo sampling. Note that the major purpose of having a variance estimation is for real-time adaptive sampling. The temporal variance should have minimal impact on it. Otherwise, the sample count will be greatly increased. Since it's real-time rendering, we also cannot afford a lot of textures to store history information to approximate best coefficient.

B.3.1 Scenario Setup

We have a spatial-temporal variant function to integrate $f(y, t) = h(t) \cdot 1/(1 + y)$, with the control variable as $g(y, t) = h(t) \cdot (1 - y)$, $\mathcal{D} \in [0, 1]$. Then $G(t) = \int_{\mathcal{D}} g(y, t) dy = 0.5 \cdot h(t)$. For simplicity, 100 frame times are performed and a fixed number of samples 16 spp are used for the correlated Monte Carlo sampling of f and g to get the estimation $\bar{f}(t)$ and $\bar{g}(t)$. The sample count is fixed to help understand how our method helps to mitigate temporal variance. We have the following five configurations for the residual and CV coefficient estimation.

1. $Res_{t,\bar{f}}$. No CV is applied. The CV coefficient is 0. Therefore $Res_{t,\bar{f}} = \bar{f}(x)$.

This configuration demonstrates the impact of temporal variance.

2. $Res_{t,SF}$. We estimate the CV with single frame covariance between f and g. Then it is used as the CV coefficient $a_{t-1,SF}$ for the next frame to calculate the residual as $Res(t) = \bar{f}(x)$-$a_{t-1,SF} \cdot \bar{g}(x)$.

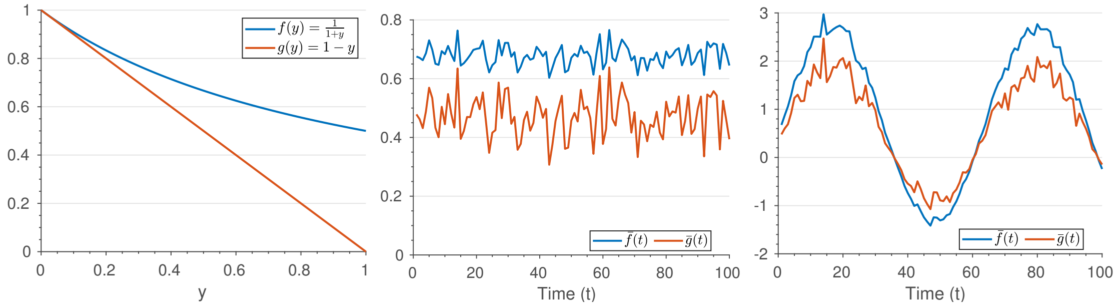
3. $Res_{t,SF+EMA}$. After estimating $a_{t-1,SF}$, exponential moving average is used to get a better estimation of the coefficient over time. Then $a_{t-1,SF+EMA}$ are used to estimate the residual.

4. $Res_{t,EMCM}$. The covariance matrix $\mathbf{\Sigma}_t$ between $\bar{f}(x)$ and $\bar{g}(x)$ is updated with EMCM over time with a weighting factor $\alpha_{cov} = 0.05$. Then the CV coefficient is $a_{t-1,EMCM}$ based on $\mathbf{\Sigma}_{t-1}$.

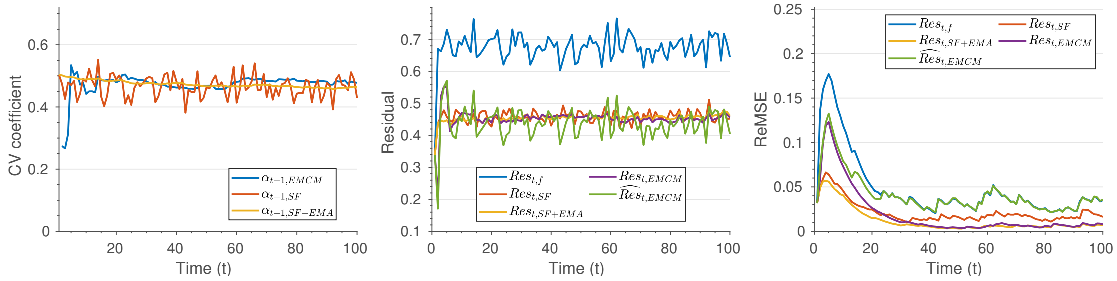
5. $\widehat{Res}_{t,EMCM}$. The CV coefficient is the same as $a_{t-1,EMCM}$. However, the residual is calculated as $\widehat{Res}_{t,EMCM} = \bar{f}(x) - a_{t-1,EMCM}G(x)$. The reason we use $G(x)$ instead of $\bar{g}(y)$ is that CV are used to deal with temporal variance, but should not affect the variance estimation of spatial variance for adaptive sampling (please see the result section for a better understanding).

With this configuration, we monitor the exponential moving variance on the residual with $\alpha_{res} = 0.2$. In the meanwhile, we also have the ground truth configuration Res_{gt}, which is to apply the optimal CV coefficient. The coefficient is estimated beforehand with large sample counts as ground truth.

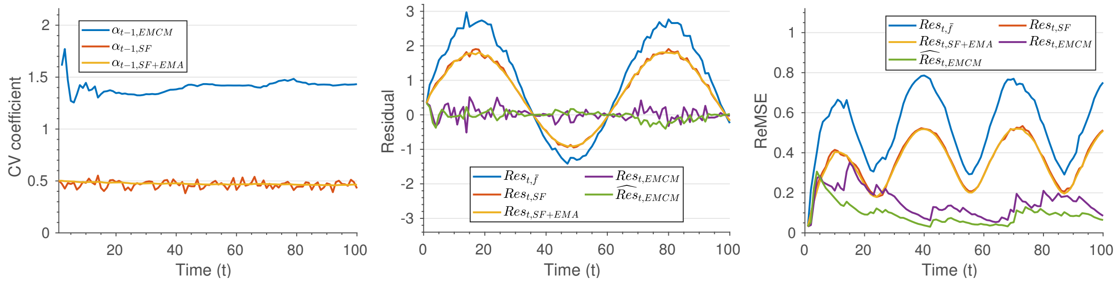
We use two temporal function, $f_1(y,t)$ with $h_1(t) = 1$ and $f_2(y,t)$ with $h_2(t) = 1 + 3 * \sin(\frac{t}{10})$ to simulate both static and dynamic scenarios.



(a) f and g (b) Static (c) Dynamic



(d) h_1 CV coefficient (e) h_1 Residual (f) h_1 Error



(g) h_2 CV coefficient (h) h_2 Residual (i) h_2 Error

Figure B.1: The CV coefficient and variance under different configuration with 16 samples per frame time.

