

DENOISING AND GUIDED UPSAMPLING OF MONTE CARLO PATH
TRACED LOW RESOLUTION RENDERINGS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

KADIR CENK ALPAY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

FEBRUARY 2021

Approval of the thesis:

**DENOISING AND GUIDED UPSAMPLING OF MONTE CARLO PATH
TRACED LOW RESOLUTION RENDERINGS**

submitted by **KADIR CENK ALPAY** in partial fulfillment of the requirements for
the degree of **Master of Science in Computer Engineering Department, Middle
East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering**

Prof. Dr. Ahmet Oğuz Akyüz
Supervisor, **Computer Engineering, METU**

Examining Committee Members:

Prof. Dr. Tolga Can
Computer Engineering, METU

Prof. Dr. Ahmet Oğuz Akyüz
Computer Engineering, METU

Assoc. Prof. Dr. Erkut Erdem
Computer Engineering, Hacettepe University

Date: 15.02.2021



I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Kadir Cenk Alpay

Signature :

ABSTRACT

DENOISING AND GUIDED UPSAMPLING OF MONTE CARLO PATH TRACED LOW RESOLUTION RENDERINGS

Alpay, Kadir Cenk

M.S., Department of Computer Engineering

Supervisor: Prof. Dr. Ahmet Oğuz Akyüz

February 2021, 81 pages

Monte Carlo path tracing is used to generate renderings by estimating the rendering equation using the Monte Carlo method. An extensive amount of ray samples per pixel is needed to be cast during this rendering process to create an image with a low enough variance to be considered visually noise-free. Casting that amount of samples requires an expensive time budget. Many studies focus on rendering a noisy image at the original resolution with a decreased sample count and then applying a post-process denoising to produce a visually appealing output. This approach speeds up the rendering process and creates a denoised image of comparable quality to the visually noise-free ground truth. However, the denoising process cannot handle the noisy image's high variance accurately if the sample count is decreased harshly to complete the rendering process in a shorter time budget. In this thesis work, we try to overcome this problem by proposing a pipeline that renders the image at a reduced resolution to cast more samples than the harshly decreased sample count in the same time budget. This noisy low-resolution image is then denoised more accurately, thanks to having a lower variance. It is then upsampled with the guidance of the auxiliary scene data rendered swiftly in a separate rendering pass at the original resolution. Experimental

evaluation shows that the proposed pipeline generates denoised and guided upsampled images in promisingly good quality compared to denoising the noisy original resolution images rendered with the harshly decreased sample count.

Keywords: Monte Carlo path tracing, rendering, denoising, guided upsampling



ÖZ

MONTE CARLO YOLAK İZLENMİŞ DÜŞÜK ÇÖZÜNÜRLÜKLÜ KAPLAMALARIN GÜRÜLTÜDEN ARINDIRILMASI VE GÜDÜMLÜ YUKARI ÖRNEKLENMESİ

Alpay, Kadir Cenk

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Prof. Dr. Ahmet Oğuz Akyüz

Şubat 2021 , 81 sayfa

Monte Carlo yolak izleme tekniği, görsel kaplama denklemini Monte Carlo yöntemiyle tahminleyerek görüntüler üretmek için kullanılmaktadır. Görsel olarak gürültüsüz sayılabilecek kadar düşük varyanslı bir görüntü üretebilmek için görsel kaplama işlemi anında piksel başına çok büyük miktarda ışın örneği gönderilmesi gerekmektedir. Bu miktarda ışın örneği göndermek pahalı bir zaman bütçesi gerektirmektedir. Birçok çalışma, orijinal çözünürlükte ışın sayısı azaltılmış bir gürültülü görüntü üretip sonrasında art-işlem gürültüden arındırma işlemi uygulayarak görsel olarak cazip bir çıktı üretmek üzerine odaklanmaktadır. Bu yaklaşım, görsel kaplama işlemini hızlandırmakta ve görsel olarak gürültüsüz olan referans görüntüyle karşılaştırılabilir kalitede gürültüden arındırılmış bir görüntü oluşturmaktadır. Ancak bu gürültüden arındırma işlemi, ışın örneği sayısının görsel kaplama işlemini daha kısa bir süre zarfında tamamlayabilmek adına aşırı azaltılması sonucu oluşan gürültülü görüntünün yüksek varyansının üstesinden tutarlı bir şekilde gelememektedir. Bu tez çalışmasında, bahsi geçen bu kısa süre zarfında bu aşırı azaltılmış ışın örneği sayısından daha fazla ışın

örneđi gönderebilmek ve böylece daha düşük varyanslı pikseller elde edebilmek için görüntünün çözünürlüğünü düşüren bir ardışık düzen önerilerek bu problemin üstesinden gelinmeye çalışılmıştır. Oluşan düşük çözünürlüklü gürültülü görüntü daha düşük varyansa sahip olduđu için daha tutarlı olarak gürültüden arındırılmakta ve sonrasında ayrı bir görsel kaplama geçişiyle yüksek çözünürlükte hızlı bir şekilde üretilmiş harici sahne verileri güdüm olarak kullanılarak güdümlü yukarı örneklendirilmektedir. Deneysel değerlendirme önerilen yöntemin, aşırı azaltılmış ışıın örneđi sayısıyla orijinal çözünürlükte oluşturulmuş görüntüyü gürültüden arındırma yöntemi ile karşılaştırıldığında umut verici derecede iyi kalitede gürültüden arındırılmış ve güdümlü yukarı örneklenmiş görüntüler ürettiğini göstermektedir.

Anahtar Kelimeler: Monte Carlo yolak izleme, görsel kaplama, gürültüden arındırma, güdümlü yukarı örnekleme



To my family

ACKNOWLEDGMENTS

First of all, I would like to thank my supervisor Prof. Dr. Ahmet Oğuz Akyüz, for his helpful and friendly guidance throughout the path I traced from the beginning of this thesis work to the successful end. I believe we will achieve more accomplishments together with our studies in the future.

I also would like to thank my dearest family: my mother, father, sister, aunt, and late uncle, "Fa Amca", for their huge support and love throughout my life. They are the ones who taught me to always push forward for a successful and happy living.

I have benefited from TUBITAK 2210/A National MSc Scholarship Program during my graduate education, which is a merit scholarship program. Moreover, the numerical calculations reported in this thesis work were partially performed at TUBITAK ULAKBIM, High Performance and Grid Computing Center (TRUBA resources). Therefore, I am grateful to TUBITAK for all the support.

TABLE OF CONTENTS

ABSTRACT	v
ÖZ	vii
ACKNOWLEDGMENTS	x
TABLE OF CONTENTS	xi
LIST OF TABLES	xiv
LIST OF FIGURES	xvi
LIST OF ABBREVIATIONS	xxi
CHAPTERS	
1 INTRODUCTION	1
1.1 Motivation and Problem Definition	1
1.2 Proposed Methods and Models	2
1.3 Contributions and Novelties	3
1.4 The Outline of the Thesis	3
2 BACKGROUND AND RELATED WORK	5
2.1 Monte Carlo Path Tracing	5
2.2 During-Rendering Variance Reduction Techniques	6
2.3 Post-Rendering Denoising of Monte Carlo Images	8
2.3.1 Kernel-Predicting Convolutional Networks	10

2.4	Guided Image Upsampling	11
2.4.1	Guided Filter	12
2.4.2	Deep Guided Filter	13
3	PROPOSED METHOD	15
3.1	Rendering in LR with Higher SPP	17
3.2	Denoising Higher SPP LR Image	24
3.3	HR Scene Feature Buffers	27
3.4	Upsampling LR Image with HR Guidance	29
4	EXPERIMENTAL EVALUATION	35
4.1	Dataset	35
4.2	Guided Filter Experiment	37
4.2.1	KPCN Training Details	37
4.2.2	Input Image Specifications	38
4.2.3	Guided Filter Settings	38
4.2.4	HR Guidance Specifications	39
4.2.5	Results	43
4.3	Deep Guided Filter Experiment	52
4.3.1	Dataset Splitting Strategy	52
4.3.2	KPCN Training Details	53
4.3.3	HR Guidance Specifications	56
4.3.4	DGF Training Details	56
4.3.5	Results	57
5	DISCUSSIONS	69

5.1	Experiment Overview	69
5.1.1	Lack of Guidance	70
5.1.2	Lack of Dataset	71
5.2	Runtime Performance	72
6	CONCLUSIONS	75
6.1	Future Work	76
	REFERENCES	79



LIST OF TABLES

TABLES

Table 3.1	The notations used in this chapter and their descriptions.	15
Table 4.1	Effects of parameter settings of Guided Filter of both the diffuse and specular guided upsamplers on the performance of the proposed pipeline. r is the window radius, and ϵ is the regularization parameter, constraining coefficient a from becoming too large. The metric results show the comparison of the final image, formed by the addition of upsampled diffuse and specular results, with the ground-truth.	40
Table 4.2	Metric results of guided filter experiment. Each metric is calculated by comparing to the 8192 SPP ground-truth, LDR image is tone-mapped version of the HDR result, SSIM stands for mean SSIM value, figure number points to the figure that includes the resulting baseline, proposed pipeline, and ground-truth visuals of the corresponding scene. . . .	42
Table 4.3	Metric results of deep guided filter experiment (DGF) together with the guided filter experiment (GF). Each metric is found by comparing to the 8192 SPP groundtruth, LDR image is tone-mapped version of the HDR result, SSIM stands for mean SSIM value, figure number points to the figure that includes the resulting baseline, proposed pipeline, and ground-truth visuals of the corresponding scene. Bold text indicates higher score than the baseline used in the corresponding experiment.	60

Table 5.1 Timing results in seconds. Input images are rendered using the Tungsten renderer. Original resolution image is rendered at 1280x720 with 8 SPP and fed to the baseline as input. Low-resolution image is rendered at 640x360 with 32 SPP and fed to our pipeline as input. Ground-truth images are rendered at the original resolution with 8192 SPP, elapsing tremendous amounts of render time. KPCN time cost depends on the resolution of the image. Therefore, we provide both baseline KPCN and our pipeline’s KPCN timing results. GF implementation we have used is an unoptimized one. Better timing results can be obtained using an optimized GF code. We assume that the original resolution scene feature buffers (normal map, depth map, albedo map) are generated with a separate rendering pass in a short time budget. 74

LIST OF FIGURES

FIGURES

Figure 2.1	Flow of the baseline approach. Denoising of a reduced SPP Monte Carlo image is done at the original resolution, resulting in a denoised image at a quality comparable with the ground-truth.	9
Figure 2.2	The network architecture of the KPCN denoiser. Adapted from the KPCN paper.	10
Figure 2.3	The architecture of the Deep Guided Filter (DGF). Adapted from the DGF paper.	13
Figure 3.1	The flow of the proposed pipeline.	16
Figure 3.2	The example scene rendered at R_{HR} , 1280x720, with SPP_{GT} , 8192 SPP.	18
Figure 3.3	The example scene rendered at R_{HR} , 1280x720, with SPP_{BL} , 128 SPP.	18
Figure 3.4	The result of the baseline approach after denoising the example scene rendered at R_{HR} , 1280x720, with SPP_{BL} , 128 SPP.	19
Figure 3.5	The example scene rendered at R_{HR} , 1280x720, with a harshly decreased SPP_{BL} , 8 SPP.	20
Figure 3.6	The result of the baseline approach after denoising the example scene rendered at R_{HR} , 1280x720, with a harshly decreased SPP_{BL} , 8 SPP.	20
Figure 3.7	The example scene rendered at R_{LR} , 640x360, with SPP_{OUR} , 32 SPP.	22

Figure 3.8	The result of applying denoising on the example scene rendered at R_{LR} , 640x360, with SPP_{OUR} , 32 SPP.	22
Figure 3.9	Noisy diffuse component of the example scene rendered at R_{LR} , 640x360, with SPP_{OUR} , 32 SPP.	23
Figure 3.10	Noisy specular component of the example scene rendered at R_{LR} , 640x360, with SPP_{OUR} , 32 SPP.	23
Figure 3.11	Visuals of the scene feature buffers of the example scene rendered at R_{LR} , 640x360, with SPP_{OUR} , 32 SPP.	24
Figure 3.12	Noisy irradiance map of the example scene rendered at R_{LR} , 640x360, with SPP_{OUR} , 32 SPP.	26
Figure 3.13	Log of noisy specular component of the example scene rendered at R_{LR} , 640x360, with SPP_{OUR} , 32 SPP.	26
Figure 3.14	Denoised diffuse component of the example scene rendered at R_{LR} , 640x360, with SPP_{OUR} , 32 SPP.	28
Figure 3.15	Denoised specular component of the example scene rendered at R_{LR} , 640x360, with SPP_{OUR} , 32 SPP.	28
Figure 3.16	Visuals of the preprocessed HR scene feature buffers of the example scene rendered at R_{HR} , 1280x720, with 1024 SPP.	30
Figure 3.17	Denoised and guided upsampled diffuse component of the example scene rendered at R_{LR} , 640x360, with SPP_{OUR} , 32 SPP, upsampled to R_{HR} , 1280x720.	32
Figure 3.18	Denoised and guided upsampled specular component of the example scene rendered at R_{LR} , 640x360, with SPP_{OUR} , 32 SPP, upsampled to R_{HR} , 1280x720.	32
Figure 3.19	Visual comparison of the outputs of the example scene, produced by using the baseline approach and the proposed pipeline, with the ground-truth.	33

Figure 4.1	Tungsten scenes on which the training dataset is based.	36
Figure 4.2	Tungsten test scenes used for the evaluation of the experiments. .	36
Figure 4.3	Guidance images of the test scenes, created by multiplying each of the preprocessed scene feature buffers, which are 3-channel normal map, 3-channel albedo map, and 3-channel depth map. 3-channel depth map is generated by cloning its single channel into three channels. Pre-processing is done to pull their values into $[0, 1]$ range.	41
Figure 4.4	Results of the livingroom1 scene for the guided filter experiment.	44
Figure 4.5	Results of the livingroom2 scene for the guided filter experiment.	45
Figure 4.6	Results of the livingroom3 scene for the guided filter experiment.	46
Figure 4.7	Results of the bathroom2 scene for the guided filter experiment. .	47
Figure 4.8	HDR-VDP results of the livingroom1 scene in comparison to the ground-truth; higher color values indicating higher probability of detection of visual differences, and Q score indicating the predicted image quality over 10 as mean-opinion-score.	48
Figure 4.9	HDR-VDP results of the livingroom2 scene in comparison to the ground-truth; higher color values indicating higher probability of detection of visual differences, and Q score indicating the predicted image quality over 10 as mean-opinion-score.	49
Figure 4.10	HDR-VDP results of the livingroom3 scene in comparison to the ground-truth; higher color values indicating higher probability of detection of visual differences, and Q score indicating the predicted image quality over 10 as mean-opinion-score.	50
Figure 4.11	HDR-VDP results of the bathroom2 scene in comparison to the ground-truth; higher color values indicating higher probability of detection of visual differences, and Q score indicating the predicted image quality over 10 as mean-opinion-score.	51

Figure 4.12	Training and validation error charts of the baseline KPCN on L1 loss, trained using halves of 128 and 256 SPP images of the dataset. The error calculation is done by adding the denoised output of diffuse and specular networks of baseline KPCN to get the final image and then compare it to the ground-truth. The blue line shows training validation error, and the red line shows test error. Training validation error being larger due to complex scenes being used to calculate the error. We use the 71K iteration checkpoint to produce the baseline approach results, as it also results in a better SSIM score (not shown).	54
Figure 4.13	Training and validation error charts of our KPCN on L1 loss, trained using halves of 512 and 1024 SPP images of the dataset. The error calculation is done by adding the denoised output of diffuse and specular networks of baseline KPCN to get the final image and then compare it to the ground-truth. The blue line shows training validation error, and the red line shows test error. Training validation error being larger due to complex scenes being used to calculate the error. We use the 72K iteration checkpoint to produce the baseline approach results, as it also results in a better SSIM score (not shown).	55
Figure 4.14	Training error charts of both diffuse and specular DGF upsamplers. Both are trained on L1 loss using the diffuse and specular components of the remaining halves of 128, 256, 512, and 1024 SPP images of the dataset.	58
Figure 4.15	Results of the livingroom1 scene for the deep guided filter experiment.	61
Figure 4.16	Results of the livingroom2 scene for the deep guided filter experiment.	62
Figure 4.17	Results of the livingroom3 scene for the deep guided filter experiment.	63
Figure 4.18	Results of the bathroom2 scene for the deep guided filter experiment.	64

Figure 4.19 HDR-VDP results of the livingroom1 scene in comparison to the ground-truth; higher color values indicating higher probability of detection of visual differences, and Q score indicating the predicted image quality over 10 as mean-opinion-score. 65

Figure 4.20 HDR-VDP results of the livingroom2 scene in comparison to the ground-truth; higher color values indicating higher probability of detection of visual differences, and Q score indicating the predicted image quality over 10 as mean-opinion-score. 66

Figure 4.21 HDR-VDP results of the livingroom3 scene in comparison to the ground-truth; higher color values indicating higher probability of detection of visual differences, and Q score indicating the predicted image quality over 10 as mean-opinion-score. 67

Figure 4.22 HDR-VDP results of the bathroom2 scene in comparison to the ground-truth; higher color values indicating higher probability of detection of visual differences, and Q score indicating the predicted image quality over 10 as mean-opinion-score. 68

LIST OF ABBREVIATIONS

ABBREVIATIONS

2D	2 Dimensional
3D	3 Dimensional
CNN	Convolutional Neural Network
DGF	Deep Guided Filter
GAN	Generative Adversarial Network
GF	Guided Filter
HDR	High Dynamic Range
HR	High Resolution
JBU	Joint Bilateral Upsampling
K	Thousand
KPCN	Kernel-Predicting Convolutional Networks
LR	Low Resolution
MC	Monte Carlo
MIS	Multiple Importance Sampling
PDF	Probability Density Function
RGB	Red Green Blue
SPP	Samples Per Pixel



CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

Monte Carlo path tracing enables one to render a 3D scene into a 2D image with physically accurate and unbiased light transport calculations, in which the global illumination detail of the scene is included naturally, by estimating the integral of the rendering equation using the Monte Carlo method [1]. However, this approach requires an extensive amount of ray samples to be cast into the scene to render a low variance (i.e., visually noise-free) image. Casting a ray sample into the scene requires time-consuming computations to be done during the ray's recursive travel inside the 3D world of the scene. Due to this nature, the Monte Carlo path tracing process requires an excessively high amount of ray samples per pixel (SPP), together with a massive time budget to successfully render a visually compelling low-variance scene.