Table B.1: The ReMSE of five configurations and the groud truth Res_{GT}. $\widehat{Res}_{t,EMCM}$ is the best choice based on our selection criteria: 1) similar ReMSE to $Res_{t,\bar{f}}$ in static scene, 2)mitigating temporal variance in dynamic scene, yet the ReMSE is no less than the static scene counterpart of $\widehat{Res}_{t,EMCM}$ or $Res_{t,\bar{f}}$.

	$Res_{t,\bar{f}}$	$Res_{t,SF}$	$Res_{t,SF+EMA}$	$Res_{t,EMCM}$	$\widehat{Res}_{t,EMCM}$	Res_{GT}
Static ($h = h_1$)	0.0325	0.0160	0.0058	0.0070	**0.0324**	0.0055
Dynamic ($h = h_2$)	0.5585	0.3797	0.3742	0.1295	**0.0732**	0.1243

B.3.2 Result

Static scenarios. Fig. B.1(a) shows the spatial function of f and g when $h = h_1$. Fig. B.1(b) shows the corresponding MC sampling result at each frame time. Fig.

B.1(d) shows the estimated optimal coefficient for different configurations. The optimal coefficient based on a MC sampling of 100k samples has $a^* \approx 0.477$. Fig. B.1(e) shows the corresponding residual after applying CV. Fig. B.1(f) shows the root exponential mean square error (ReMSE), which is applying a root operation on the exponential moving variance. Although it is straight forward that CV can reduce the variance estimation, however, please note that the reason we apply CV is to remove temporal dimension variance, it should not significantly affect the spatial variance estimation, otherwise, we could under estimate the number of samples in static scenes. Since our application is in sample domain instead of shading domain, by utilizing a good configuration like $Res_{t,SF+EMA}$ or $Res_{t,EMCM}$ will lead to under sampling in shading domain as the constant term $a \cdot G(x)$ is not added in shading domain. Because of this, $\widehat{Res}_{t,EMCM}$ is the best choice as it has almost the same ReMSE as $Res_{t,\bar{f}}$, which is not applying CV. Table B.1 shows the average ReMSE of frame from 25 to 100 (to not count cold start).

Dynamic scenarios. With $h = h_2$, the problem is different as temporal variance is introduced. Fig. B.1(c) shows the corresponding MC sampling result at each frame time. We want to minimize the temporal variance. The CV coefficient should be derived from the covariance matrix of temporal domain t instead of spatial domain y as $Cov(\bar{f}_2(x), \bar{g}_2(x))$. With 1024 samples per frame, we estimate the optimal CV coefficient $a^* \approx 1.388$. Fig. B.1(g) shows the run-time estimation of the coefficient. Fig. B.1(h) shows the residual of all configurations. Fig. B.1(i) shows the ReMSE. Here we could observe that the spatial coefficient $a_{t-1,SF}$ and $a_{t-1,SF+EMA}$ can reduce the variance but not the temporal variance. With $Res_{t,EMCM}$ and $\widehat{Res}_{t,EMCM}$, most of the temporal variance has been removed. Table B.1 shows the mean ReMSE. It looks like both are viable for our adaptive sampling purpose. However, $Res_{t,EMCM}$ causes under sampling during static scene. Therefore, the configuration that is suitable for our use is $\widehat{Res}_{t,EMCM}$.

In a summary, the CV coefficient is updated with the EMCM. In side the residual function, $G(x)$ instead of $\bar{g}(y)$ is selected to allow CV deal with temporal variance, but do not significantly affect the estimation of spatial variance for adaptive sampling.

B.4 Theoretical Foundation

In this section, we provide the theoretical foundation for:

1. why the control variates updating function works based on EMCM.
2. the relationship among CV coefficient, the expectation and variance of CV residual under different lighting condition.
3. time-variant CV coefficient with in-frame constant control variable.

B.4.1 Updating Function

Generally, if we have three random variables T, F, and G, where $E[TG]$ is a known constant. Then the unbiased Monte Carlo estimator for TF

$$\langle TF \rangle = aE[TG] + \langle TF - a \cdot TG \rangle \tag{B.28}$$

has the minimal variance based on the control variates concept where the optimal control variates coefficient is

$$a^* = \frac{Cov(TF, TG)}{Var(TG)}, \tag{B.29}$$

which can also be derived with the partial derivates of $Var(\langle TF \rangle)$ according to a as

$$Var(\langle TF \rangle) = Var(\langle TF \rangle) + a^2 \cdot Var(\langle TG \rangle) - 2a\,Cov(TF, TG) \tag{B.30}$$

In subsurface scattering, T is the temporal intensity changing term. TF is the joint random variable of temporal intensity and the subsurface scattering function, while TG the one of temporal intensity change and the pre-integration irradiance function. Please note that T might not be independent from F and G, and the point of interest is *per pixel*. Namely, T, F, and G are functions on pixel level. In the adaptive sampling framework, they are scalar, the luminance.

B.4.1.1 Time-invariant Scenario

In time-Invariant scenario, lighting intensity is temporally stable ($E[T]$ is constant and $Var(T) = 0$). Therefore, Eq. B.29 is simplified to

$$a_s^* = \frac{Cov(F, G)}{Var(G)}, \tag{B.31}$$

which indicates we can use any method to use temporal samples to improve the estimate of CV coefficient. Note that Eq. B.31 and Eq. B.30 is only true when $E[TG]$ can be derived analytically. Otherwise, the formulation is more complex [Rousselle et al., 2016]. As illustrated in our experimental example shown in Fig. B.1(d), applying EMCM to estimate CV coefficient ($a_{t-1,EMCM}$) is equivalent to applying EMA on single frame estimation ($a_{t-1,SF+EMA}$). They all converge to $a^* \approx 0.477$.

B.4.1.2 Time-variant Scenario

When the lighting is not temporal stable, the optimal stable CV coefficient by Eq. B.31 does not hold as a valid alternative as a^*. We have to fall back to the original Eq. B.29 instead. The temporal information in T has to be used for good estimation. This is why $a_{t-1,SF+EMA}$ fails the purpose. What it does is to estimate the in-frame CV coefficient a_s^*. Because inside each frame, the time-invariant property holds. It only uses temporal history to improve the a_s^* estimation instead of considering the temporal random variable T, the temporal unstable intensity. Due to the time-variant feature of T, we are unable to know the future of $E[TG]$ in real-time rendering beforehand. However EMCM provides the capability to focus most recent changes to estimate temporal unstable CV coefficient, leading to a different coefficient. For example, the one as shown in Fig. B.1(g) ($a^* \approx 1.388$), where $\int_{\mathcal{D}} \frac{1}{1+y} - a^*(1-y)dy = -8.53 \times 10^{-4}$, and the variance remains resistant to temporal change (Fig. B.1(i)).