There have been various ideas proposed with the aim of decreasing the time cost of the Monte Carlo path tracing process without sacrificing the quality of the output image. Among those ideas, which are detailed in Chapter 2, one idea is to render the scene with a decreased amount of ray samples to generate a noisy image and then apply a post-process denoising operation to this image in order to produce a visually compelling denoised image as if the rendering is done with a high sample count. As the advantage of this approach, the rendering time is decreased vastly, which enables one to render and denoise more than one scene during the time budget of a high SPP rendering run. Moreover, this approach considers the renderer as a black box. Therefore, any Monte Carlo renderer can adopt this approach with ease.

However, this approach fails to produce good quality denoised images if the num-

ber of ray samples is decreased harshly to shorten the time cost of the Monte Carlo rendering process even more. Lowering the number of ray samples below a limit causes the rendered image to have a significantly higher variance. The Monte Carlo denoiser’s capacity cannot handle that amount of variance. Therefore, the accuracy of the denoising process decreases, resulting in a poor quality denoised output.

1.2 Proposed Methods and Models

In this thesis work, we try to overcome the problem of the harshly decreased SPP causing the denoiser to fail due to the rendered image having excessively high variance by proposing a pipeline composed of: (i) rendering at a lower resolution, (ii) denoising, and (iii) guided upsampling steps. In the first step of the pipeline, the noisy image is produced at a reduced resolution with a higher sample count than the harshly decreased SPP in the same time budget. This approach results in a lower variance image at the reduced resolution, which can be denoised accurately by the Monte Carlo denoiser. The denoised result is then upsampled using the scene feature buffers, normal map, depth map, and albedo map as guidance throughout the upsampling process. Those feature buffers are produced by a separate rendering pass at the original resolution in a negligibly short time budget, as they do not require the in-scene recursive tracing of the ray samples. Moreover, they are often noise-free.

The design of the proposed pipeline enables one to choose the desired algorithms to be used as the Monte Carlo denoiser and the guided upsampler. This design allows the proposed pipeline to stay up-to-date by plugging in the state-of-the-art algorithms of both Monte Carlo denoising and guided upsampling operations into the corresponding steps in the pipeline, granting flexibility and ease of use of the proposed approach.

To evaluate the performance of the proposed pipeline, we have conducted two experiments using the Kernel-Predicting Convolutional Networks (KPCN) of Bako et al. [2] as the Monte Carlo denoiser in both experiments. We have used Guided Filter [3] in the first experiment and Deep Guided Filter [4] in the second experiment as the guided upsampling algorithms.

1.3 Contributions and Novelties

Our contributions are as follows:

- We propose the idea of rendering Monte Carlo images at a reduced resolution to be able to cast more ray samples to get more accurate (i.e., lower variance) pixel information in the same time budget of casting a harshly decreased amount of ray samples at the original resolution, enabling the images' accurate denoising, which is not possible with the harshly decreased sample count due to the capacity of the denoiser.
- In the guided upsampling step, the auxiliary feature buffers rendered swiftly in the original resolution with higher SPP in a negligible time cost are leveraged as guidance.
- With the proposed pipeline, the time cost of rendering a visually appealing image is reduced largely thanks to the reduced amount of ray samples being cast during the rendering process.

1.4 The Outline of the Thesis

The remaining chapters of this thesis are organized as follows: In Chapter 2, we examine the related studies made with the aim of denoising Monte Carlo renderings. We also provide the reader with the required preliminary information for a better understanding of the work done in this thesis. In Chapter 3, we examine the proposed pipeline by providing detailed explanations for each step and studying an example scene throughout the chapter to visualize the inputs, intermediates, and outputs of those steps. In Chapter 4, the dataset used in the study is detailed, and the experimental evaluation results are provided under two experiments with comprehensive visual and numeric comparisons. In Chapter 5, a thorough discussion is done to uncover the findings of our study. In Chapter 6, a brief conclusion is provided together with the future work ideas, finalizing this thesis work.



CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, the preliminary knowledge required for a clear understanding of the proposed pipeline is introduced. Furthermore, studies related to the steps used in the proposed pipeline are provided.

2.1 Monte Carlo Path Tracing

Monte Carlo path tracing is the process of estimating the rendering integral to produce an image with physically accurate and unbiased illumination data [1].

$$L_o(x, \omega_o) = L_E(x, \omega_o) + \int_{\Omega} f_r(x, \omega_o, \omega_i) L_i(x, \omega_i) \cos\theta_i d\omega_i \quad (2.1)$$

The rendering equation is shown in Equation 2.1. In this equation:

- $L_o(x, \omega_o)$ is the outgoing radiance from point x , the hitpoint of the ray.
- $L_E(x, \omega_o)$ is the radiance emitted from x .
- $L_i(x, \omega_i)$ is the incoming radiance to x from a direction ω_i on the hemisphere centered at x .
- $\cos\theta_i$ is the cosine of the angle between ω_i and the surface normal of x .
- $f_r(x, \omega_o, \omega_i)$ is the Bidirectional Reflectance Distribution Function.

In Monte Carlo path tracing, ray samples are sent from the camera into the scene and traced recursively by sampling new rays at the in-scene hitpoints to reach light

sources or light-emitting surfaces and finalize camera-to-light paths. The illumination data of the pixel is calculated over these paths generated by casting the ray samples.

However, Monte Carlo path tracing requires a vast amount of ray samples per pixel to converge the rendering integral to a low variance output. Therefore, if the rendering integral is sampled randomly by casting a low number of ray samples to reduce the rendering time, it converges to a high variance output. This is because this low number of ray samples does not include all the lighting data of the scene accurately. As they are randomly sampled and recursively traced inside the scene, paths of some of those ray samples converge before reaching the light sources. This high variance is seen as visible noise in the produced output. To generate a visually noise-free image with a low amount of variance, methods that leverage those low numbers of samples in more beneficial ways are proposed. Thanks to those methods, the rendering process's time cost is significantly reduced by lowering the sample count without sacrificing the output quality.

We discuss the Monte Carlo variance-reduction methods in two groups. The studies in the first group focus on generating better ray paths to reduce variance during the rendering. The studies in the second group employ a post-process denoising operation to be applied on the high variance output after the rendering is completed. There is a vast amount of related work literature on this area. Therefore, we focus on and detail some of those related works and techniques in Section 2.2 for the first group and in Section 2.3 for the second group, and then redirect the reader to a comprehensive survey to see the other works.

2.2 During-Rendering Variance Reduction Techniques

The variance reduction techniques described in this section are employed while the rendering is in progress to generate practical paths. As the study done in this thesis is more related to the post-process denoising of the Monte Carlo images, in this section, we mention the during-rendering variance reduction techniques very briefly without providing much detail. We point the reader to the papers of these techniques for more detailed information.

When estimating the rendering equation with the Monte Carlo method, a Probability Density Function (PDF) is included in the estimator to importance sample the equation. Multiple Importance Sampling (MIS) [5] is a technique that combines different PDFs optimally to generate better paths for ray samples.

The process of tracing the ray samples from the camera into the scene with the aim of reaching a light source is called Forward Path Tracing. The reverse of this process, casting ray samples from the light source into the scene to reach the camera, is called Light Tracing. Bidirectional Path Tracing [6] leverages those two approaches and generates those two ray paths simultaneously. One path is traced starting from the camera, and the other path is traced starting from the light source. Both paths are connected using the MIS technique to finalize a good path between the camera and the light source.

Photon Mapping [7] technique employs a two-pass rendering approach. In the first pass, photons emitted from the light sources are mapped into the scene. In the second pass, rays are traced with the help of the photon information. Metropolis Light Transport [8] technique generates a set of paths by randomly mutating the path's vertices, then chooses the ideal path among them to trace using a probabilistic approach. Keller [9] introduced the technique that distributes Virtual Point Lights to the scene to capture the global illumination by employing GPU-acceleration.

Currently, techniques that first try to capture an understanding of the scene and then use this global scene information to create better ray paths emerged. These techniques are called Path Guiding techniques. The studies of Müller et al. [10] and Bako et al. [11] focus on the path guiding technique. The former learns the Spatio-Directional (SD) radiance field of the scene beforehand. The latter proposes a deep network trained on diverse scenes to remove the need for per-scene training of path guiding techniques during the rendering. Their trained network can swiftly guide ray paths on unseen scenes by importance sampling the first-bounce ray paths.

In the next section, we discuss the second group of the related studies, the Monte Carlo denoisers, which apply post-rendering denoising operation to the noisy output image to produce a denoised result. In that section, we also redirect the reader to a comprehensive survey for more information on both groups of the related works.

2.3 Post-Rendering Denoising of Monte Carlo Images

The denoising algorithms in the second group, which focuses on applying a post-process denoising operation on the noisy output of the renderer, has the advantage of having the auxiliary scene feature buffers, such as the normal map, depth map, and albedo map, already generated by the renderer, as the rendering process is completed before their execution start. Those denoising algorithms use these feature buffers as guidance during the denoising of the Monte Carlo image. Another advantage of this group is that the renderer can be seen as a black-box. As the algorithms work on the output image, they do not require the editing of the renderer code, granting generalization and ease of use with various Monte Carlo renderers. Furthermore, a Monte Carlo renderer’s noise characteristic differs from the spatially-invariant noise characteristic of natural imagery. This is why denoising methods exclusive to Monte Carlo images are studied, and the natural image denoising algorithms are not employed.

Some of the studies that are categorized in the second group are as follows. Rouselle et al. [12] obtain the denoising filters that are to be applied to denoise the Monte Carlo image using a combination of color and feature buffers. Bitterli et al. [13] propose a first-order model to efficiently leverage the information of the correlation of the noisy Monte Carlo image and the scene feature buffers. Kalantari et al. [14] employ a multilayer perceptron network for learning the kernel weights of each of the feature buffers to be used to denoise the image.

The survey of Zwicker et al. [15] details the studies made with the aim of denoising Monte Carlo renderings. The reader is encouraged to see this survey for a more comprehensive list of related works.

Currently, the focus for Monte Carlo denoising is on leveraging deep networks. Bako et al. [2] introduces the first deep learning-based algorithm, Kernel-Predicting Convolutional Networks (KPCN), to denoise Monte Carlo images by training the network on production-level renderings. Xu et al. [16] propose a Generative Adversarial Network (GAN) based Monte Carlo denoiser, which is an improvement over KPCN, and it is the current state-of-the-art of Monte Carlo denoisers. This work introduces a feature modulation architecture to leverage the scene feature buffers better

than KPCN. Moreover, its GAN-based architecture enables the learning of perceptual quality, rather than direct pixel-wise loss calculation of KPCN, granting higher perceptual quality denoising of the noisy renderings.

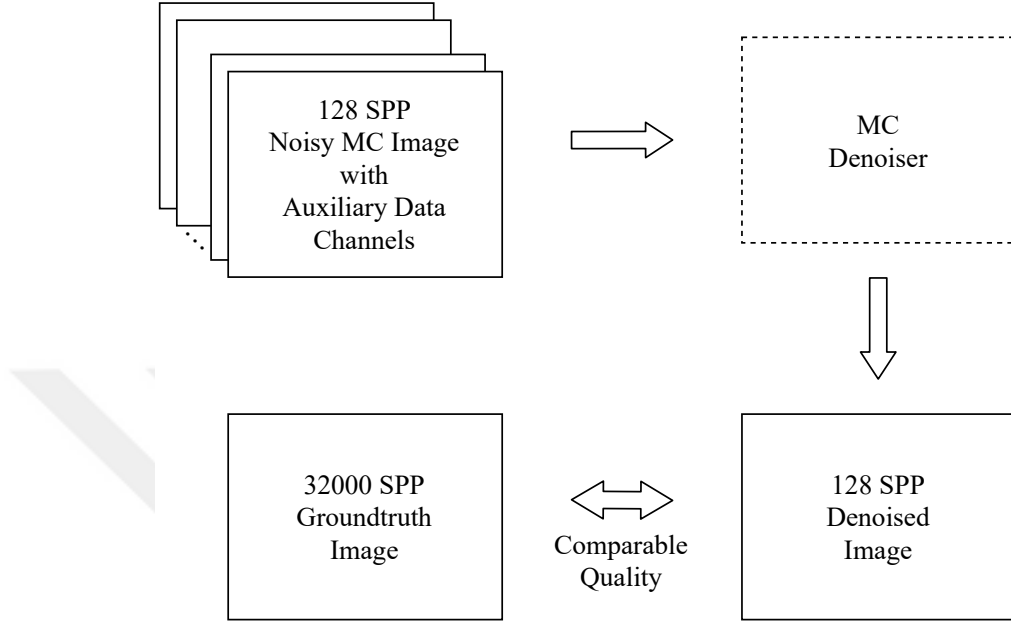


Figure 2.1: Flow of the baseline approach. Denoising of a reduced SPP Monte Carlo image is done at the original resolution, resulting in a denoised image at a quality comparable with the ground-truth.

In this thesis work, we use the KPCN denoiser as the baseline approach that denoises Monte Carlo images at the original resolution. The overall flow of the baseline approach for Monte Carlo denoising at the original resolution is shown in Figure 2.1 with example amounts of ray samples. A 128 SPP image is fed into the Monte Carlo denoiser together with the auxiliary scene data. The denoiser applies its algorithm to denoise the input image to make it look as if the image is rendered at a ground-truth SPP.

We build our proposed idea on the baseline approach’s failure case, which is the inaccurate denoising of the Monte Carlo renderings that are rendered with harshly reduced sample counts. However, our approach can also be applied to the GAN-based Monte Carlo denoiser by focusing on its failure case sample count. We discuss the

architecture of the KPCN denoiser in Section 2.3.1 briefly. Its failure case is discussed in Chapter 3, together with our proposed approach that overcomes this failure.

2.3.1 Kernel-Predicting Convolutional Networks

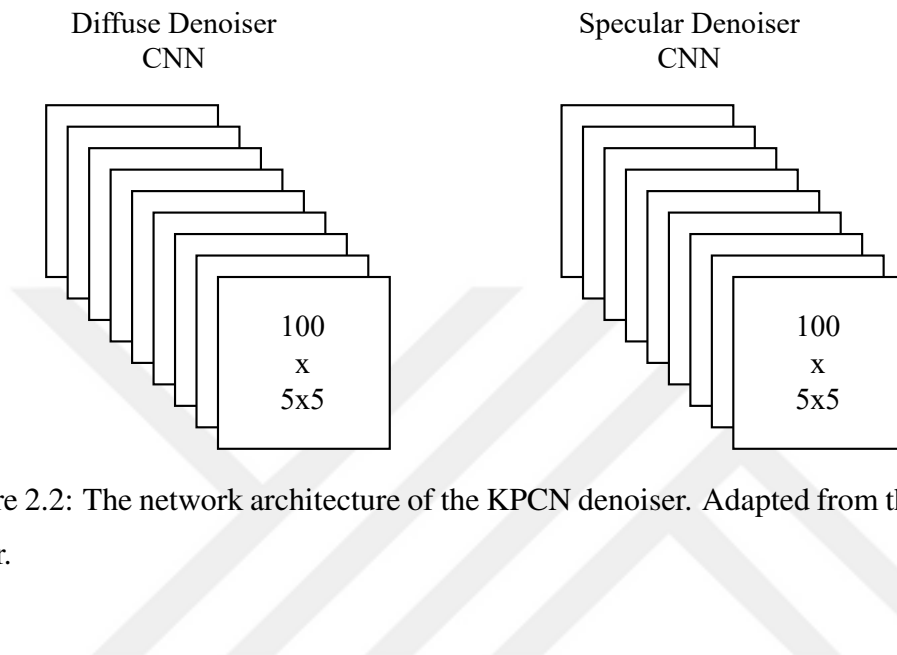


Figure 2.2: The network architecture of the KPCN denoiser. Adapted from the KPCN paper.

The deep network that the KPCN denoiser employs is composed of 9 convolutional layers. Each layer has 100 filters of size 5x5. ReLU activation function is used in between the layers. At the output, the Softmax function is applied to finalize the weights of the filtering kernel of size 21x21, to be applied on the neighborhood of the noisy pixel in consideration to denoise it.

The KPCN architecture is composed of two copies of this fully-convolutional network structure, shown in Figure 2.2, for the denoising of both the diffuse and specular components of the input Monte Carlo image. This image includes those components in its additional data channels. The noisy Monte Carlo image can also be acquired by the pixel-wise addition of those diffuse and specular components.

For each of the networks, different pre-processing and post-processing operations are applied on the components, detailed in Chapter 3. They are first trained separately and then fine-tuned jointly. The outputs of both networks are added together pixel-wise to get the denoised Monte Carlo output.

2.4 Guided Image Upsampling

Guided image upsampling algorithms use an original resolution guidance image to upsample the low-resolution input image. This idea is beneficial if a time-consuming operation is to be applied to the input image, enabling the application of this operation on the downsampled version of the image. This low-resolution solution is then upsampled using the original image or any additional image as guidance. Joint Bilateral Upsampling (JBU) [17] and Bilateral Guided Upsampling [18] are the studies whose motivations stem from this idea.

Moreover, Guided Filter (GF) [3] and Deep Guided Filter (DGF) [4] both have their upsampling applications. We choose to use them in the experimental evaluation of this thesis work in place of the proposed pipeline’s guided upsampling step. We detail GF and DGF briefly in Section 2.4.1 and Section 2.4.2, respectively.

Furthermore, deep learning-based super-resolution algorithms are proposed with the aim of generating high-resolution versions of the input images. The recent survey of Wang et al. [19] can be examined for the details of the current studies on the super-resolution of natural images.

Monte Carlo rendering is suitable for employing the guided image upsampling process as it is a highly time-consuming operation. Moreover, the advantage of the upsampling of the Monte Carlo renderings over the upsampling of the natural images is that various guidance images can be rendered in high-resolution from the same scene configuration using a separate rendering pass. Those guidance images include the normal map, albedo map, and the depth map of the scene.

Bauszat et al. [20] employs a decomposition of the Monte Carlo image into direct and indirect lighting maps and employs the GF to denoise the indirect lighting result in the original resolution using the normal map and the depth map of the scene as guidance. Our study uses the GF as an upsampler for upsampling the low-resolution denoised Monte Carlo image by using the albedo map together with the normal map and the depth map as guidance. Boughida et al. [21] employs the same direct and indirect decomposition and uses JBU to upsample them with the help of auxiliary feature buffers during the rendering on a GPU-accelerated raytracer. Unlike their

method, our approach does not use direct and indirect decomposition at first but uses the diffuse and specular decomposition directly and denoises the low-resolution noisy diffuse and specular maps using a desired Monte Carlo denoiser. The denoised results are then upsampled to the original resolution with the guidance of the aforementioned guidance images.