B.4.2 CV Coefficient, Mean and Variance of CV Residual

To provide a better understanding of the effect of an additional temporal term on CV (the relationship among a^*, the variance and mean of CV residual), we provide an analysis of a general case in real-time rendering where T is independent from F and G. This assumption is universal in real-time rendering where nearly all types of lights have an intensity term and there is one major light changing intensity in subsurface scattering range. As long as it is still subsurface scattering, the intensity change will not cause the change of the subsurface scattering and the pre-integrated irradiance function when the history is properly projected.

Generally in statistics, if we have three random variables T, F, G and a constant parameter a, where T is independent from both F and G. We then have the variance function for the residual as

$$\begin{aligned} Var(TF - aTG) &= Var(T(F - aG)) && \text{(B.32)} \\ &= E[T]^2 Var(F - aG) + E[T]^2 Var(F - aG) + Var(T)E[F - aG]^2 && \text{(B.33)} \end{aligned}$$

B.4.2.1 Stable Lighting

During stable lighting ($E[T]^2 \gg Var(T)$), the intensity $E[T]$ becomes constant and the variance term zero. Then Eq. B.33 simplifies to

$$Var_{E[T]^2 \gg Var(T)}(TF - aTG) = E[T]^2\, Var(F - aG) \tag{B.34}$$

where the only variance contribution is $Var(F - aG)$. Note that the time-invariant part of CV residual, $E[F - aG]$, is masked by $Var(T) = 0$. Therefore normally, CV coefficient does not care the expectation of the residual $E[F - aG]$. It can be any value. The CV coefficient is just to reduce the variance part $Var(F - aG)$ of the residual.

B.4.2.2 Dynamic Lighting

With T being time-variant. $Var(T)$ and $E[T]^2$ can be both non-zero and uncontrollable by our algorithm. The only controllable terms in our algorithm are the expectation and variance of the redisual term, $Var(F - aG)$ and $E[F - aG]$. Moreover, the variance term $Var(T)$ could be *arbitrarily large* in game with continuous gunshots, lightning, and rhythmic lighting in nightclub ($Var(T) \gg E[T]^2$), then

$$\begin{aligned} Var_{Var(T) \gg E[T]^2}(TF - aTG) &= Var(T)(Var(F - aG) + E[F - aG]^2) && \text{(B.35)} \\ &= Var(T)E[(F - aG)^2] && \text{(B.36)} \end{aligned}$$

Since Eq. B.30 derived the optimal CV coefficient solution to minimize the total variance analytically, an alternative representattion based on Eq. B.36 is

$$a^* = \underset{a}{\operatorname{argmin}}\, E[(F - aG)^2], \tag{B.37}$$

where $Var(T)$ is removed as it does not affect the optimization. This formulation is least square estimation. When $a = a^*$, we have $E[(F - aG)^2]$ minimized.

B.4.3 In-frame Constant Control Variable

We try to resolve the temporal stability issue in *real-time* adaptive sampling. Therefore being real-time is important. We cannot afford samples to sample both TF and TG if they all need to access memory, especially if the memory access pattern is incoherent. Monte Carlo sampling makes it worse. Because of this, TG is accessed as cheaply as possible without Monte Carlo sampling, we use the analytic solution (e.g., $G(x) = 0.5 \cdot h(t)$). Namely, we have the following assumption:

$$TG = E[G]T \text{ and } Var(G) = 0. \tag{B.38}$$

B.4.3.1 Stable Lighting

With the given assumption in Eq. B.38, the time-invariant CV coefficient updating formula (Eq. B.31) does not hold anymore and the estimator is

$$\langle TF \rangle = TF - a \cdot (\cancel{TG - E[TG]}), \tag{B.39}$$

where a is not meaningful and can be any number. If T is still independent, the residual variance is

$$Var(TF - aTG) = Var(\langle TF \rangle) = Var(F). \tag{B.40}$$

B.4.3.2 Dynamic Lighting

During dynamic lighting, the optimal CV coefficient Eq. B.29 simplifies to

$$a^* = \frac{Cov(TF, T)}{E[G]\,Var(T)}, \tag{B.41}$$

which means, even with a constant in-frame control variable, CV is still valid and working in temporal domain. This is key to resolve the temporal stability issue in real-time adaptive sampling.

T is independent. If we can further assume that F and T are independent, Eq. B.41 can be as simple as

$$a^* = \frac{E[F]}{E[G]}. \tag{B.42}$$

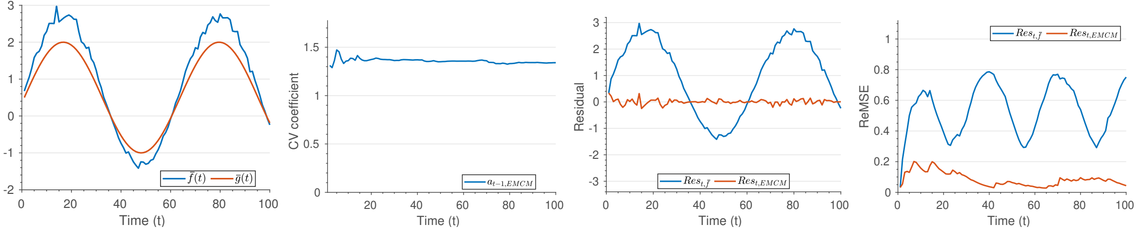
With the optimal CV coefficient, the time-invariant part of the residual becomes zero as

$$E[F - a^*G] = 0. \tag{B.43}$$

More specifically, $\int f(y) - a^* g(y) dy = 0$, and the variance Eq. B.33 is simplified to

$$Var(TF - aTG) = \left(Var(T) + E[T]^2 \right) Var(F). \tag{B.44}$$

To help the understanding of this concept, we repeated the same experiment shown in Section B.3. In this example the F random variable $f(y) = \frac{1}{1+y}$ and the G random varriable $g(y) = 1 - y$ is independent from the time-variant variable T, $h_2(t)$. The in-frame constant control variate uses $TG = E[G]T = 0.5 \cdot h_2(t)$ directly. The result is illustrated in Fig. B.2.



(a) Dynamic (b) h_2 CV coefficient (c) h_2 Residual (d) h_2 Error

Figure B.2: The CV coefficient and variance with time independent in-frame constant control variable.

B.5 More Static Lighting Insights

Fig. B.3 shows more temporal insights of the recorded history information and the derived CV coefficient with $\epsilon_a = 10^{-6}, a_{cv} = 0.005$ for the Eq. 6.21.

B.6 Additional Images

Fig. B.4 shows a qualitative comparison between with no subsurface scattering, Separable [Jimenez et al., 2015] and ours.

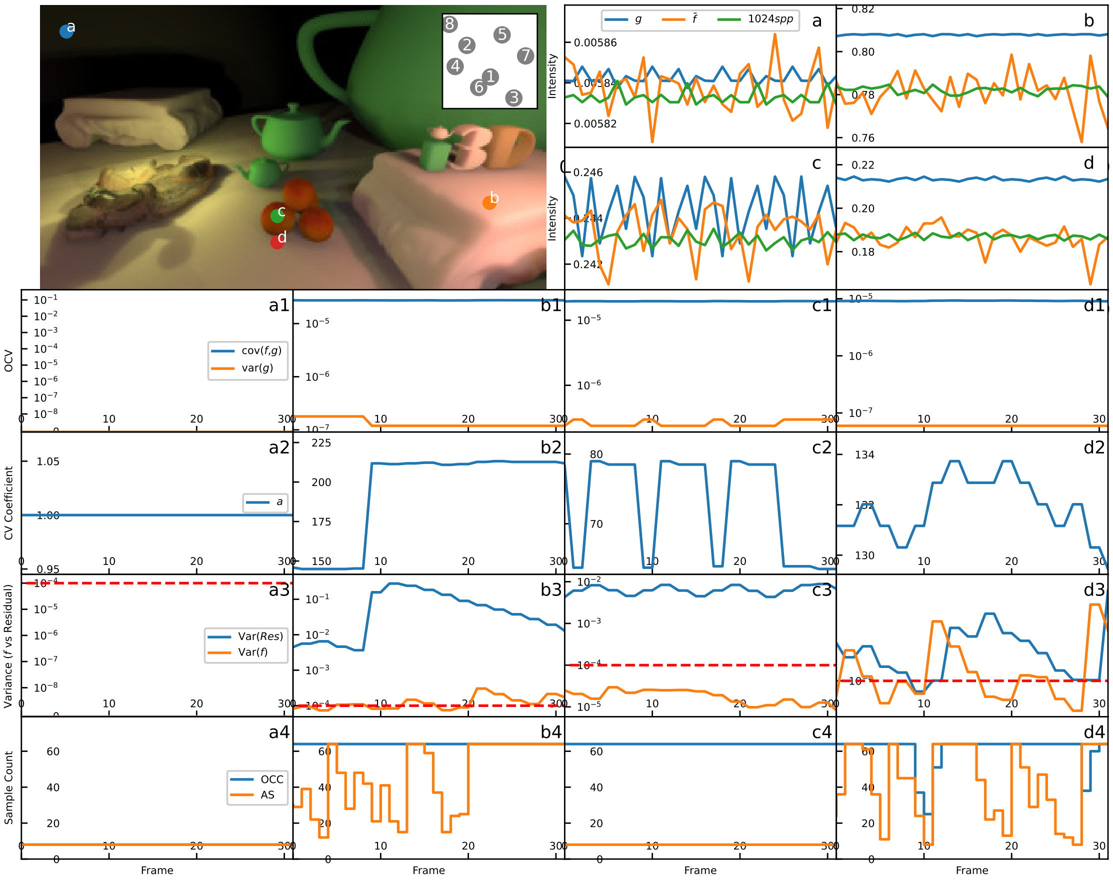
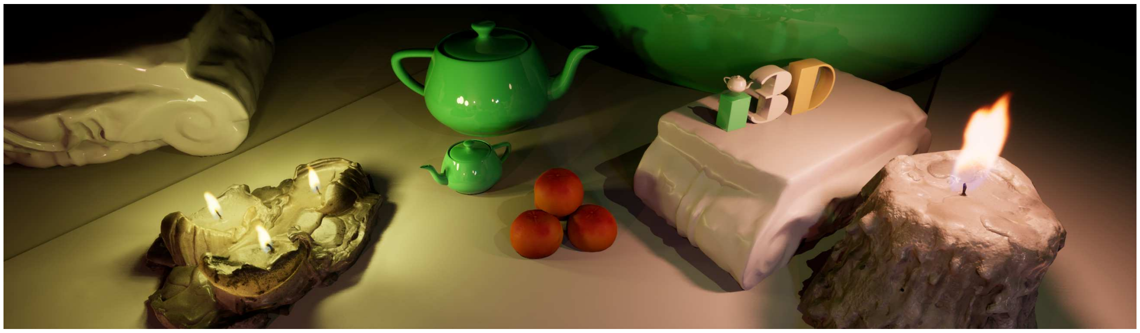
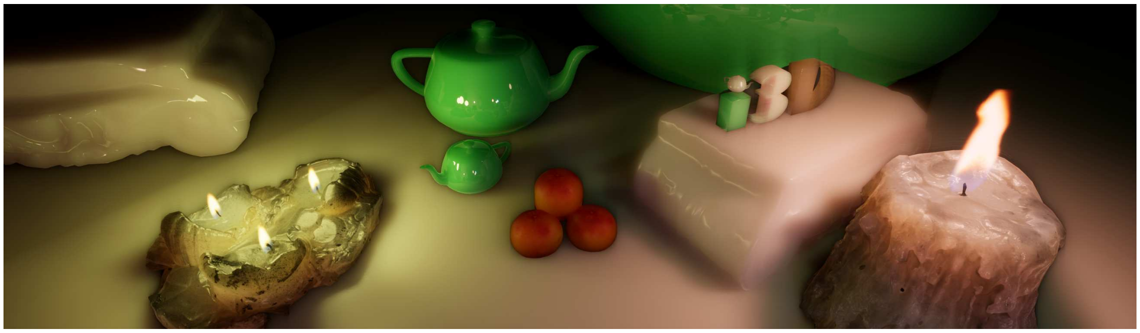


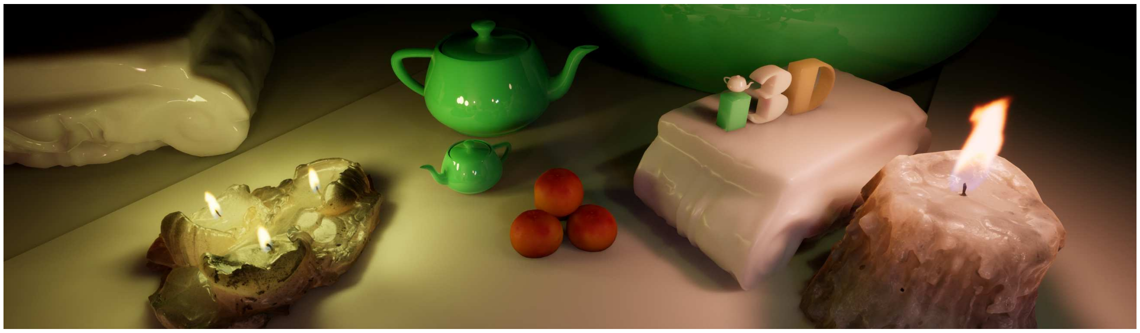
Figure B.3: Static lighting temporal insight with Control Variates for four pixels.