2.4.1 Guided Filter

Guided Filter (GF) [3] assumes that the output image q is the denoised version of the input image p . It applies a local transformation function on the guidance image I using a window-based approach to generate the output q using constraints from the input p , shown as:

$$q = p - n \quad (2.2)$$

$$q_i = a_k I_i + b_k, \forall i \in \omega_k \quad (2.3)$$

where n denotes the noise, k denotes the window index, and i denotes the index of the pixel in k th window. a and b are the coefficient matrices, and they are calculated by minimizing the following energy function:

$$E(a_k, b_k) = \sum_{i \in \omega_k} ((a_k I_i + b_k - p_i)^2 + \epsilon a_k^2) \quad (2.4)$$

After calculating a and b coefficient matrices using Equation 2.4, the filtered output is generated by applying the linear transformation shown in Equation 2.3.

GF has two parameters. r is the window radius, and ϵ is the regularization parameter used to decide which pixels are edges and which are not. In the experimental evaluation phase, we use GF as the upsampler by applying the following steps:

1. Downsample the original resolution guidance image I to the low-resolution.

2. Apply GF algorithm and find a and b coefficient matrices at the low-resolution.
3. Upsample those a and b matrices to the original resolution.
4. Calculate the output image at the original resolution by applying Equation 2.3 on the original resolution guidance I using upsampled a and b matrices.

2.4.2 Deep Guided Filter

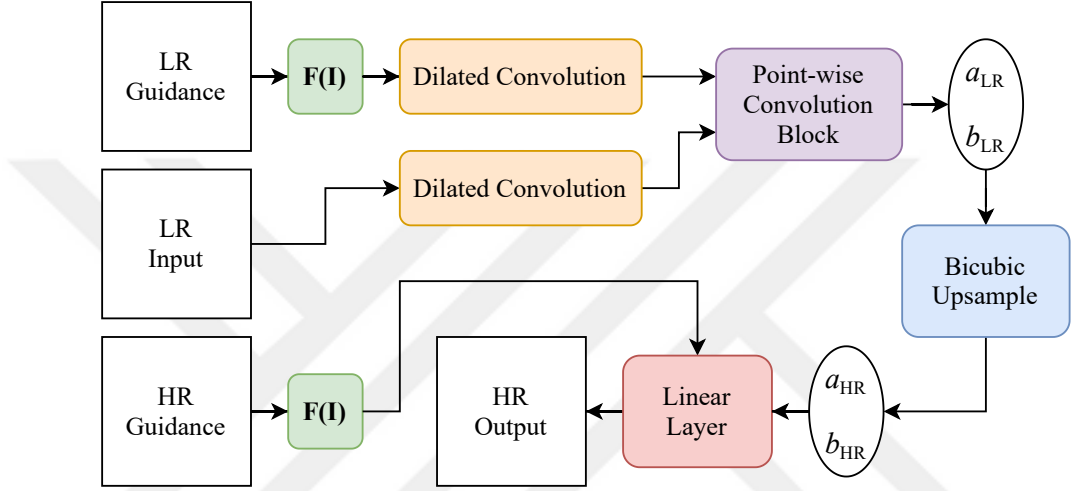


Figure 2.3: The architecture of the Deep Guided Filter (DGF). Adapted from the DGF paper.

Deep Guided Filter (DGF) [4] is a deep learning-based implementation of the upsampling application of GF, with the addition of a task-specific learnable transformation function that is to be applied to the original resolution guidance image to leverage it better.

The mean filter and the local linear model of GF are replaced by the dilated convolutions and a point-wise convolution block, respectively. With those learnable blocks, the DGF network is trained end-to-end to learn task-specific guided upsampling robustly.

The architecture of DGF is shown in Figure 2.3. In this figure, HR Guidance is the original resolution guidance image. LR Guidance is the downsampled version of the HR Guidance. LR Input is the image to be guided upsampled. $F(I)$ is the

task-specific learnable transformation function. a_{LR} and b_{LR} are the low-resolution coefficient matrices. a_{HR} and b_{HR} are the upsampled coefficient matrices. In our experiments, we employ bicubic downsampling and bicubic upsampling to generate the LR Guidance and the upsampled a and b matrices, respectively.

In the experimental evaluation phase of this thesis work, we have edited the network architecture of DGF to increase the input channel size of the guidance image from 3 to 7 so that our 7-channel guidance, detailed in Chapter 3, can be given as input to the DGF network.



CHAPTER 3

PROPOSED METHOD

In this chapter, we discuss the proposed pipeline in detail. The flow of the proposed pipeline is provided in Figure 3.1. Throughout the chapter, we explain this flow step by step while showing the application of each step on an example scene to be able to show the input images, intermediates, and final images of the proposed pipeline. The notations used in this chapter are provided in Table 3.1, together with their descriptions.

Table 3.1: The notations used in this chapter and their descriptions.

R_{HR}	Resolution of the HR Monte Carlo image. This is the required output resolution and the original resolution used in the baseline approach, which is detailed in Section 3.1. The format is Width x Height.
R_{LR}	Resolution of the LR Monte Carlo image. Same format as R_{HR} .
$f_{\downarrow}(R, x)$	Function that returns the resolution found by dividing the width and height of the given resolution R by the given amount x . For instance: $640 \times 360 = f_{\downarrow}(1280 \times 720, 2)$
SPP_{GT}	SPP used to render a Monte Carlo image in HR with sufficiently low noise that is visually not distracting.
SPP_{BL}	SPP used in baseline approach to render a Monte Carlo image in HR.
SPP_{OUR}	SPP used in our approach to render a Monte Carlo image in LR.
t_{GT}	The time budget required to render the HR image with SPP_{GT} .
t_{BL}	The time budget required to render the HR image with SPP_{BL} .
t_{OUR}	The time budget required to render the LR image with SPP_{OUR} .

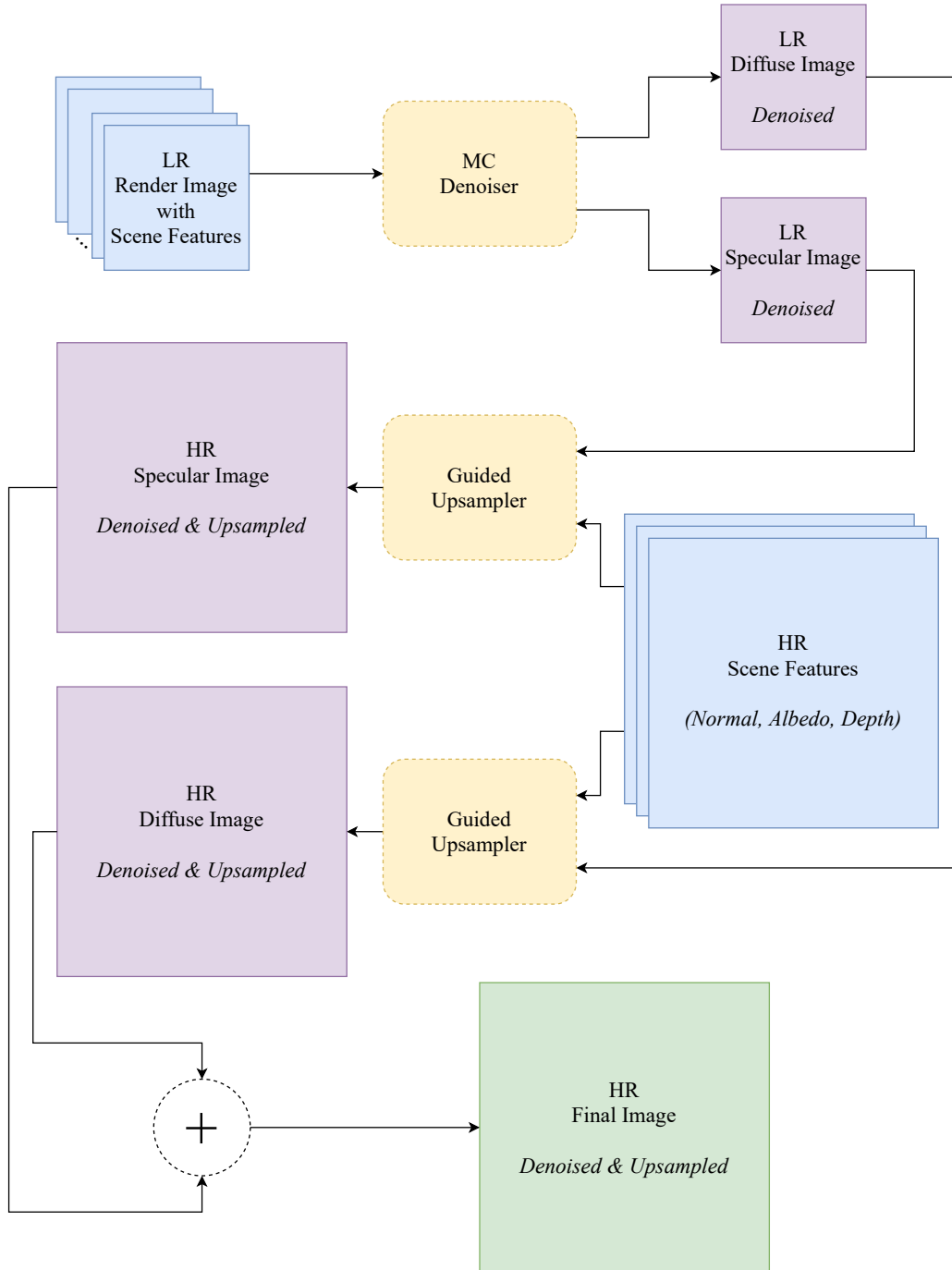


Figure 3.1: The flow of the proposed pipeline.

In Figure 3.1, yellow boxes with dotted and rounded borders specify the operation names, meaning that we do not constraint a specific algorithm to be used in them. A desired algorithm that performs the indicated operation can be plugged in place of those steps. The other boxes are shown in scale with respect to their resolutions. Blue boxes indicate the inputs to our proposed pipeline. They are generated by a desired Monte Carlo renderer, and the stack visual indicates that they have multiple image channels. Purple boxes show the intermediate results generated and then used during the execution of our pipeline. The green box at the end indicates the denoised and upsampled final image, which is the output of our pipeline. In the following sections, these steps of the proposed pipeline are detailed from the start to the end.

3.1 Rendering in LR with Higher SPP

The baseline approach that we compare our pipeline with renders scenes at a decreased sample count, SPP_{BL} . This results in a noisy image due to high variance of the information coming from ray samples, but the time cost of the rendering process becomes t_{BL} such that:

$$SPP_{BL} \ll SPP_{GT} \quad (3.1)$$

and

$$t_{BL} \ll t_{GT} \quad (3.2)$$

where t_{GT} is the time cost of rendering a low-variance and visually noise-free image by launching the Monte Carlo renderer at an excessively high sample count, SPP_{GT} , with no denoising process being required. The visuals of the example scene rendered with SPP_{GT} and SPP_{BL} are provided in Figure 3.2 and Figure 3.3, respectively.

This noisy image rendered with SPP_{BL} is then denoised by a Monte Carlo denoiser, resulting in a final image (see Figure 3.4) at a quality comparable to rendering with SPP_{GT} , with the benefit of a largely decreased time cost, as shown in Equation 3.2.



Figure 3.2: The example scene rendered at R_{HR} , 1280x720, with SPP_{GT} , 8192 SPP.



Figure 3.3: The example scene rendered at R_{HR} , 1280x720, with SPP_{BL} , 128 SPP.



Figure 3.4: The result of the baseline approach after denoising the example scene rendered at R_{HR} , 1280x720, with SPP_{BL} , 128 SPP.

This approach, however, has the limitation that if SPP_{BL} is decreased below a limit, the capacity of the denoiser cannot handle the high variance of the image correctly due to lack of information as pixels. Therefore, severe artifacts start to occur in the resulting denoised image. The visual of the example scene rendered with a harshly decreased SPP_{BL} is given in Figure 3.5, and its poorly denoised result is provided in Figure 3.6.

Therefore in our pipeline, we propose to reduce the resolution of this noisy image to be rendered to a lower resolution R_{LR} , with a rate of x on both dimensions of its original resolution R_{HR} , such that:

$$R_{LR} = f_{\downarrow}(R_{HR}, x) \quad (3.3)$$

Decreasing the resolution speeds up the rendering process by the rate of x^2 , as the number of pixels decreases by this amount. We use this extra time budget to cast more ray samples into the scene during the rendering of the LR image; however, the total amount of ray samples per pixel still being less than SPP_{GT} . We call this new



Figure 3.5: The example scene rendered at R_{HR} , 1280x720, with a harshly decreased SPP_{BL} , 8 SPP.



Figure 3.6: The result of the baseline approach after denoising the example scene rendered at R_{HR} , 1280x720, with a harshly decreased SPP_{BL} , 8 SPP.

sample count SPP_{OUR} , where:

$$(SPP_{BL} \cdot x^2) = SPP_{OUR} \ll SPP_{GT} \quad (3.4)$$

Rendering at R_{LR} with SPP_{OUR} results in a LR output image with a lower variance, due to the sample count being increased, in time budget t_{OUR} that is similar to the time budget of the baseline approach, t_{BL} , and much smaller than t_{GT} , shown as:

$$t_{BL} \approx t_{OUR} \ll t_{GT} \quad (3.5)$$

This approach allows us to apply the denoising process accurately to the LR image rendered with SPP_{OUR} , as it has a higher sample count and lower variance than an image rendered with an SPP_{BL} that is below the aforementioned limit of the baseline approach to work correctly. Figure 3.7 visualizes the example scene rendered at R_{LR} with the SPP_{OUR} found by using Equation 3.4 by setting x to 2 and by using the harshly decreased SPP_{BL} of Figure 3.5. The denoised image that is found after the denoising process is applied on this SPP_{OUR} image at R_{LR} is shown in Figure 3.8.

In Figure 3.1, the LR render image is shown to have multiple channels. This rendered image we are working with is in High Dynamic Range (HDR). It is composed of diffuse and specular components as additional channels, which can be directly output by the MC renderers and summed pixel-wise to obtain the rendered image itself. This decomposition of the rendered image into diffuse and specular components during rendering is proposed by Zimmer et al. [22]. It allows robust MC denoising of both components separately, as applying the denoising operation directly on the rendered image causes blurry results at the end because of the denoiser not being able to accurately decide between scene detail and MC noise [2]. Figure 3.9 and Figure 3.10 show the noisy diffuse and specular components of the LR example scene, respectively.

Other channels of the LR render output include the scene features, such as normal, depth, and albedo maps of the scene in LR, visualized in Figure 3.11. Those are the auxiliary scene data generated during the travel of the rays sent to calculate the rendered image itself. Therefore, they cause no additional time cost to calculate, as they do not require additional ray samples to be cast into the scene exclusively.



Figure 3.7: The example scene rendered at R_{LR} , 640x360, with SPP_{OUR} , 32 SPP.



Figure 3.8: The result of applying denoising on the example scene rendered at R_{LR} , 640x360, with SPP_{OUR} , 32 SPP.

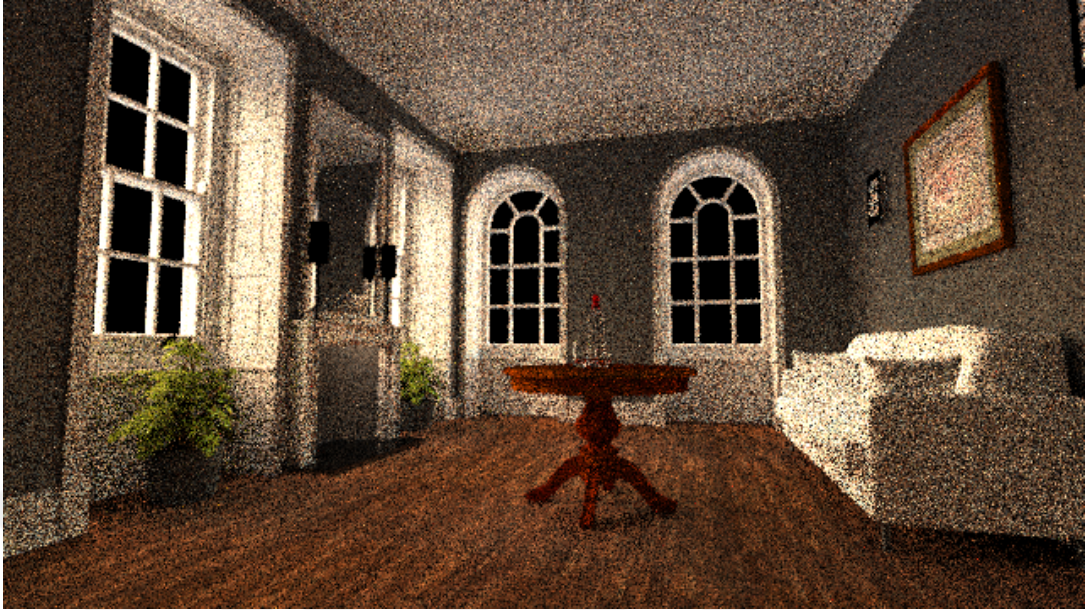


Figure 3.9: Noisy diffuse component of the example scene rendered at R_{LR} , 640x360, with SPP_{OUR} , 32 SPP.



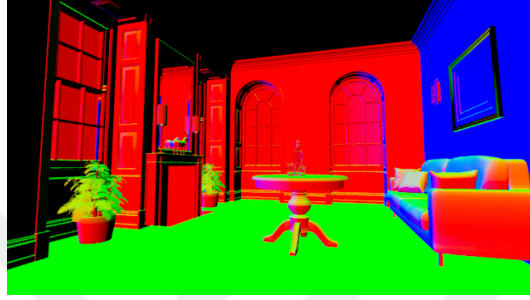
Figure 3.10: Noisy specular component of the example scene rendered at R_{LR} , 640x360, with SPP_{OUR} , 32 SPP.



(a) Albedo map



(b) Depth map



(c) Normal map

Figure 3.11: Visuals of the scene feature buffers of the example scene rendered at R_{LR} , 640×360 , with SPP_{OUR} , 32 SPP.