(a) Without subsurface scattering



(b) Separable



(c) Ours

Figure B.4: Additional qualitative comparison between without subsurface scattering, Separable and Ours.

Appendix C: Common Material Parameters for Burley's Normalized Profile

The diffuse mean free path parameters are directly computed from experiment data from [Jensen et al., 2001] with the assumption that light are evenly scattered ($g = 0$) in the isotropic media composed of the corresponding materials. The result is shown in Table C.1.

Table C.1: The measured parameters and the corresponding diffuse mean free path ℓ_d when $g = 0$.

Material	$\sigma_s{}'[mm^{-1}]$			$\sigma_a[mm^{-1}]$			$\ell_d[mm](g=0)$			Diffuse Reflectance (A)			η
	R	G	B	R	G	B	R	G	B	R	G	B	
Apple	2.29	2.39	1.97	0.0030	0.0034	0.046	6.97	6.40	1.92	0.85	0.84	0.53	1.3
Chicken1	0.15	0.21	0.38	0.015	0.077	0.19	12.12	4.37	2.23	0.31	0.15	0.10	1.3
Chicken2	0.19	0.25	0.32	0.018	0.088	0.20	9.84	3.76	2.16	0.32	0.16	0.10	1.3
Cream	7.38	5.47	3.15	0.0002	0.0028	0.0163	15.03	4.67	2.57	0.98	0.90	0.73	1.3
Ketchup	0.18	0.07	0.03	0.061	0.97	1.45	5.33	0.80	0.56	0.16	0.01	0.00	1.3
Marble	2.19	2.62	3.00	0.0021	0.0041	0.0071	8.51	5.57	3.96	0.83	0.79	0.75	1.5
Potato	0.68	0.70	0.55	0.0024	0.0090	0.12	14.29	7.27	2.31	0.77	0.62	0.21	1.3
Skimmilk	0.70	1.22	1.90	0.0014	0.0025	0.0142	18.44	10.45	3.51	0.81	0.81	0.69	1.3
Skin1	0.74	0.88	1.01	0.032	0.17	0.48	3.75	1.47	0.79	0.44	0.22	0.13	1.3
Skin2	1.09	1.59	1.79	0.013	0.070	0.145	4.85	1.11	0.62	0.63	0.44	0.34	1.3
Spectralon	11.6	20.4	14.9	0.00	0.00	0.00	-	-	-	1.00	1.00	1.00	1.3
Wholemilk	2.55	3.21	3.77	0.0011	0.0024	0.014	10.90	6.58	2.54	0.91	0.88	0.76	1.3

Bibliography

Sameer Agarwal, Ravi Ramamoorthi, Serge Belongie, and Henrik Wann Jensen. Structured Importance Sampling of Environment Maps. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, page 605612, New York, NY, USA, 2003. Association for Computing Machinery. ISBN 1581137095. doi: 10.1145/1201775.882314. URL `https://doi.org/10.1145/1201775.882314`.

C Alexader. Risk management and analysis. volume 1: Measuring and modelling financial risk, 1999.

AMD. Introducing RDNA Architecture, 2019. URL `https://www.amd.com/system/files/documents/rdna-whitepaper.pdf`.

AMD. RX 6800 graphics card, Nov 2020. URL `https://www.amd.com/en/products/graphics/amd-radeon-rx-6800`.

Mahdi M Bagher, Cyril Soler, Kartic Subr, Laurent Belcour, and Nicolas Holzschuch. Interactive rendering of acquired materials on dynamic geometry using bandwidth prediction. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 127–134, 2012.

Ali Bakhoda, George L Yuan, Wilson WL Fung, Henry Wong, and Tor M Aamodt. Analyzing CUDA workloads using a detailed GPU simulator. In *2009 IEEE International Symposium on Performance Analysis of Systems and Software*, pages 163–174. IEEE, 2009.

Colin Barré-Brisebois, Henrik Halén, Graham Wihlidal, Andrew Lauritzen, Jasper Bekkers, Tomasz Stachowiak, and Johan Andersson. Hybrid rendering for real-time ray tracing. In *Ray Tracing Gems*, pages 437–473. Springer, 2019.

Thiago Bastos and Waldemar Celes. GPU-accelerated adaptively sampled distance fields. In *2008 IEEE International Conference on Shape Modeling and Applications*, pages 171–178. IEEE, 2008.

Alex Battaglia. Inside Unreal Engine 5: how Epic delivers its generational leap, May 2020. URL `https://www.eurogamer.net/articles/digitalfoundry-2020-unreal-engine-5-playstation-5-tech-demo-analysis`.

Richard J Bauer and Julie R Dahlquist. *Technical Markets Indicators: Analysis & Performance*, volume 64. John Wiley & Sons, 1998.

Laurent Belcour, Cyril Soler, Kartic Subr, Nicolas Holzschuch, and Fredo Durand. 5D covariance tracing for efficient defocus and motion blur. *ACM Transactions on Graphics (TOG)*, 32(3):1–18, 2013.

Laurent Belcour, Kavita Bala, and Cyril Soler. A local frequency analysis of light scattering and absorption. *ACM Transactions on Graphics (TOG)*, 33(5):1–17, 2014.

Jakub Boksansky, Michael Wimmer, and Jiri Bittner. *Ray Traced Shadows: Maintaining Real-Time Frame Rates*, pages 159–182. Apress, Berkeley, CA, 2019. ISBN 978-1-4842-4427-2. doi: 10.1007/978-1-4842-4427-2_13. URL https://doi.org/10.1007/978-1-4842-4427-2_13.

Mark R. Bolin and Gary W. Meyer. A perceptually based adaptive sampling algorithm. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, page 299309, New York, NY, USA, 1998. Association for Computing Machinery. ISBN 0897919998. doi: 10.1145/280814.280924. URL https://doi.org/10.1145/280814.280924.

George Borshukov and John P Lewis. Realistic human face rendering for" The Matrix Reloaded". In *ACM SIGGRAPH 2005 Courses*, page 13, New York, NY, 2005. ACM.

Brent Burley. Extending the Disney BRDF to a BSDF with integrated subsurface scattering. In *SIGGRAPH Course: Physically Based Shading in Theory and Practice*, New York, NY, 2015. ACM.