Moreover, current Monte Carlo denoisers make use of those auxiliary scene buffers, along with any additional data channels that might be added by the user, to guide the denoising process and achieve better results. In the next section, this Monte Carlo denoising operation is to be discussed in detail.

3.2 Denoising Higher SPP LR Image

The denoising step of the proposed pipeline is not restricted to a particular Monte Carlo denoiser. A desired Monte Carlo denoiser that can denoise the diffuse and specular components separately can be plugged in and used at this step. The same idea applies to the guided upsampling step, which is detailed in Section 3.4. This grants flexibility and ease of use for the proposed pipeline. Moreover, this renders the proposed pipeline stay up-to-date as a whole anytime in the future, as the state-of-the-art of MC denoisers and guided upsamplers can be plugged in by the user and

used at these steps.

The Monte Carlo denoiser denoises the diffuse and specular components of the LR image independently and outputs them. The reason for this MC denoiser being shown as one box in Figure 3.1 is that the denoiser we have used during the experimental evaluation phase of the study is a deep learning-based one, and it employs a joint fine-tuning step while learning to denoise the renderings, that are the composition of the diffuse and specular components, by first doing the training of two copies of the same network separately, one for each of those components.

The LR image is preprocessed before it is given as input to the Monte Carlo denoiser. Various preprocessing operations are applied to the diffuse and specular components of the image separately.

The diffuse component is the albedo multiplied version of the calculated irradiance. Albedo map includes the texture of the scene. Therefore it generally includes high-frequency data, as visualized in Figure 3.11a. Applying denoising directly on the diffuse component without preprocessing harms the robustness of the denoising process, as the denoising operation not being able to handle the high-frequency data well and blurring it [2]. As the albedo map is already among the additional scene feature buffers comparably less noisy than the irradiance map, the diffuse image is divided by this albedo map to get the noisy irradiance map of the scene. This noisy irradiance map is smoother than the diffuse component. Therefore can be denoised more robustly than the diffuse component. After denoising the irradiance map, the denoised irradiance map is multiplied back by the albedo map to get the denoised diffuse image. This preprocessing of the diffuse component is introduced by Zimmer et al. [22]. Figure 3.12 shows the noisy irradiance map of the example scene, created by dividing the diffuse component by the albedo map.

For the specular component, we employ the preprocessing method proposed by Bako et al. [2]. The data included in the specular component ranges in broad intervals from pixel to pixel due to specular lighting result being in HDR. To prevent the denoising process from getting affected negatively by those spikes, they propose to apply log transform to pull the specular data into the log domain for a smoother specular map. After denoising the specular map in the log domain, the denoised result can then be



Figure 3.12: Noisy irradiance map of the example scene rendered at R_{LR} , 640x360, with SPP_{OUR} , 32 SPP.

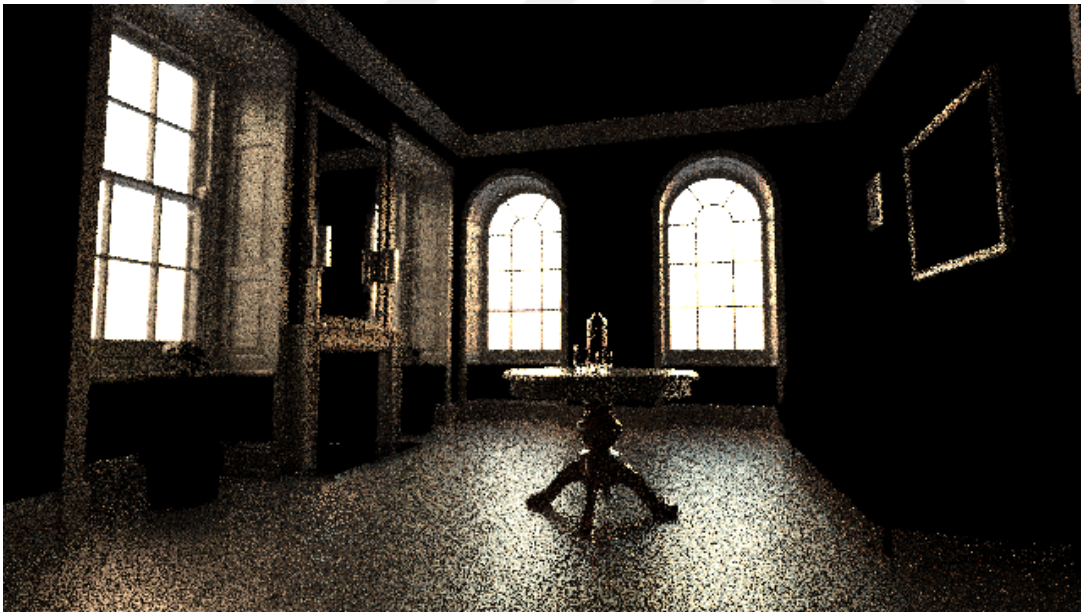


Figure 3.13: Log of noisy specular component of the example scene rendered at R_{LR} , 640x360, with SPP_{OUR} , 32 SPP.

converted back to its original domain by applying the exponential operation to all pixels. Figure 3.13 shows the log-applied noisy specular component of the example scene.

Both the denoised diffuse and denoised specular components, that are shown in Figure 3.14 and Figure 3.15, respectively, are still in the low-resolution, R_{LR} . To be able to finalize the pipeline, those images are needed to be upsampled to the original resolution, R_{HR} . The next section discusses the HR scene feature buffers of the scene that are used to guide this upsampling process.

3.3 HR Scene Feature Buffers

The main advantage of the upsampling of computer-generated imagery over the natural image upsampling is that one can generate auxiliary scene features at high-resolution with a relatively higher SPP in a separate rendering pass in a short and often negligible time budget and use them to guide the upsampling process of the low-resolution Monte Carlo image. As they are the data coming from the first hit point of the ray sent into the 3D scene, they do not require the recursive in-scene travel of the rays and can be calculated in this negligible time budget. Those HR feature buffers are the normal map, depth map, and albedo map of the scene. As they are much less noisy than the low SPP Monte Carlo image, they include the edges and texture details of the scene in solid condition, thanks to having a low variance.

Therefore, we have designed the upsampling step of the proposed pipeline to be a guided upsampling process to leverage those less noisy auxiliary scene features. To use them as guidance during the upsampling process, we first preprocess them and pull their data to the $[0, 1]$ range to increase the upsampling algorithms' robustness by applying the following operations.

The normal map includes three channels, with the pixels being in the $[-1, 1]$ range. This data shows the 3D normal vector of the first hit point of the ray samples sent for that pixel. Therefore, we first add 1 to all the pixels to set the minimum values of the normal map to 0. As the maximum value of the normal map becomes 2, we normalize this maximum value to 1 by dividing all pixels by 2.



Figure 3.14: Denoised diffuse component of the example scene rendered at R_{LR} , 640x360, with SPP_{OUR} , 32 SPP.



Figure 3.15: Denoised specular component of the example scene rendered at R_{LR} , 640x360, with SPP_{OUR} , 32 SPP.

The depth map is rendered as a single-channel image, where each pixel shows the in-scene distance of the first hit point of the ray samples to the camera. This depth map is first clipped to the range $[0, \text{maxDepth}]$, where maxDepth being the maximum value of the pixels of the depth map, to prevent negative values that could be output by the renderer. Then, the depth map is normalized linearly by dividing each pixel by the maxDepth value, resulting in a preprocessed depth map with all its pixels being in the $[0, 1]$ range.

The albedo map includes the RGB texture colors, and it is produced by the renderer in the $[0, 1]$ range. Therefore, no preprocessing operation is applied on the albedo map.

Figure 3.16 visualizes the example scene’s preprocessed scene feature buffers that are rendered with a higher SPP than the LR Monte Carlo image’s, in a separate but quick rendering pass at R_{HR} .

With all of these HR scene feature buffers are preprocessed into the range $[0, 1]$, they are now ready to be used as guidance during the guided upsampling phase of the denoised LR image. The following section details this guided upsampling process and finalizes the proposed pipeline’s output image at the end.

3.4 Upsampling LR Image with HR Guidance

As the denoising step, the guided upsampling step, too, is designed to be plugged in a desired guided upsampler algorithm, granting customizability and up-to-date performance by enabling the use of state-of-the-art algorithms at any time.

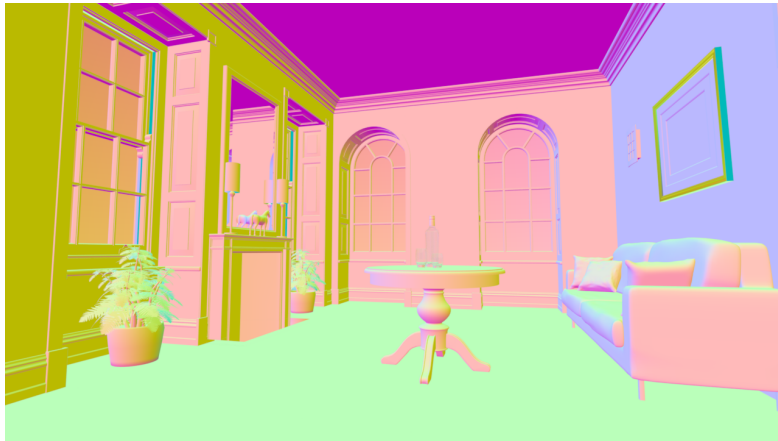
The reason for having two separate guided upsampler boxes in Figure 3.1 is that the diffuse and specular components of the rendered image require different parameter settings for the guided upsampling process as those components have different characteristics, and the guided upsampling algorithms we have used during the experimental evaluation phase cannot apply a joint process while upsampling those specular and diffuse components. Therefore, we have placed two guided upsampler boxes indicating two differently-parameterized guided upsampling flows, even though the



(a) Albedo map



(b) Depth map



(c) Normal map

Figure 3.16: Visuals of the preprocessed HR scene feature buffers of the example scene rendered at R_{HR} , 1280x720, with 1024 SPP.

algorithms used are the same.

To have a smooth input for the guided upsampling process, we preprocess the input data, which are the denoised diffuse and specular components, by applying the same preprocessing methods employed in the Monte Carlo denoising step. We factor out the albedo map from the denoised diffuse component to input the denoised irradiance map to the irradiance data upsampler. We also apply log transform to the denoised specular component to get the logged specular data as the input for the specular data upsampler.

After the guided upsampling operation is done using the HR scene features detailed in Section 3.3 as guidance, the resulting denoised and upsampled diffuse component is produced by multiplying the upsampled irradiance data by the HR albedo map that is already given among the HR scene feature buffers, and the resulting denoised and upsampled specular component is generated by applying the exponential operation on the upsampled logged specular data. Figure 3.17 shows the example scene’s denoised and upsampled diffuse component. Figure 3.18 shows the example scene’s denoised and upsampled specular component.

As the last operation of the proposed pipeline, the denoised and upsampled diffuse component and the denoised and upsampled specular component are added together pixel-wise, composing the final denoised and upsampled Monte Carlo image and concluding the proposed pipeline. Figure 3.19 shows the resulting denoised and upsampled image of the example scene produced by using the proposed pipeline, together with the result of the baseline approach and the ground-truth result. In this figure, the result of the baseline approach is rendered at R_{HR} , 1280x720, with harshly decreased SPP_{BL} , 8 SPP, and then denoised. The result of the proposed pipeline is rendered at R_{LR} , 640x360, with SPP_{OUR} , 32 SPP, and then denoised at R_{LR} and guided upsampled to R_{HR} , 1280x720. The ground-truth result is rendered at R_{HR} , 1280x720, with SPP_{GT} , 8192 SPP, with no denoising operation being applied.



Figure 3.17: Denoised and guided upsampled diffuse component of the example scene rendered at R_{LR} , 640x360, with SPP_{OUR} , 32 SPP, upsampled to R_{HR} , 1280x720.



Figure 3.18: Denoised and guided upsampled specular component of the example scene rendered at R_{LR} , 640x360, with SPP_{OUR} , 32 SPP, upsampled to R_{HR} , 1280x720.



(a) Result of the baseline approach



(b) Result of the proposed pipeline



(c) Ground-truth result

Figure 3.19: Visual comparison of the outputs of the example scene, produced by using the baseline approach and the proposed pipeline, with the ground-truth.



CHAPTER 4

EXPERIMENTAL EVALUATION

In this chapter, we first provide the details of our dataset and then evaluate the proposed pipeline in two experiments by plugging in a Monte Carlo denoiser and two different guided upsamplers. In the first experiment, we use the Guided Filter of He et al. [3] as the guided upsampler. In the second experiment, we plug in the Deep Guided Filter of Wu et al. [4], which is a deep learning-based implementation of the guided filter, as the guided upsampler. For both experiments, we use the deep learning-based MC denoiser: Kernel-Predicting Convolutional Networks (KPCN) of Bako et al. [2]. The architectures of all three algorithms are described briefly in Chapter 2. In the experiment sections of this chapter, the values set for the parameters of those algorithms are provided, and the final results are reported.

4.1 Dataset

We have used the Tungsten Renderer and its publicly available scene description files to generate Monte Carlo renderings at the desired sample counts during the experimental evaluation phase of the proposed pipeline. The renderer and its scene files are provided by Benedikt Bitterli [23].

Using 8 of those publicly available Tungsten scene files, visualized in Figure 4.1, a large Monte Carlo denoising dataset is generated as the supplemental material for the KPCN study and provided by Disney Research Studios [2]. This dataset is composed of (noisy image, ground-truth image) pairs of the renderings of 1451 scene files that are produced based on the aforementioned 8 scenes by randomly changing the lighting parameters, textures, and camera parameters. Those pairs' noisy images are

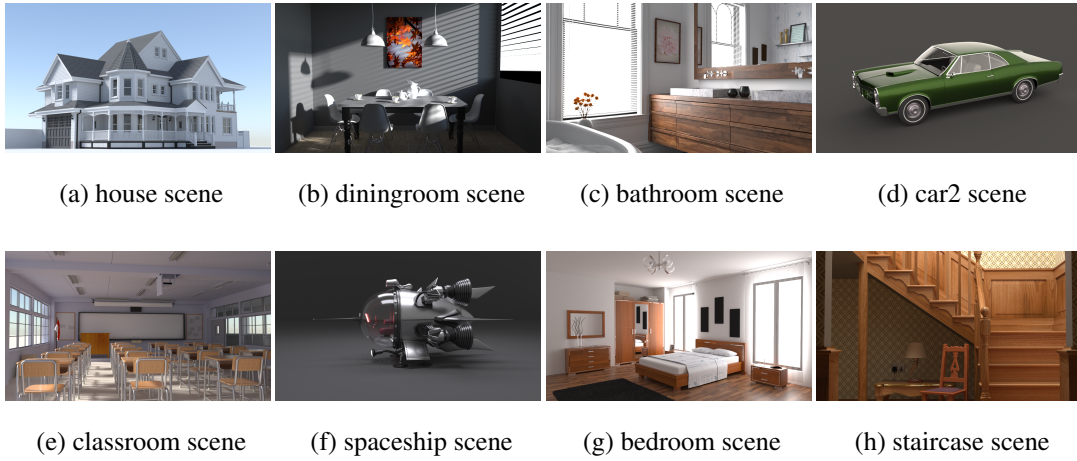


Figure 4.1: Tungsten scenes on which the training dataset is based.

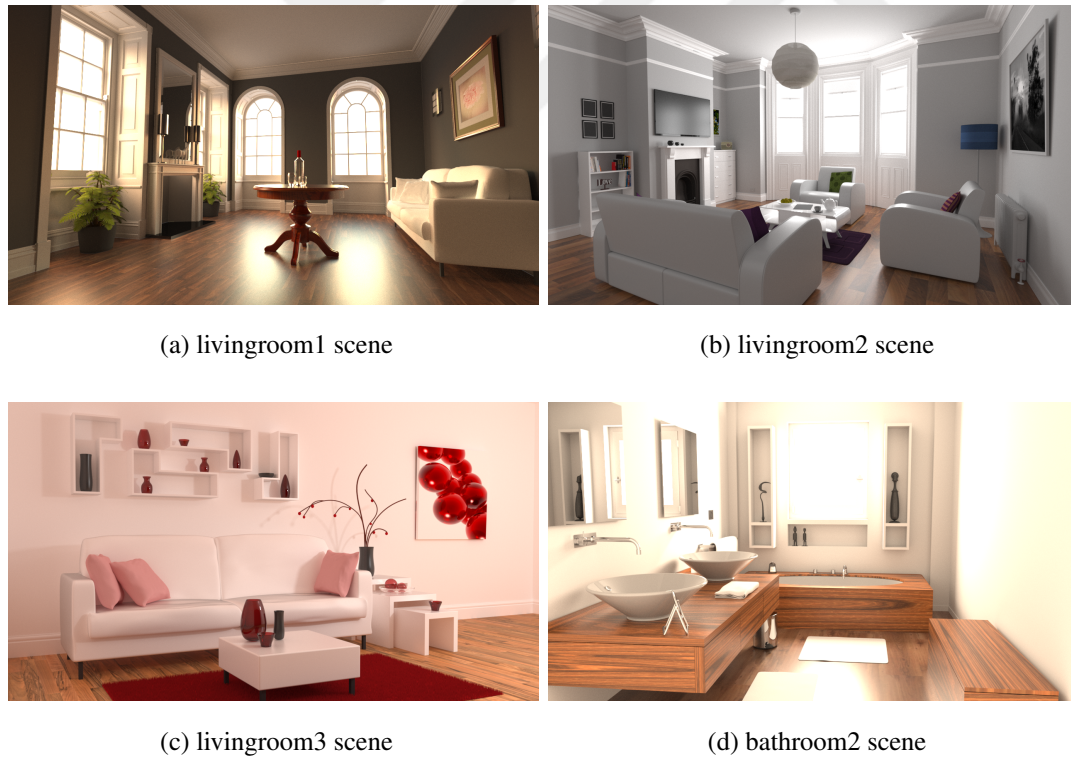


Figure 4.2: Tungsten test scenes used for the evaluation of the experiments.

rendered at the sample counts of 128, 256, 512, and 1024, using 8192 SPP for the ground-truth image. Therefore, there are 5804 image pairs in this dataset, rendered at the original resolution, 1280x720. We use this dataset as the training dataset.

The test set we use is composed of 4 scenes that are different from the base scenes of the training set and selected from the publicly available Tungsten scenes. We evaluate the proposed pipeline using the renderings of those 4 scenes with different sample per pixel values. Figure 4.2 visualizes the renderings of those 4 scenes.