Per H Christensen. An approximate reflectance profile for efficient subsurface scattering. In *ACM SIGGRAPH 2015 Talks*, page 25, New York, NY, 2015. ACM.

Per H. Christensen and Brent Burley. Approximate Reflectance Profiles for Efficient Subsurface Scattering. Technical report, Pixar, 2015.

Thomas F Coleman and Yuying Li. An interior trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on optimization*, 6(2):418–445, 1996.

Cyril Crassin, Fabrice Neyret, Miguel Sainz, Simon Green, and Elmar Eisemann. Interactive indirect illumination using voxel cone tracing. In *Computer Graphics Forum*, volume 30, pages 1921–1930. Wiley Online Library, 2011.

Holger Dammertz, Johannes Hanika, Alexander Keller, and Hendrik Lensch. A hierarchical automatic stopping condition for Monte Carlo global illumination. In *Eurographics WSCG 2010: Full Paper Proceedings*, pages 159–164. Václav Skala-UNION Agency, 2010.

Eugene d'Eon and Geoffrey Irving. A quantized-diffusion model for rendering translucent materials. *ACM transactions on graphics (TOG)*, 30(4):1–14, 2011.

Eugene d'Eon, David Luebke, and Eric Enderton. Efficient rendering of human skin. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pages 147–157, Aire-la-Ville, Switzerland, 2007. Eurographics Association.

Craig Donner and Henrik Wann Jensen. Light diffusion in multi-layered translucent materials. *ACM Transactions on Graphics (ToG)*, 24(3):1032–1039, 2005.

Michal Drobot. Software-Based Variable Rate Shading in Call of Duty: Modern Warfare. In *SIGGRAPH Course: Advances in Real-Time Rendering in Games: Part 1*, Washington, D.C., 2020. ACM.

Frédo Durand, Nicolas Holzschuch, Cyril Soler, Eric Chan, and François X Sillion. A frequency analysis of light transport. *ACM Transactions on Graphics (TOG)*, 24(3):1115–1126, 2005.

Eugene dEon and David Luebke. Advanced techniques for realistic real-time skin rendering. *GPU Gems*, 3(3):293–347, 2007.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

WG Egan, To Hilgeman, and J Reichman. Determination of absorption and scattering coefficients for nonhomogeneous media. 2: Experiment. *Applied Optics*, 12 (8):1816–1823, 1973.

Unreal Engine. Dynamic Resolution, Dec 2020. URL https://docs.unrealengine.com/en-US/RenderingAndGraphics/DynamicResolution/index.html. 4.26.

Shaohua Fan, Stephen Chenney, Bo Hu, Kam-Wah Tsui, and Yu-chi Lai. Optimizing control variate estimators for rendering. *Computer Graphics Forum*, 25(3):351–357, 2006.

Tony Finch. Incremental calculation of weighted mean and variance. *University of Cambridge*, 4(11-5):41–42, 2009.

Sarah F Frisken, Ronald N Perry, Alyn P Rockwood, and Thouis R Jones. Adaptively sampled distance fields: A general representation of shape for computer graphics. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 249–254, 2000.

Henry Fuchs, John Poulton, John Eyles, Trey Greer, Jack Goldfeather, David Ellsworth, Steve Molnar, Greg Turk, Brice Tebbs, and Laura Israel. Pixel-planes 5: A heterogeneous multiprocessor graphics system using processor-enhanced memories. *ACM Siggraph Computer Graphics*, 23(3):79–88, 1989.

Evgenii Golubev. Efficient screen-space subsurface scattering using Burleys normalized diffusion in real-time, 2018. URL http://advances.realtimerendering.com/s2018/Efficient%20screen%20space%20subsurface%20scattering%20Siggraph%202018.pdf.

Evgenii Golubev. Sampling Burley's Normalized Diffusion Profiles, 2019. URL https://zero-radiance.github.io/post/sampling-diffusion/.

Pascal Grittmann, Iliyan Georgiev, Philipp Slusallek, and Jaroslav Křivánek. Variance-Aware Multiple Importance Sampling. *ACM Trans. Graph. (SIGGRAPH Asia 2019)*, 38(6), 2019. doi: 10.1145/3355089.3356515.

CC Grosjean. Multiple Isotropic Scattering in Convex Homogeneous Media Bounded by Vacuum. Part II. Some Practical Applications. Technical report, Univ. of Ghent, 1959.

Till Guldimann, Peter Zangari, Jacques Longerstaey, John Matero, and Scott Howard. Riskmetrics technical document. Technical report, Morgan Guaranty Trust Company, 1995.

Ralf Habel, Per H Christensen, and Wojciech Jarosz. Photon beam diffusion: a hybrid Monte Carlo method for subsurface scattering. In *Proceedings of the Eurographics Symposium on Rendering*, pages 27–37, Aire-la-Ville, Switzerland, 2013. Eurographics Association.

Eric Haines and Tomas Akenine-Möller. *Ray Tracing Gems: High-Quality and Real-Time Rendering with DXR and Other APIs*. Springer, 2019.

Jon Hasselgren, Jacob Munkberg, Marco Salvi, Anjul Patney, and Aaron Lefohn. Neural temporal adaptive sampling and denoising. In *Computer Graphics Forum*, volume 39, pages 147–155. Wiley Online Library, 2020.

Bingsheng He, Naga K Govindaraju, Qiong Luo, and Burton Smith. Efficient gather and scatter operations on graphics processors. In *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, page 46. ACM, 2007.

Yong He, Yan Gu, and Kayvon Fatahalian. Extending the Graphics Pipeline with Adaptive, Multi-Rate Shading. *ACM Trans. Graph.*, 33(4), July 2014. ISSN 0730-0301. doi: 10.1145/2601097.2601105. URL https://doi.org/10.1145/2601097.2601105.

E. Heitz. Can't Invert the CDF? The Triangle-Cut Parameterization of the Region under the Curve. *Computer Graphics Forum*, 39(4):121–132, 2020. doi: https://doi.org/10.1111/cgf.14058. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14058.

Timothy C Hesterberg and Barry L Nelson. Control variates for probability and quantile estimation. *Management Science*, 44(9):1295–1312, 1998.

Mark D Hill and Alan Jay Smith. Evaluating associativity in CPU caches. *IEEE Transactions on Computers*, 38(12):1612–1630, 1989.

Rama Karl Hoetzlein. GVDB: raytracing sparse voxel database structures on the GPU. In *Proceedings of High Performance Graphics*, pages 109–117. Eurographics Association, 2016.

Nikolai Hofmann, Jon Hasselgren, Petrik Clarberg, and Jacob Munkberg. Interactive Path Tracing and Reconstruction of Sparse Volumes. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 4(1):1–19, 2021.

Jose A Iglesias-Guitian, Bochang Moon, Charalampos Koniaris, Eric Smolikowski, and Kenny Mitchell. Pixel History Linear Models for Real-Time Temporal Filtering. *Computer Graphics Forum*, 35(7):363–372, 2016.

Mark Jarzynski and Marc Olano. Hash Functions for GPU Rendering. *Journal of Computer Graphics Techniques (JCGT)*, 9(3):20–38, October 2020. ISSN 2331-7418. URL http://jcgt.org/published/0009/03/02/.

Henrik Wann Jensen and Juan Buhler. A rapid hierarchical rendering technique for translucent materials. In *ACM SIGGRAPH 2005 Courses*, page 12, New York, NY, 2005. ACM.

Henrik Wann Jensen, Stephen R Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 511–518, New York, NY, 2001. ACM.

Zhe Jia, Marco Maggioni, Jeffrey Smith, and Daniele Paolo Scarpazza. Dissecting the NVidia Turing T4 GPU via Microbenchmarking. *arXiv preprint arXiv:1903.07486*, 2019.

Jorge Jimenez, Veronica Sundstedt, and Diego Gutierrez. Screen-space perceptual rendering of human skin. *ACM Transactions on Applied Perception (TAP)*, 6(4): 23, 2009.

Jorge Jimenez, Kroly Zsolnai, Adrian Jarabo, Christian Freude, Thomas Auzinger, Xian-Chun Wu, Javier von der Pahlen, Michael Wimmer, and Diego Gutierrez. Separable Subsurface Scattering. *Computer Graphics Forum*, 34(6):188–197, September 2015.

Viktor Kämpe, Erik Sintorn, and Ulf Assarsson. High resolution sparse voxel DAGs. *ACM Transactions on Graphics (TOG)*, 32(4):101, 2013.

Brian Karis. High Quality Temporal Supersampling, 2014. URL http://advances.realtimerendering.com/s2014/epic/TemporalAA.pptx.

Alexander Keller, Timo Viitanen, Colin Barré-Brisebois, Christoph Schied, and Morgan McGuire. Are we done with ray tracing? In *SIGGRAPH Courses*, pages 3–1, 2019.

Mahmoud Khairy, Zhesheng Shen, Tor M Aamodt, and Timothy G Rogers. Accel-Sim: An extensible simulation framework for validated GPU modeling. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pages 473–486. IEEE, 2020.

David Kirk and James Arvo. Unbiased sampling techniques for image synthesis. *ACM SIGGRAPH Computer Graphics*, 25(4):153–156, 1991.

Ivo Kondapaneni, Petr Vévoda, Pascal Grittmann, Tomáš Skřivan, Philipp Slusallek, and Jaroslav Křivánek. Optimal multiple importance sampling. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019.

Jaroslav Křivánek and Mark Colbert. Real-time shading with filtered importance sampling. *Computer Graphics Forum*, 27(4):1147–1154, 2008.

Eric P Lafortune and Yves D Willems. Bi-directional path tracing. 1993.

Christian Lauterbach, Michael Garland, Shubhabrata Sengupta, David Luebke, and Dinesh Manocha. Fast BVH construction on GPUs. In *Computer Graphics Forum*, volume 28, pages 375–384. Wiley Online Library, 2009.

Stephen S Lavenberg, Thomas L Moeller, and Peter D Welch. Statistical results on control variables with application to queueing network simulation. *Operations Research*, 30(1):182–202, 1982.

Mark E Lee, Richard A Redner, and Samuel P Uselton. Statistically optimized sampling for distributed ray tracing. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 61–68, 1985.

Edward Liu. DLSS 2.0 - Image Reconstruction for Real-time Rendering with Deep Learning, 2020. URL `https://developer.nvidia.com/gtc/2020/video/s22698-vid`.

Ian Mallett and Cem Yuksel. Deferred adaptive compute shading. In *Proceedings of the Conference on High-Performance Graphics*, pages 1–4, 2018.

Marco Manzi, Markus Kettunen, Frédo Durand, Matthias Zwicker, and Jaakko Lehtinen. Temporal gradient-domain path tracing. *ACM Transactions on Graphics (TOG)*, 35(6):246, 2016.

Adam Marrs, Josef Spjut, Holger Gruen, Rahul Sathe, and Morgan McGuire. Adaptive temporal antialiasing. In *Proceedings of the Conference on High-Performance Graphics*, page 1, New York, NY, 2018. ACM.

Daniel Meister, Jakub Boksansky, Michael Guthe, and Jiri Bittner. On Ray Reordering Techniques for Faster GPU Ray Tracing. In *Symposium on Interactive 3D Graphics and Games*, pages 1–9, 2020.

Xiaoxu Meng, Ruofei Du, Matthias Zwicker, and Amitabh Varshney. Kernel foveated rendering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(1):1–20, 2018.

Tom Mertens, Jan Kautz, Philippe Bekaert, Frank Van Reeth, and H-P Seidel. Efficient rendering of local subsurface scattering. In *11th Pacific Conference on-Computer Graphics and Applications, 2003. Proceedings.*, pages 51–58. IEEE, 2003.

Don P. Mitchell. Generating antialiased images at low sampling densities. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, page 6572, New York, NY, USA, 1987. Association for Computing Machinery. ISBN 0897912276. doi: 10.1145/37401.37410. URL https://doi.org/10.1145/37401.37410.

Martin Mittring and Crytek GmbH. Advanced virtual texture topics. In *ACM SIGGRAPH 2008 Games*, pages 23–51. 2008.

Bochang Moon, Nathan Carr, and Sung-Eui Yoon. Adaptive Rendering based on Weighted Local Regression. *ACM Transactions on Graphics (TOG)*, 33(5):170, 2014.

Bochang Moon, Jose A Iglesias-Guitian, Sung-Eui Yoon, and Kenny Mitchell. Adaptive rendering with linear predictions. *ACM Transactions on Graphics (TOG)*, 34(4):121, 2015.

Thomas Müller, Fabrice Rousselle, Alexander Keller, and Jan Novák. Neural control variates. *ACM Trans. Graph.*, 39(6), December 2020. doi: 10.1145/3414685.3417804. URL http://doi.acm.org/10.1145/3414685.3417804.

Barry L Nelson. Control variate remedies. *Operations Research*, 38(6):974–992, 1990.

Fred Edwin Nicodemus, Joseph C Richmond, Jack J Hsia, Irving W Ginsberg, and Thomas Limperis. *Geometrical considerations and nomenclature for reflectance.* Final Report National Bureau of Standards, Washington, DC. Inst. for Basic Standards. US Department of Commerce, National Bureau of Standards, Gaithersburg, MD, 1977.