We have used the edited version of the Tungsten renderer provided by the creators of the training dataset. The training images of the dataset are generated using this scripted Tungsten renderer. This Tungsten version appends the by-products of the rendering process as additional image channels to the output render. Those channels include the diffuse and specular components, the scene feature buffers, such as the normal map, depth map, and albedo map of the scene, along with the calculated per pixel variances of those additional data, to be used as guidance during the denoising operation.

4.2 Guided Filter Experiment

In this section, we report the settings and findings of the experiment conducted by using KPCN as the Monte Carlo denoiser and Guided Filter as the guided upsampler.

4.2.1 KPCN Training Details

For the Monte Carlo denoising step of this experiment, we apply the network weights provided by [2], which are trained and converged on the aforementioned Monte Carlo denoising dataset at the original resolution. The KPCN denoiser is used in the kernel-prediction mode instead of the direct-pixel-prediction mode. Therefore, it learns to denoise the images by predicting a filter kernel and applying it. This mode allows the KPCN denoiser, whose weights are trained on the original resolution dataset, to be used to denoise the low-resolution renderings using the same weights in the Monte Carlo denoising step of the proposed pipeline.

Before the training of the KPCN denoiser, it is set to sample 400 patches of size 65x65, including all the channels, from all image pairs of the training dataset as the preprocessing for the training phase. Then those patches are separated into diffuse and specular components together with the additional scene feature channels and forward-passed in a batch size of 100 during the training phase in both the diffuse and specular denoising networks of KPCN to be able to learn the noise characteristic of the Tungsten renderer accurately. The training is done on L1 loss, using ADAM optimizer with a learning rate of 1e-4 during the individual training of the networks, and 1e-6 during the fine-tuning stage where the output of both networks are combined to create the final RGB image and L1 loss is run on this final image compared to the ground-truth RGB. The training is converged after a total of 360000 iterations together with the fine-tuning stage.

4.2.2 Input Image Specifications

The baseline approach that we compare the results of this experiment with renders the images at the original resolution, 1280x720, but with a harshly reduced sample count, 8 SPP. Then, this image is given to the baseline KPCN denoiser that uses the same trained weights as the ones used in our experiment and gets denoised inaccurately with visual artifacts.

We run the experiment by first rendering the image at a 2x2 reduced resolution, 640x360, with an increased sample count, 32 SPP. This low-resolution image is then denoised using the KPCN denoiser, and the denoised diffuse and specular components are produced.

4.2.3 Guided Filter Settings

The guided filter algorithm [3] is used as the guided upsampler in this experiment. The denoised diffuse and specular components that are to be input to the upsampler are in low-resolution. The guidance image that will be used during the upsampling process is in the original resolution. Therefore, we first downsample the guidance image using bicubic interpolation to the low-resolution. The guided filter algorithm is

run for both components at the low-resolution to calculate the a and b coefficient matrices of the guided filters, detailed in Chapter 2, in low-resolution. Those coefficient matrices are then upsampled using bicubic interpolation to the original resolution. Then a linear transformation is applied to the original resolution guidance image using the upsampled a and b to produce the upsampled output.

To find out the best-performing r and ϵ parameters of the guided filter, we have tested increasing values of them on all the test scenes. Table 4.1 shows the detailed results of this test using MSE and SSIM metrics by running the proposed pipeline using the corresponding r and ϵ values.

According to this test’s findings, the guided filter performs best in MC upsampling case when $r=1$ and $\epsilon=1e-4$ by producing metric scores that are better, primarily focusing on the SSIM results. Increasing both parameters causes the guided filter output blurry edges. Decreasing ϵ more causes the algorithm to produce overflowed values. Therefore, we have used those values for both diffuse and specular upsamplers to report this experiment’s final results.

4.2.4 HR Guidance Specifications

The HR guidance we have used is a 3-channel guidance image that is found by multiplying the scene feature buffers channel-wise to keep as many edges of the scene as possible by including each scene feature in the guidance image. Those scene features are listed as: (i) preprocessed 3-channel normal map, (ii) preprocessed 1-channel depth map, (iii) 3-channel albedo map. The single channel of the depth map is cloned so that the depth map has 3-channels before multiplication. We report the final results of this experiment using those combined guidances of each of the four test scenes. The combined guidances of each scene are visualized in Figure 4.3.

Table 4.1: Effects of parameter settings of Guided Filter of both the diffuse and specular guided upsamplers on the performance of the proposed pipeline. r is the window radius, and ϵ is the regularization parameter, constraining coefficient a from becoming too large. The metric results show the comparison of the final image, formed by the addition of upsampled diffuse and specular results, with the ground-truth.

Scene	r	ϵ	MSE	SSIM
livingroom1	1	1e-4	0.2564	0.9195
		1e-3	0.2402	0.9151
	2	1e-4	0.4256	0.9069
		1e-3	0.4188	0.8986
livingroom2	1	1e-4	0.1007	0.9367
		1e-3	0.1264	0.9335
	2	1e-4	0.1893	0.9279
		1e-3	0.2753	0.9208
livingroom3	1	1e-4	0.0005	0.9700
		1e-3	0.0005	0.9691
	2	1e-4	0.0005	0.9678
		1e-3	0.0006	0.9654
bathroom2	1	1e-4	0.6702	0.9242
		1e-3	0.5461	0.9174
	2	1e-4	0.9654	0.9096
		1e-3	1.0591	0.8972



(a) livingroom1 guidance



(b) livingroom2 guidance



(c) livingroom3 guidance



(d) bathroom2 guidance

Figure 4.3: Guidance images of the test scenes, created by multiplying each of the preprocessed scene feature buffers, which are 3-channel normal map, 3-channel albedo map, and 3-channel depth map. 3-channel depth map is generated by cloning its single channel into three channels. Preprocessing is done to pull their values into $[0, 1]$ range.

Table 4.2: Metric results of guided filter experiment. Each metric is calculated by comparing to the 8192 SPP ground-truth, LDR image is tone-mapped version of the HDR result, SSIM stands for mean SSIM value, figure number points to the figure that includes the resulting baseline, proposed pipeline, and ground-truth visuals of the corresponding scene.

Scene	Method	LDR SSIM	LDR PSNR	HDR SSIM	HDR-VDP Q Score	Figure No
livingroom1	Base	0.8856	31.5859	0.9129	8.8873	4.4
	Ours	0.8945	29.8924	0.9195	9.2774	
livingroom2	Base	0.9447	36.8402	0.9454	9.4450	4.5
	Ours	0.9407	35.4369	0.9367	9.4960	
livingroom3	Base	0.9597	38.0783	0.9585	9.4146	4.6
	Ours	0.9716	38.5494	0.9700	9.6613	
bathroom2	Base	0.9656	37.9993	0.9344	9.4762	4.7
	Ours	0.9574	35.2454	0.9242	9.5942	

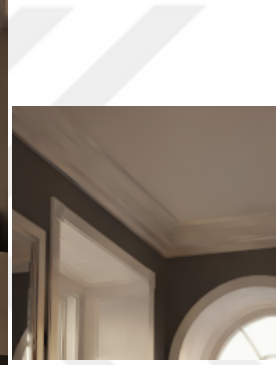
4.2.5 Results

In Table 4.2, the metric results of the guided filter experiment are provided. SSIM metric is the mean of Structural Similarity Index Measure of two images in comparison. PSNR metric indicates the peak signal-to-noise ratio of those images. HDR-VDP is a perceptual metric whose visuals denote the predicted probabilities of detection of the differences between two images by the user, and Q score indicates the quality prediction of the image as mean-opinion-score. During the calculations, HDR-VDP parameters are set as follows. The monitor is set to be a 30-inch display with a resolution of 3840x2160, and the viewing distance is set to be 0.5 meters away from the monitor. The version of HDR-VDP we have used is HDR-VDP-3.0.6 [24], and it is used in the side-by-side mode. We report the HDR-VDP visuals using the P_det results of the HDR-VDP calculations. LDR images are produced by applying Reinhard tone-mapping operator [25]. The table includes pointers to the resulting visuals of each scene by indicating the corresponding figure numbers. Moreover, we provide the visual results of the HDR-VDP operation in the following figures:

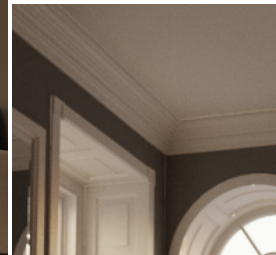
- Figure 4.8 for the livingroom1 scene,
- Figure 4.9 for the livingroom2 scene,
- Figure 4.10 for the livingroom3 scene,
- Figure 4.11 for the bathroom2 scene.



(a) Result of the baseline approach



(b) Result of the proposed pipeline

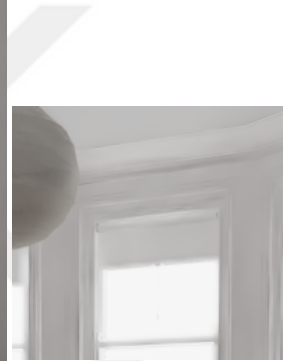


(c) Ground-truth result

Figure 4.4: Results of the livingroom1 scene for the guided filter experiment.



(a) Result of the baseline approach



(b) Result of the proposed pipeline



(c) Ground-truth result

Figure 4.5: Results of the livingroom2 scene for the guided filter experiment.



(a) Result of the baseline approach



(b) Result of the proposed pipeline



(c) Ground-truth result

Figure 4.6: Results of the livingroom3 scene for the guided filter experiment.



(a) Result of the baseline approach

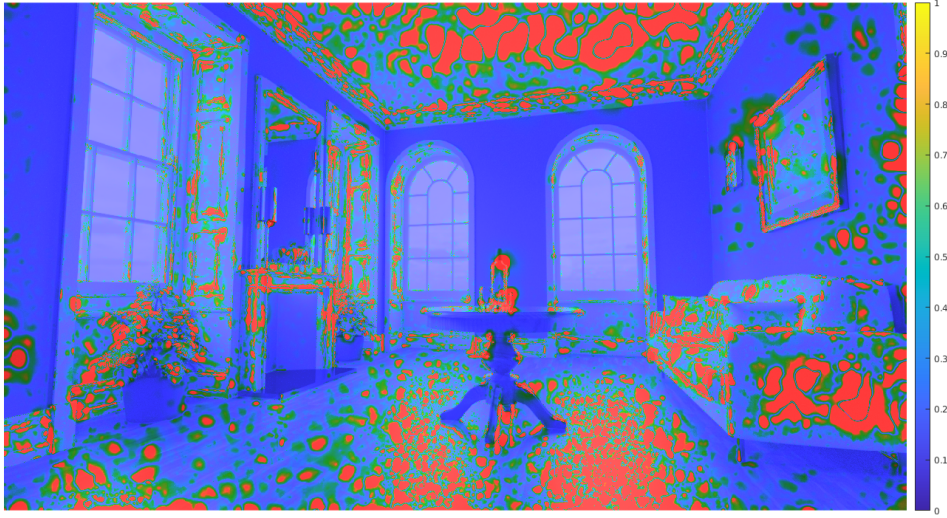


(b) Result of the proposed pipeline

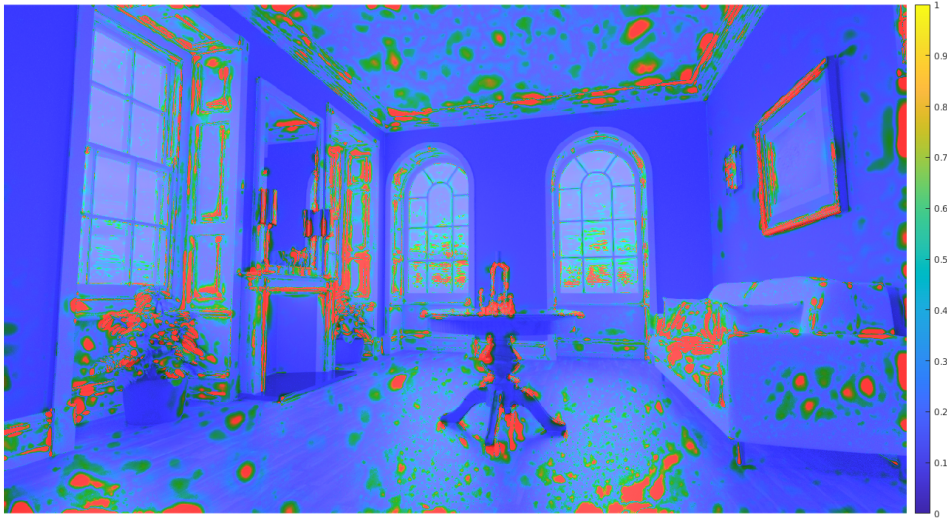


(c) Ground-truth result

Figure 4.7: Results of the bathroom2 scene for the guided filter experiment.

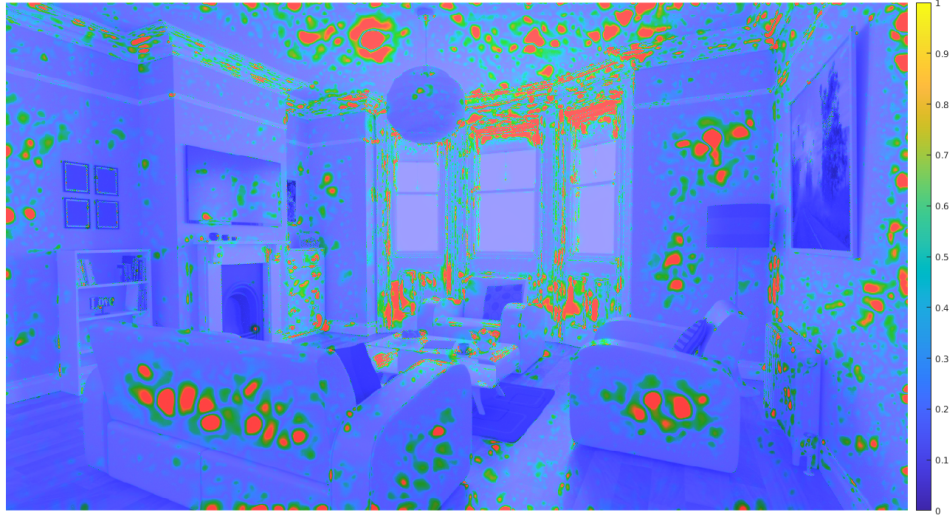


(a) Baseline, $Q = 8.8873$

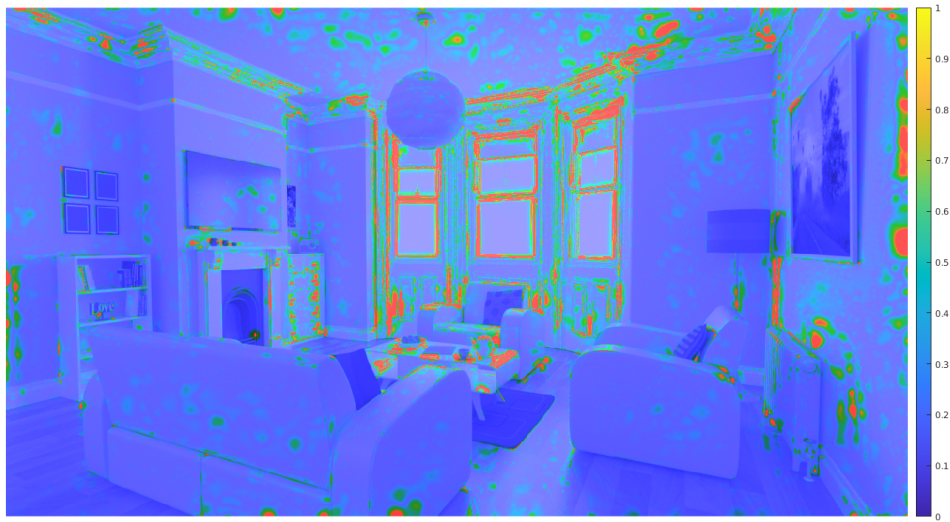


(b) Ours, $Q = 9.2774$

Figure 4.8: HDR-VDP results of the livingroom1 scene in comparison to the ground-truth; higher color values indicating higher probability of detection of visual differences, and Q score indicating the predicted image quality over 10 as mean-opinion-score.

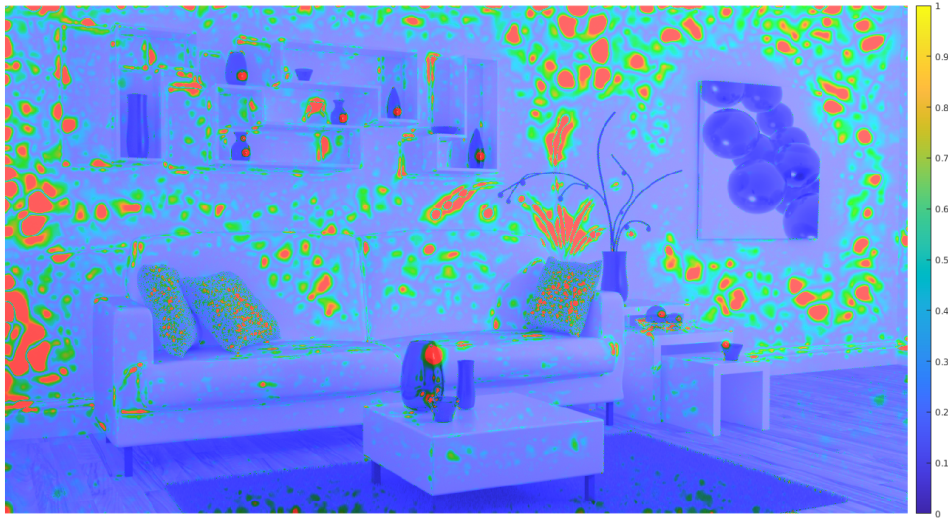


(a) Baseline, $Q = 9.4450$

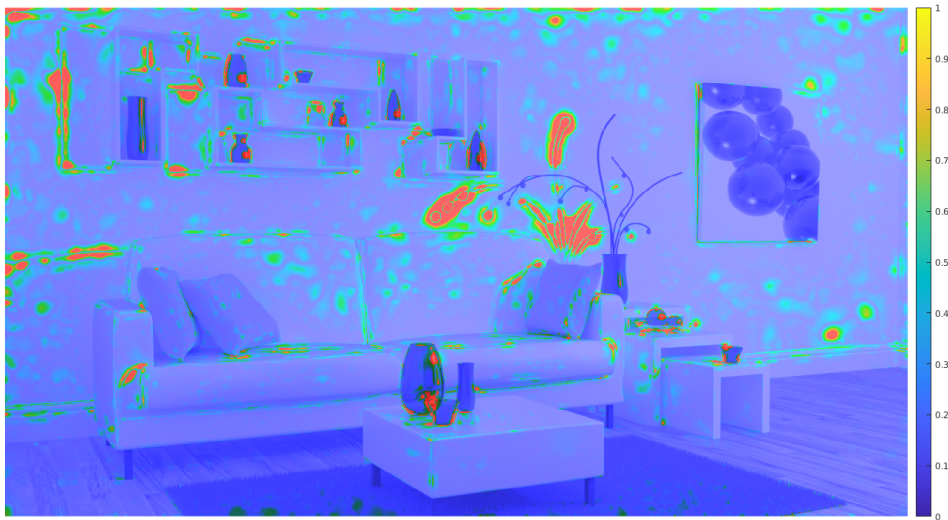


(b) Ours, $Q = 9.4960$

Figure 4.9: HDR-VDP results of the livingroom2 scene in comparison to the ground-truth; higher color values indicating higher probability of detection of visual differences, and Q score indicating the predicted image quality over 10 as mean-opinion-score.

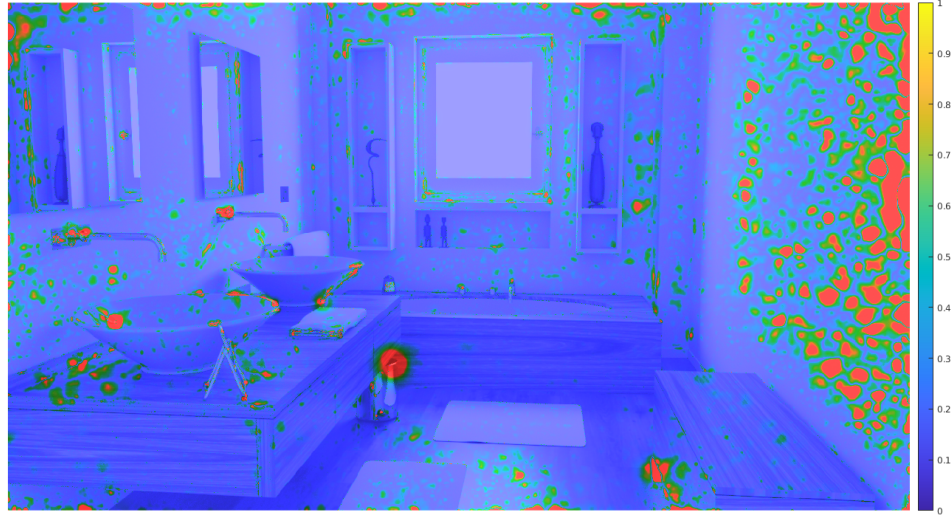


(a) Baseline, $Q = 9.4146$

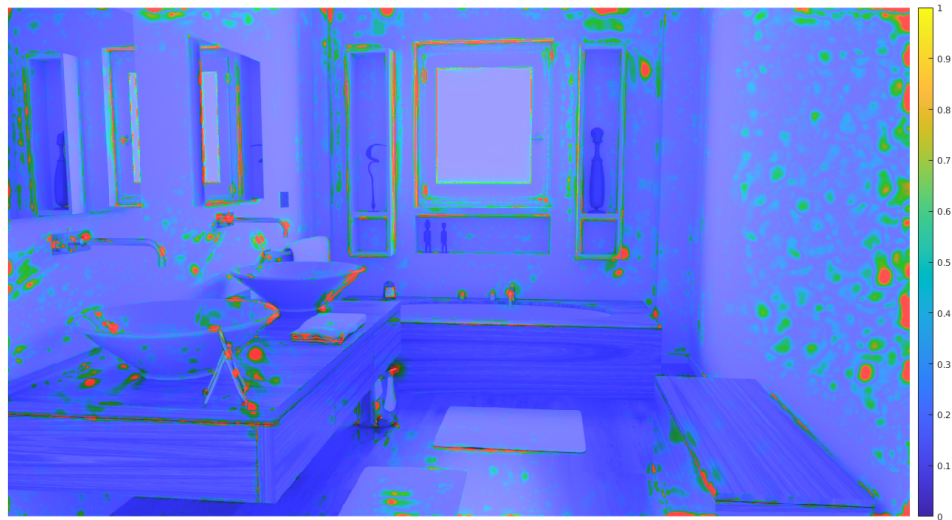


(b) Ours, $Q = 9.6613$

Figure 4.10: HDR-VDP results of the livingroom3 scene in comparison to the ground-truth; higher color values indicating higher probability of detection of visual differences, and Q score indicating the predicted image quality over 10 as mean-opinion-score.



(a) Baseline, $Q = 9.4762$



(b) Ours, $Q = 9.5942$

Figure 4.11: HDR-VDP results of the bathroom2 scene in comparison to the ground-truth; higher color values indicating higher probability of detection of visual differences, and Q score indicating the predicted image quality over 10 as mean-opinion-score.

4.3 Deep Guided Filter Experiment

DGF has a learnable implementation of the guided filter. Moreover, it includes a task-specific learnable transformation function applied to the input guidance to benefit from it better. As we have two different tasks, the upsampling of the diffuse and specular components, it is an intriguing idea to study our pipeline with DGF as the guided upsampler to see whether the guided upsampling step of the pipeline can be improved using these learnable models. In this section, we report the settings and findings of the experiment conducted by using KPCN as the Monte Carlo denoiser and DGF as the guided upsampler.

4.3.1 Dataset Splitting Strategy

All three of KPCN of the baseline, KPCN of our pipeline, and DGF networks require datasets to be trained. Moreover, KPCN used in our approach must have a different dataset than DGF. This is because the DGF training dataset is required to be forward-passed (denoised) by the KPCN of our method first. Therefore, KPCN should not have seen this training dataset of DGF to prevent memorized results during the denoising process. Therefore, we have split the Disney dataset as follows:

- KPCN of our pipeline is trained with half of the scenes in the dataset, rendered with 512 and 1024 SPP. Even though our KPCN denoises low-resolution images, we have trained it using the original resolution training dataset to prevent training data loss due to downsampling. As KPCN learns to predict denoising kernels instead of directly predicting the output pixels, this training approach does not negatively affect the denoising process.
- Baseline KPCN is trained with the same half of the scenes in the dataset but rendered with 128 and 256 SPP. We employ this approach as the baseline approach is used to denoise lower SPP images than the proposed pipeline during the evaluation phase. Therefore, this separation grants a fair comparison of the denoising capabilities of both the baseline and the proposed pipeline.
- The other half of the scenes in the dataset rendered with 128, 256, 512, and

1024 SPP are denoised using the KPCN of our pipeline, and then they are given to the DGF as the training dataset.

4.3.2 KPCN Training Details

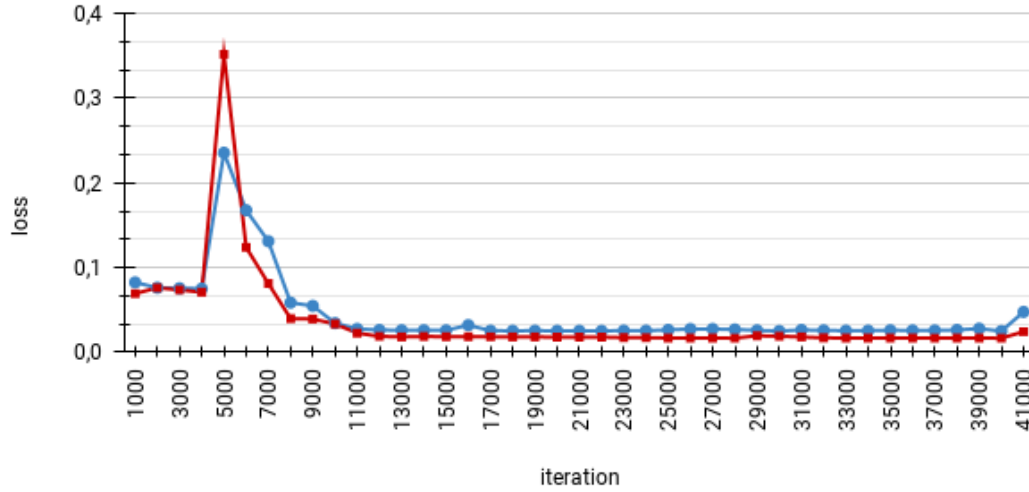
We have applied the same training parameters used in the guided filter experiment and detailed in 4.2.1, for both the baseline’s KPCN and our KPCN. As the dataset is decreased harshly, we have tried sampling 800 patches per image instead of 400 to increase training data. However, this did not result in better training. Therefore, we kept the amount of 65x65 patches sampled from each 1280x720 image of the dataset at 400.

Each KPCN denoiser of the baseline and the proposed pipeline has two identical-but-separately-trained networks inside them for the diffuse and specular components. Those networks of each KPCN denoiser are trained separately first. Later, they are jointly fine-tuned.

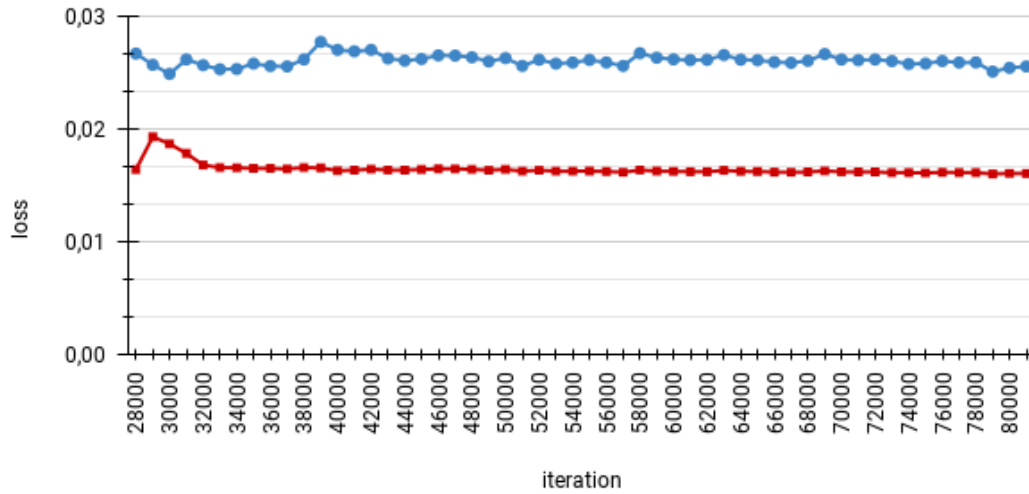
We render all of the four test scenes with 128 SPP and 512 SPP at the original resolution and use them to report the test errors during the training of the baseline KPCN and our KPCN, respectively. The test errors reported in the KPCN training charts (see Figure 4.12 and Figure 4.13) are calculated using those test images.

Figure 4.12a shows the training and test error chart of the separate training phase of the baseline KPCN. Figure 4.12b shows the training and test error chart of the joint fine-tuning phase of the baseline KPCN. The blue line shows the training error, and the red line shows the test error. Training error is higher due to the use of complex-to-denoise scenes for the calculation of this error. The fine-tuning is started at the iteration checkpoint 39K of the separate training phase as this checkpoint results in a lower test error. However, the networks could not learn much from the fine-tuning stage, as seen in the figure. After the joint training ends, we choose iteration checkpoint 71K to produce the final results of this experiment’s baseline approach.

Figure 4.13a shows the training and test error chart of the separate training phase of our KPCN. Figure 4.13b shows the training and test error chart of the joint fine-tuning phase of our KPCN. The blue line shows the training error, and the red line shows

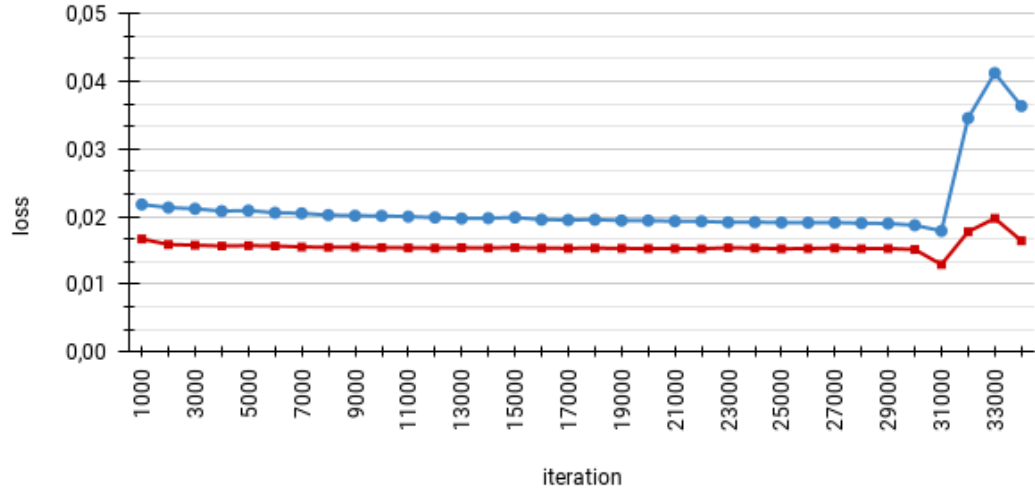


(a) Separate training

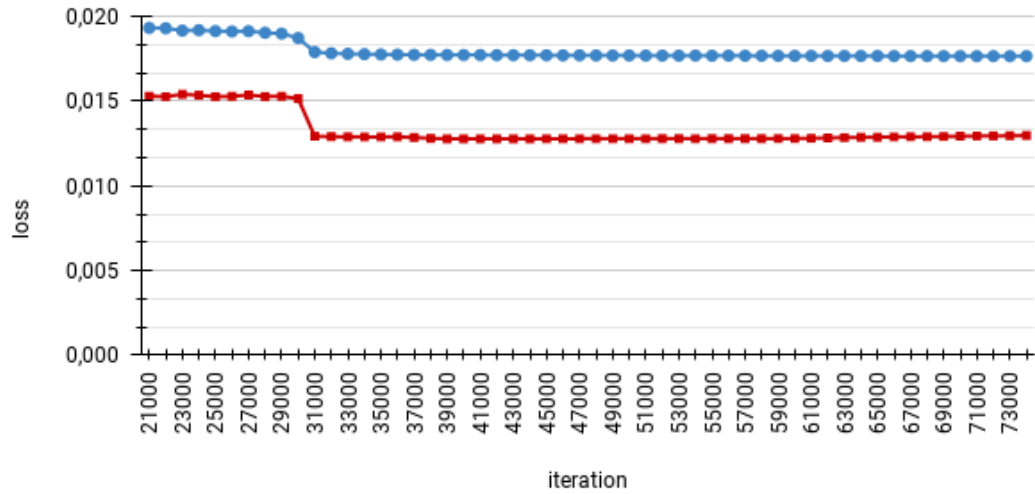


(b) Joint training of diffuse and specular networks after iteration checkpoint 39K, chart zoomed in

Figure 4.12: Training and validation error charts of the baseline KPCN on L1 loss, trained using halves of 128 and 256 SPP images of the dataset. The error calculation is done by adding the denoised output of diffuse and specular networks of baseline KPCN to get the final image and then compare it to the ground-truth. The blue line shows training validation error, and the red line shows test error. Training validation error being larger due to complex scenes being used to calculate the error. We use the 71K iteration checkpoint to produce the baseline approach results, as it also results in a better SSIM score (not shown).



(a) Separate training



(b) Joint training of diffuse and specular networks after iteration checkpoint 31K, chart zoomed in

Figure 4.13: Training and validation error charts of our KPCN on L1 loss, trained using halves of 512 and 1024 SPP images of the dataset. The error calculation is done by adding the denoised output of diffuse and specular networks of baseline KPCN to get the final image and then compare it to the ground-truth. The blue line shows training validation error, and the red line shows test error. Training validation error being larger due to complex scenes being used to calculate the error. We use the 72K iteration checkpoint to produce the baseline approach results, as it also results in a better SSIM score (not shown).

the test error. Training error is higher due to the use of complex-to-denoise scenes for the calculation of this error. The fine-tuning is started at the iteration checkpoint 31K of the separate training phase as this checkpoint results in a lower test error. However, the networks could not learn much from the fine-tuning stage, as seen in the figure. After the joint training ends, we choose iteration checkpoint 72K to produce the results of the proposed approach’s low-resolution denoising step.

4.3.3 HR Guidance Specifications

We have edited the network structure of DGF to increase the guidance input channel from 3 to 7. Those seven channels include the preprocessed HR guidance images in the range $[0, 1]$. Those guidances are 3-channel albedo map, 3-channel normal map, and 1-channel depth map, detailed in Section 4.2.4. We have used the depth map as 1-channel without cloning it into 3-channels. Those preprocessed HR scene feature buffers are then merged into a 7-channel image to be input to the DGF. Those scene feature buffers already exist in additional channels of 1024 SPP images of the dataset. Therefore, they are extracted from those 1024 SPP images to be used during the training.

4.3.4 DGF Training Details

The DGF training dataset is required to be in low-resolution. Therefore, we have experimented with:

- Denoising the DGF dataset at the original resolution, and then downsampling the denoised images 2x2 by using the nearest neighbor (NN) downsampling strategy,
- Downsampling the original resolution images 2x2 by using the NN downsampling strategy before applying denoising operation.

However, both of those approaches resulted in a similar metric score. Therefore, we choose to downsample beforehand and then denoise the images to reduce the time required to denoise the dataset of DGF.

We choose to employ the nearest neighbor downsampling strategy to keep the Tungsten renderer characteristics intact in the downsampled image, effectively not using 3 of 4 pixel values in a 2x2 window of the images while downsampling and using the remaining 1 pixel as is without any interpolation. This approach simulates rendering at a low-resolution with the same sample count.