Anthony E Nocentino and Philip J Rhodes. Optimizing memory access on GPUs using morton order indexing. In *Proceedings of the 48th Annual Southeast Regional Conference*, pages 1–4, 2010.

Jan Novák, Andrew Selle, and Wojciech Jarosz. Residual ratio tracking for estimating attenuation in participating media. *ACM Trans. Graph.*, 33(6):179–1, 2014.

Cedric Nugteren, Gert-Jan Van den Braak, Henk Corporaal, and Henri Bal. A detailed GPU cache model based on reuse distance theory. In *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, pages 37–48. IEEE, 2014.

NVIDIA. NVIDIA Turing GPU Architecture Whitepaper, 2018. URL `https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf`.

Marc Olano, Bob Kuehne, and Maryann Simmons. Automatic shader level of detail. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 7–14. Eurographics Association, 2003.

Anjul Patney, Marco Salvi, Joohwan Kim, Anton Kaplanyan, Chris Wyman, Nir Benty, David Luebke, and Aaron Lefohn. Towards foveated rendering for gaze-tracked virtual reality. *ACM Transactions on Graphics (TOG)*, 35(6):179, 2016.

Eric Penner and George Borshukov. Pre-integrated skin shading. *Gpu Pro*, 2:41–55, 2011.

Ken Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985.

Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, Cambridge, MA, 2016.

Fábio Policarpo, Manuel M Oliveira, and Joao LD Comba. Real-time relief mapping on arbitrary polygonal surfaces. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 155–162, 2005.

Ravi Ramamoorthi and Pat Hanrahan. On the relationship between radiance and irradiance: determining the illumination from images of a convex lambertian object. *JOSA A*, 18(10):2448–2459, 2001.

Ravi Ramamoorthi, Dhruv Mahajan, and Peter Belhumeur. A first-order analysis of lighting, shading, and shadows. *ACM Transactions on Graphics (TOG)*, 26(1):2–es, 2007.

Zhong Ren, Kun Zhou, Stephen Lin, and Baining Guo. Gradient-based Interpolation and Sampling for Real-time Rendering of Inhomogeneous, Single-scattering Media. In *Computer Graphics Forum*, volume 27, pages 1945–1953. Wiley Online Library, 2008.

Maurice Ribble. Next-Gen Tile-Based GPUs, February 2008. URL http://amd-dev.wpengine.netdna-cdn.com/wordpress/media/2012/10/gdc2008_ribble_maurice_TileBasedGpus.pdf.

Brian D Ripley. *Stochastic simulation*, volume 316. John Wiley & Sons, New Jersey, 2009.

Martin Roberts. The Unreasonable Effectiveness of Quasirandom Sequences, 2018. URL http://extremelearning.com.au/unreasonable-effectiveness-of-quasirandom-sequences/.

Fabrice Rousselle, Wojciech Jarosz, and Jan Novák. Image-space control variates for rendering. *ACM Transactions on Graphics (TOG)*, 35(6):1–12, 2016.

Daniel Scherzer, Lei Yang, Oliver Mattausch, Diego Nehab, Pedro V Sander, Michael Wimmer, and Elmar Eisemann. Temporal coherence methods in real-time rendering. 31(8):2378–2408, 2012.

Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn, and Marco Salvi. Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination. In *Proceedings of High Performance Graphics*, page 2, New York, NY, 2017. ACM.

Christoph Schied, Christoph Peters, and Carsten Dachsbacher. Gradient estimation for real-time adaptive temporal filtering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(2):24, 2018.

Cyril Soler, Kartic Subr, Frédo Durand, Nicolas Holzschuch, and François Sillion. Fourier depth of field. *ACM Transactions on Graphics (TOG)*, 28(2):1–12, 2009.

Gilbert W Stewart. On the early history of the singular value decomposition. *SIAM review*, 35(4):551–566, 1993.

László Szécsi, Mateu Sbert, and László Szirmay-Kalos. Combined correlated and importance sampling in direct light source computation and environment mapping. *Computer Graphics Forum*, 23(3):585–593, 2004.

Surya T Tokdar and Robert E Kass. Importance sampling: a review. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(1):54–60, 2010.

Ruey S Tsay. *Analysis of financial time series*, volume 543. John Wiley & Sons, New Jersey, 2005.

Unity. Dynamic Resolution, Apr 2019. URL https://docs.unity3d.com/Manual/DynamicResolution.html. 2019.4 LTS.

Eric Veach and Leonidas J. Guibas. Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, page 419428, New York, NY, USA, 1995. Association for Computing Machinery. ISBN 0897917014. doi: 10.1145/218380.218498. URL https://doi.org/10.1145/218380.218498.

Alberto Jaspe Villanueva, Fabio Marton, and Enrico Gobbetti. SSVDAGs: symmetry-aware sparse voxel DAGs. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 7–14. ACM, 2016.

Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. 1995.

BP Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.

Graham Wihlidal. 4K Checkerboard in Battlefield 1 and Mass Effect Andromeda. In *Game Developers Conference*, volume 3, page 8, 2017.

Lance Williams. Pyramidal parametrics. In *Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, pages 1–11, 1983.

Kai Xiao, Gabor Liktor, and Karthik Vaidyanathan. Coarse pixel shading with temporal supersampling. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, page 1, New York, NY, 2018. ACM.

Tiantian Xie and Marc Olano. Real-time Subsurface Control Variates: Temporally Stable Adaptive Sampling. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 4(1):1–18, 2021.

Tiantian Xie, Marc Olano, Brian Karis, and Krzysztof Narkowicz. Real-time subsurface scattering with single pass variance-guided adaptive importance sampling. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 3(1): 1–21, 2020.

Lei Yang, Diego Nehab, Pedro V Sander, Pitchaya Sitthi-amorn, Jason Lawrence, and Hugues Hoppe. Amortized supersampling. *ACM Transactions on Graphics (TOG)*, 28(5):135, 2009.

Lei Yang, Shiqiu Liu, and Marco Salvi. A survey of temporal antialiasing techniques. In *Computer Graphics Forum*, volume 39, pages 607–621. Wiley Online Library, 2020.

Zheng Zeng, Shiqiu Liu, Jinglei Yang, Wang Lu, and Yan Ling-Qi. Temporally Reliable Motion Vectors for Real-time Ray Tracing. In *Computer Graphics Forum*, volume 40. Wiley Online Library, 2021.

Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, Bochang Moon, Ravi Ramamoorthi, Fabrice Rousselle, Pradeep Sen, Cyril Soler, and S-E Yoon. Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. In *Computer graphics forum*, volume 34, pages 667–681. Wiley Online Library, 2015.