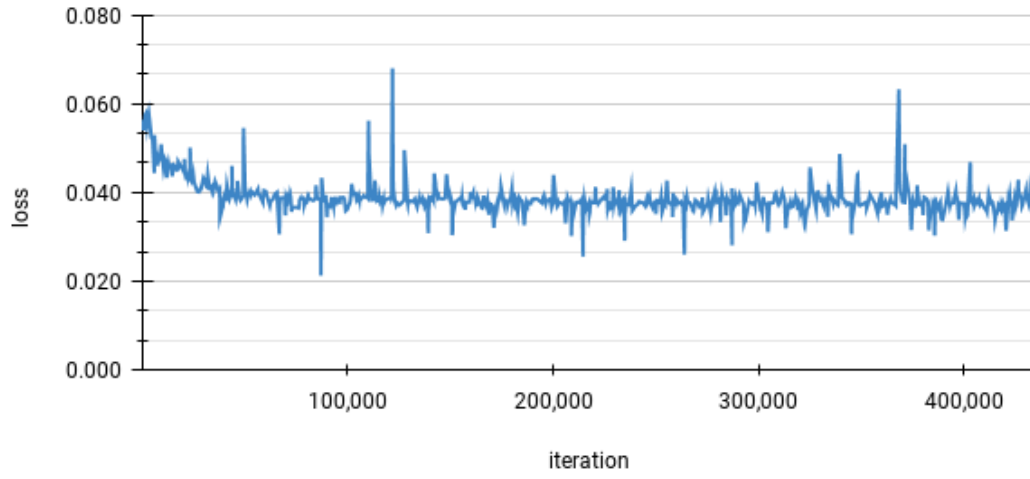
As the DGF training dataset includes diffuse and specular components for each image, we have trained two copies of the DGF architecture, one for each of those components. As those components have different characteristics, we have followed this separation of components approach to get a better upsampling result in the final. Both networks are trained on L1 loss, using a batch size of 1. At each iteration, the networks take 1 image input as a whole with its guidance data, without following any patch-based approach. The training is done 150 epochs for each of the networks, with no gradient clipping, and no fine-tuning phase being included. ADAM optimizer is used with a learning rate of $1e-4$ throughout the training process.

We have provided the training error charts for the diffuse DGF and specular DGF in Figure 4.14a and Figure 4.14b, respectively. However, we did not have the opportunity to calculate the test error of the training process. Therefore, we have used the weights found at the end of epoch 150 to evaluate the 4 test images to produce the final visuals.

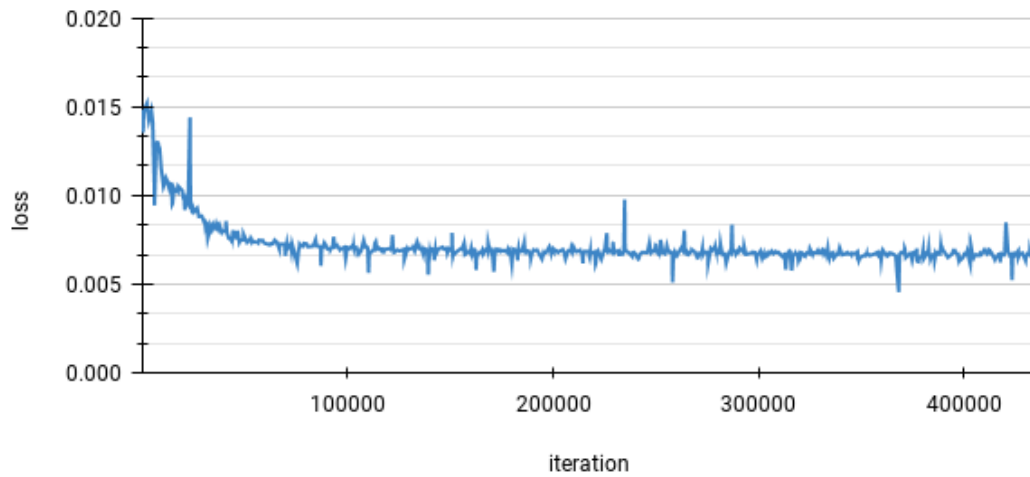
4.3.5 Results

We render all of 4 test scenes with 8 SPP at the original resolution, 1280x720, and with 32 SPP at the reduced resolution, 640x360, and use them to evaluate and report the performance of the baseline and the proposed pipeline used in this experiment, respectively.

The numerical results of the experiment are provided in Table 4.3, together with the numerical results of the first experiment where we use the guided filter. Those calculations are made with the same metrics used in the first experiment to facilitate comparisons between the two experiments. The table includes pointers to the resulting visuals of each scene by indicating the corresponding figure numbers. Moreover,



(a) Train error chart of the diffuse DGF



(b) Train error chart of the specular DGF

Figure 4.14: Training error charts of both diffuse and specular DGF upsamplers. Both are trained on L1 loss using the diffuse and specular components of the remaining halves of 128, 256, 512, and 1024 SPP images of the dataset.

we provide the visual results of the HDR-VDP operation in the following figures:

- Figure 4.19 for the livingroom1 scene,
- Figure 4.20 for the livingroom2 scene,
- Figure 4.21 for the livingroom3 scene,
- Figure 4.22 for the bathroom2 scene.

The performance of the baseline KPCN of this experiment is lower than the baseline KPCN of the GF experiment. As we have split the training dataset to perform a fair comparison between the baseline KPCN and the proposed pipeline’s KPCN during this experiment, the performance of the baseline KPCN of this experiment could not be as good as the fully-trained baseline KPCN of the guided filter experiment where the dataset is used for the training as a whole. Therefore, the visual results that the proposed pipeline generates at the end of this DGF experiment has some noisy parts that could not be denoised due to KPCN denoiser not being able to be converged to a low error during the training. Moreover, the DGF training is also done with low amount of dataset. Therefore, those noisy parts are spread out after the upsampling is done. Those noisy parts can be seen at the legs of the table in Figure 4.15b, on the TV in Figure 4.16b, and around the small hand mirror on the counter in Figure 4.18b.

There exists a 0.67 points difference between the HDR-VDP Q scores of the Base (GF) and Base (DGF) results of the livingroom1 scene in Table 4.3. When this difference is added to Ours (DGF) of the livingroom1 scene, a higher Q score than Ours (GF) is achieved. We can argue that when the baselines of both experiments are trained in equal conditions, Ours (DGF) will result in a higher HDR-VDP Q score than Ours (GF), meaning that we would achieve better performance by using the deep guided filter instead of the guided filter as the guided upsampler in the proposed pipeline. The same idea also applies to the livingroom3 scene with a 0.18 points difference between its Base (GF) and Base (DGF) results. This argument can be evaluated after employing two different but diverse datasets and granting equal convergence during the training of both baselines.

Table 4.3: Metric results of deep guided filter experiment (DGF) together with the guided filter experiment (GF). Each metric is found by comparing to the 8192 SPP groundtruth, LDR image is tone-mapped version of the HDR result, SSIM stands for mean SSIM value, figure number points to the figure that includes the resulting baseline, proposed pipeline, and ground-truth visuals of the corresponding scene. Bold text indicates higher score than the baseline used in the corresponding experiment.

Scene	Method	LDR SSIM	LDR PSNR	HDR SSIM	HDR-VDP Q Score	Figure No
livingroom1	Base (GF)	0.8856	31.5859	0.9129	8.8873	4.4
	Ours (GF)	0.8945	29.8924	0.9195	9.2774	
	Base (DGF)	0.8496	28.5898	0.8870	8.2197	4.15
	Ours (DGF)	0.8515	27.9254	0.8883	8.7185	
livingroom2	Base (GF)	0.9447	36.8402	0.9454	9.4450	4.5
	Ours (GF)	0.9407	35.4369	0.9367	9.4960	
	Base (DGF)	0.9344	35.5736	0.9348	9.2137	4.16
	Ours (DGF)	0.9184	33.8427	0.9194	9.2635	
livingroom3	Base (GF)	0.9597	38.0783	0.9585	9.4146	4.6
	Ours (GF)	0.9716	38.5494	0.9700	9.6613	
	Base (DGF)	0.9548	37.5303	0.9523	9.2347	4.17
	Ours (DGF)	0.9592	37.1072	0.9561	9.5209	
bathroom2	Base (GF)	0.9656	37.9993	0.9344	9.4762	4.7
	Ours (GF)	0.9574	35.2454	0.9242	9.5942	
	Base (DGF)	0.9541	36.2531	0.9191	9.2549	4.18
	Ours (DGF)	0.9351	33.8661	0.9041	9.3088	



(a) Result of the baseline approach



(b) Result of the proposed pipeline



(c) Ground-truth result

Figure 4.15: Results of the livingroom1 scene for the deep guided filter experiment.



(a) Result of the baseline approach



(b) Result of the proposed pipeline



(c) Ground-truth result

Figure 4.16: Results of the livingroom2 scene for the deep guided filter experiment.



(a) Result of the baseline approach



(b) Result of the proposed pipeline



(c) Ground-truth result

Figure 4.17: Results of the livingroom3 scene for the deep guided filter experiment.



(a) Result of the baseline approach

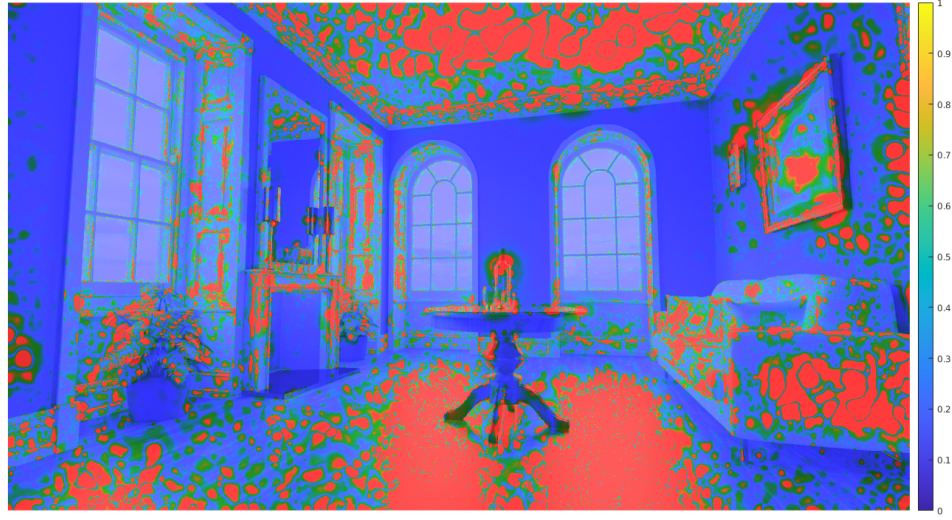


(b) Result of the proposed pipeline

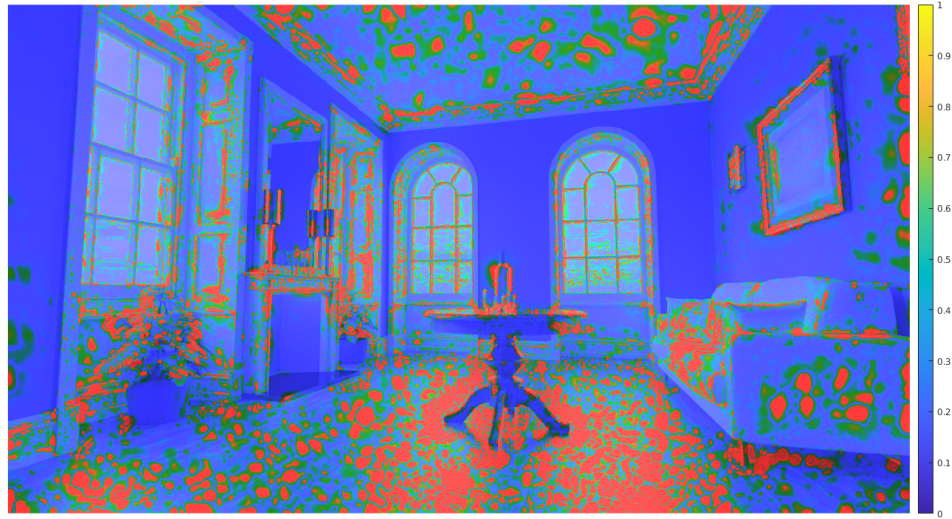


(c) Ground-truth result

Figure 4.18: Results of the bathroom2 scene for the deep guided filter experiment.

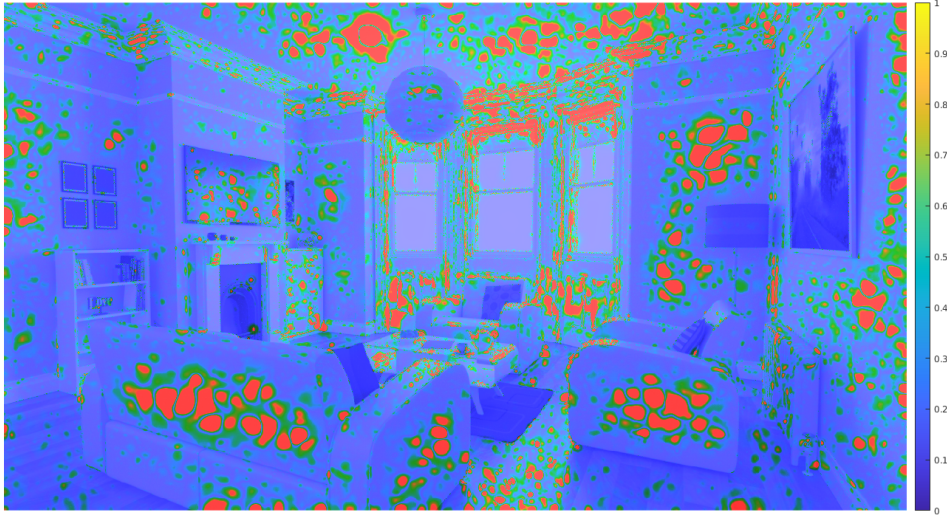


(a) Baseline, $Q = 8.2197$

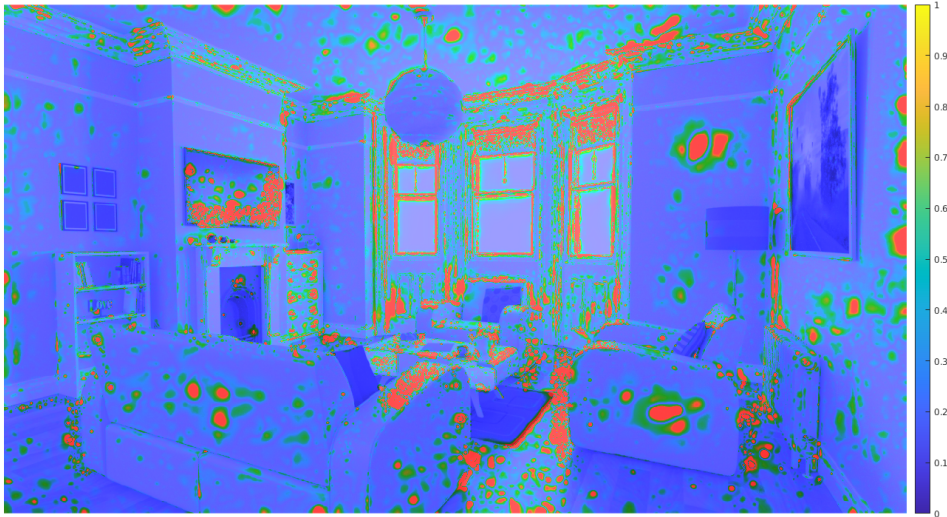


(b) Ours, $Q = 8.7185$

Figure 4.19: HDR-VDP results of the livingroom1 scene in comparison to the ground-truth; higher color values indicating higher probability of detection of visual differences, and Q score indicating the predicted image quality over 10 as mean-opinion-score.

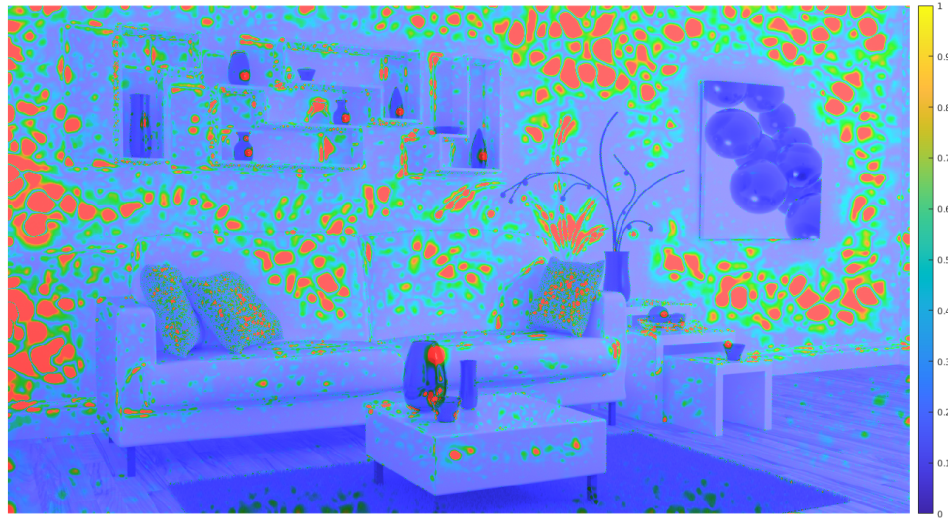


(a) Baseline, $Q = 9.2137$

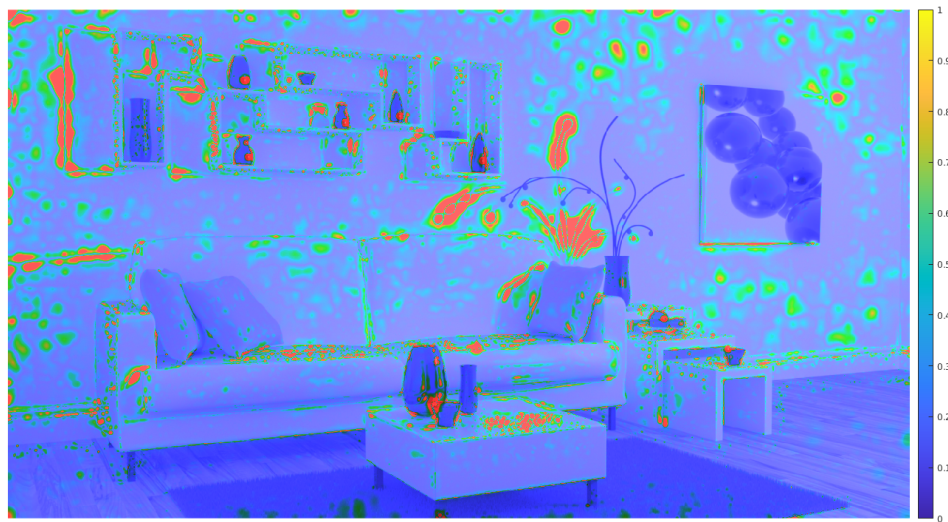


(b) Ours, $Q = 9.2635$

Figure 4.20: HDR-VDP results of the livingroom2 scene in comparison to the ground-truth; higher color values indicating higher probability of detection of visual differences, and Q score indicating the predicted image quality over 10 as mean-opinion-score.

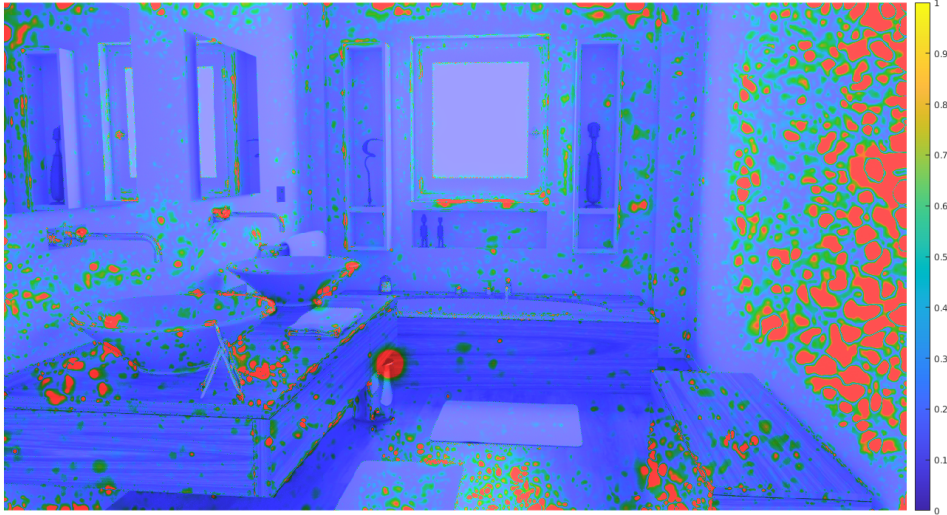


(a) Baseline, $Q = 9.2347$

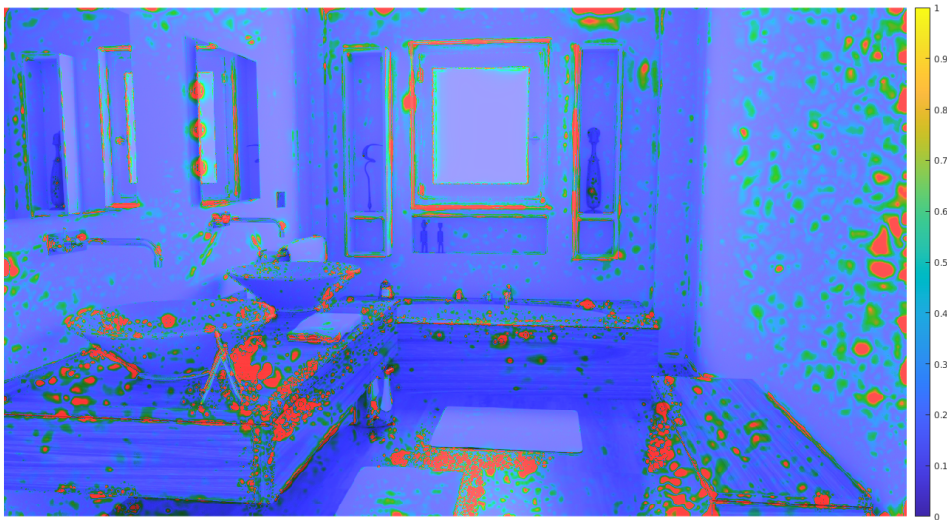


(b) Ours, $Q = 9.5209$

Figure 4.21: HDR-VDP results of the livingroom3 scene in comparison to the ground-truth; higher color values indicating higher probability of detection of visual differences, and Q score indicating the predicted image quality over 10 as mean-opinion-score.



(a) Baseline, $Q = 9.2549$



(b) Ours, $Q = 9.3088$

Figure 4.22: HDR-VDP results of the bathroom2 scene in comparison to the ground-truth; higher color values indicating higher probability of detection of visual differences, and Q score indicating the predicted image quality over 10 as mean-opinion-score.

CHAPTER 5

DISCUSSIONS

In this chapter, we comment on the experimental evaluation phase of the proposed pipeline, and we discuss each discussion point in separate sections and subsections throughout the chapter.

5.1 Experiment Overview

Overall, the proposed pipeline performs promisingly well for rendering and denoising low SPP Monte Carlo renderings compared to the baseline. Our approach results in a higher HDR-VDP Q score for each of the test scenes in both experiments, along with comparably good SSIM and PSNR metrics on those scenes. Moreover, the visuals generated by the HDR-VDP visual metric show that for an observer, compared to the output of the baseline, the output of the proposed pipeline has less amount of different-from-ground-truth areas that have higher probabilities of detection, indicating a better result.

We have also experimented with the 4x4 downsampling setting for the proposed pipeline. In this setting, we have multiplied the sample count by 16 to render the low-resolution Monte Carlo image in the same time budget of rendering at the original resolution with too harshly decreased sample count. However, we could not achieve good results using this setting. Therefore, we have run the experiments using only the 2x2 downsampling setting, multiplied the sample count by 4, and reported the results. Finding the downsampling rate for optimal performance is left as an intriguing idea to be studied by comprehensively evaluating various downsampling rates.

Creating a Monte Carlo dataset, especially the rendering of the ground-truth images at excessive sample counts, requires a vast amount of time. This is why we have used the Disney dataset, detailed in Section 4.1. One discussion point is whether the baseline approach Monte Carlo denoising can learn to perform better at the harshly decreased sample counts than the proposed pipeline if trained with a low SPP dataset. Even though we had a dataset created from those harshly decreased sample count images, the denoiser’s capacity may not be enough to accurately learn to handle the excessive amount of noise in the rendering outputs.

We have conducted the deep guided experiment to achieve better metric results than the guided filter experiment by learning a better guided upsampling. However, in the deep guided filter experiment, even though the HDR-VDP results of the proposed pipeline are better than its retrained baseline, and also the SSIM results are higher on half of the test scenes, the KPCN denoiser and DGF upsamplers of the deep guided filter experiment could not be trained sufficiently to produce a better quality output than the guided filter experiment’s. Thus, resulting in a lower performance than the guided filter experiment. In the following subsections, we first discuss the importance of the guidance on the denoising and upsampling of the renderings. Then, we discuss why the deep guided filter experiment resulted in a lower result than the guided filter experiment, respectively.

5.1.1 Lack of Guidance

In the experiments conducted, the guided upsampling process blurs the scene details that are not available in the guidance image, such as soft shadows and caustics. The same limit applies to the Monte Carlo denoising step. If the training dataset of KPCN does not include a specific scene detail, the diffuse and specular networks cannot learn it and blur it out, confusing it with the scene noise mistakenly. Therefore, we have used all the auxiliary scene feature buffers, normal map, albedo map, and depth map in our experiments as guidance to keep as much scene information as possible. Alongside this, the KPCN denoiser also leverages additional channels in the input image, such as per-pixel variance calculations of the diffuse component, specular component, and those auxiliary scene feature buffers during the denoising.

5.1.2 Lack of Dataset

The main reason for the deep guided filter experiment resulting in an unsatisfactory performance for the baseline KPCN denoiser and the proposed pipeline’s KPCN denoiser and DGF upsampler is the division of the dataset and distribution of it to all networks to be able to train them without showing the dataset of DGF to our KPCN denoiser to prevent memorization. This dataset division resulted in insufficient training data for those networks, and their training could not converge to an expectedly-lower loss value. Moreover, this dataset division also includes the SPP-wise division for the baseline’s KPCN and the proposed pipeline’s KPCN. 128 and 256 SPP dataset are used to train the baseline KPCN, and 512 and 1024 SPP dataset is used for the training of the KPCN of the proposed pipeline. Therefore, both the baseline KPCN and the proposed pipeline’s KPCN are now trained on one forth of the dataset of the baseline KPCN and the proposed pipeline’s KPCN of the guided filter experiment, their performance being cut down.

Due to the lack of a training dataset with diverse sample counts, both the baseline and our KPCN denoisers used in the deep guided filter experiment could not successfully be trained to perform better and correctly extrapolate to the low sample count test scenes that we have used, 8 SPP and 32 SPP, respectively for each denoisers, as the lack of dataset prevented the networks from learning noise characteristics of different sample counts.

Due to the lack of dataset in the deep guided filter experiment, we could not study it more and provided the results by training the diffuse and specular DGF upsamplers only once. Moreover, as we did not have the test error chart of those upsamplers’ training phase, we have used the network checkpoint where the training error is low enough without considering over-fitting. A more comprehensive study could have been done with a separate Monte Carlo dataset for the training of the DGF upsamplers so that the Disney dataset could be used only to train the baseline KPCN and our KPCN, granting a better comparison with the guided filter experiment results.

Therefore, as the next step of the deep guided filter experiment and this thesis study, we believe that constructing a deep network model for joint denoising and guided

upsampling of Monte Carlo images using original resolution scene feature buffers as multi-channel guidance can be constructed by merging the KPCN and Deep Guided Filter network architectures. Training it end-to-end using the Disney dataset as a whole without splitting for two networks, as we did in the deep guided filter experiment, may perform better than our two experiment configurations by alleviating this splitting of the dataset.

5.2 Runtime Performance

In this section, we provide the detailed timing results of the Monte Carlo rendering, denoising, and guided upsampling steps. We also provide the hardware configurations that we have used to conduct the experiments.

The primary motivation of this study is that we can upsample an accurately denoised low-resolution Monte Carlo image to the original resolution using the original resolution scene feature buffers as guidance and get better results than the baseline. Those buffers are efficiently rendered in a separate rendering pass. Therefore, the additional time cost of their rendering is small and acceptable when their tasks in the pipeline are considered. We assume that those scene feature buffers are separately rendered in that short amount and ready to be used in the proposed pipeline.

We have used two hardware configurations during the experimental evaluation of this thesis work. We call those configurations C1 and C2 and detail them as follows:

- **C1:**
 - CPU: Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz
 - GPU: NVIDIA Quadro K420
 - RAM: 64 GB
- **C2:**
 - CPU: Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz
 - GPU: NVIDIA GeForce RTX 2080 SUPER
 - RAM: 32 GB

We use configuration C1 to render the test images with the Tungsten renderer. The test images are rendered at 1280x720 to evaluate the baseline approach and 640x360 to evaluate the proposed pipeline. We launch the renderer on 8 CPU threads to provide comparable timing results to the case when standard commodity hardware is used. We run the unoptimized Guided Filter implementation written in Python programming language [26] on this configuration. We run the Deep Guided Filter code and the KPCN code on configuration C2, as it has a better GPU for running deep learning applications.

Table 5.1 shows the detailed timing results of rendering the input images for both baseline and our pipeline, the denoising of the input images, and the guided upsampling phases. The results are shown in seconds and calculated without including the time costs of reading/writing the images from/to the disk. As can be seen in the table, ground-truth Monte Carlo rendering elapses hours for each test scene. The differences in the rendering time costs of the ground-truth images stem from the light complexity of the scene. The original resolution image and low-resolution image are rendered at similar time costs as both have different SPP counts. Moreover, when the proposed pipeline is used with KPCN and DGF, the total run time of the pipeline is less than the baseline KPCN approach. This will also be the case for the KPCN and GF configuration of the proposed pipeline when an optimized GF implementation is employed, instead of the time-consuming unoptimized implementation that we have used during the experimental evaluation phase.

The use of the proposed pipeline is still tempting than casting more ray samples into the scene when the timing results are considered. Indoor scenes lit by indirect light coming inside from the windows are often high variance scenes. As the light complexity of the scene determines the rendering time of a visually-appealing output, those indoor scenes require a vast amount of SPP to render a compelling visual. As SPP increases, the render time increases. Therefore, our pipeline can be employed to denoise low resolution and higher-than-baseline SPP renderings of indoor scenes with complex light paths.

Table 5.1: Timing results in seconds. Input images are rendered using the Tungsten renderer. Original resolution image is rendered at 1280x720 with 8 SPP and fed to the baseline as input. Low-resolution image is rendered at 640x360 with 32 SPP and fed to our pipeline as input. Ground-truth images are rendered at the original resolution with 8192 SPP, elapsing tremendous amounts of render time. KPCN time cost depends on the resolution of the image. Therefore, we provide both baseline KPCN and our pipeline’s KPCN timing results. GF implementation we have used is an unoptimized one. Better timing results can be obtained using an optimized GF code. We assume that the original resolution scene feature buffers (normal map, depth map, albedo map) are generated with a separate rendering pass in a short time budget.

Scene	Input Render Time			Algorithm Run Time					
	Original Resolution (Base)	Low Resolution (Ours)	Ground-truth	KPCN (Base)	KPCN (Ours)	GF Diffuse	GF Specular	DGF Diffuse	DGF Specular
livingroom1	15.311	15.435	22,458.635	9.993	2.813	15.331	15.049	0.138	0.135
livingroom2	21.548	21.525	28,147.578	9.681	2.804	14.826	14.999	0.136	0.136
livingroom3	26.620	26.994	33,356.486	9.292	2.737	14.900	14.711	0.136	0.135
bathroom2	16.195	16.554	22,843.733	9.466	2.694	14.760	14.658	0.138	0.137

CHAPTER 6

CONCLUSIONS

In this thesis work, we have proposed a pipeline for the denoising and guided upsampling of Monte Carlo images that are rendered at a reduced resolution with relatively higher ray samples compared to being rendered at the original resolution using less number of ray samples in the same time budget. This approach’s advantage is that the low-resolution Monte Carlo image has a lower variance thanks to being rendered with a higher ray sample count. Therefore, it can be denoised more accurately. Moreover, auxiliary scene feature buffers are leveraged as guidance during the upsampling phase of the proposed pipeline. Those buffers are rendered at the original resolution in a short and often negligible time, as no in-scene recursive path tracing is done for the ray samples sent during the rendering of those features. As these auxiliary feature buffers are rendered with a higher number of ray samples, they have a low variance. Therefore, they provide the upsampling step with solid edge information, enhancing the robustness of the guided upsampling operation.

We have conducted two experimental evaluation studies on the proposed pipeline and reported the results by using the PSNR and SSIM metrics together with the HDR-VDP visual metric. In the first experiment, we have used the Guided Filter (GF) [3] for the guided upsampling step, where the original resolution auxiliary scene feature buffers, (i) normal map, (ii) albedo map, and (iii) depth map are used as guidances jointly. In the second experiment, we have used the Deep Guided Filter (DGF) [4] for the guided upsampling step with the same auxiliary scene buffers rendered at the original resolution. In both experiments, we have used Kernel-Predicting Convolutional Networks [2] as the Monte Carlo denoiser. The results of the experiments show that the proposed pipeline generates denoised and upsampled Monte Carlo images with

promisingly good quality compared to the baseline approach, which is rendering images without lowering the resolution, but with a lower ray sample count than the sample count used in the proposed pipeline.

Furthermore, by analyzing the timing results, we have found out that the proposed pipeline can perform the denoising and guided upsampling of the low-resolution Monte Carlo images in a vastly reduced time when compared to rendering the scenes at the original resolution using the ground-truth sample count. The proposed pipeline is also found to be operating in a shorter time budget than the baseline approach, which is rendering the Monte Carlo images at the original resolution and then applying denoising to them at this resolution.

6.1 Future Work

In the experimental evaluation phase of this thesis work, the importance of a Monte Carlo dataset is seen clearly, as the DGF experiment could not show better performance than the GF experiment due to the lack of dataset. New Monte Carlo datasets with diverse scenes can be created in the future to facilitate better training of both the Monte Carlo denoisers and the guided upsampler networks.

Building on top of this dataset requirement discussion, the study of Xu et al. [16], in which they have proposed the GAN-based state-of-the-art Monte Carlo denoiser, provides a new Monte Carlo dataset as the supplemental material. This dataset can be used to train the DGF upsampler networks of the DGF experiment so that the Disney dataset can be used as a whole to train the KPCN denoiser to make it converge to a better training loss and learn the denoising of high-variance Monte Carlo images accurately.

Using more than one dataset for the KPCN and DGF networks will also grant a fair comparison ground for both experiments. The baseline KPCNs of both the GF and DGF experiments and also the proposed pipeline’s KPCN in the DGF experiment would have trained on the same complete Disney dataset thoroughly, providing a better metric and visual results for the DGF experiment and enabling a more comprehensive comparison of the results of both experiments.

As another future work idea, both the denoising and the guided upsampling steps of the proposed pipeline can be merged in one deep network for joint denoising and guided upsampling of Monte Carlo images with the guidance of the original resolution scene feature buffers so that the need for separate datasets for the training of both the Monte Carlo denoising and deep guided upsampling steps can be alleviated. A single complete dataset can be used as a whole for the end-to-end training of this deep network that jointly denoises and upsamples low-resolution Monte Carlo renderings.

As discussed in Chapter 5, an intriguing idea to be studied is to find the optimal downsampling rate that shows the best performance. We have run the experiments using the 2x2 downsampling rate and provided those results in this thesis work. We have also experimented with the 4x4 downsampling rate, but it has shown less performance. A comprehensive study can be conducted on various downsampling rates. A downsampling rate versus performance chart can be generated to determine the optimal downsampling rate using which the proposed pipeline produces the highest visual quality and the best metric scores.

Last but not least, the low-resolution denoised diffuse and specular images can be used as additional guidances iteratively during each other's guided upsampling phase. More specifically, the diffuse image can be upsampled first by using the scene feature buffers. Then, during the specular image's upsampling phase, the upsampled diffuse image can be used as guidance along with the scene feature buffers. Lastly, the upsampled specular image can be used as guidance along with the scene feature buffers to upsample the diffuse image more accurately. The performance of this iterative approach can be examined in depth by conducting a detailed study. We leave this study as future work.



REFERENCES

- [1] J. T. Kajiya, “The rendering equation,” *SIGGRAPH Comput. Graph.*, vol. 20, p. 143–150, Aug. 1986.
- [2] S. Bako, T. Vogels, B. Mcwilliams, M. Meyer, J. Novák, A. Harvill, P. Sen, T. Deroose, and F. Rousselle, “Kernel-predicting convolutional networks for denoising monte carlo renderings,” *ACM Trans. Graph.*, vol. 36, July 2017.
- [3] K. He, J. Sun, and X. Tang, “Guided image filtering,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 6, pp. 1397–1409, 2013.
- [4] H. Wu, S. Zheng, J. Zhang, and K. Huang, “Fast end-to-end trainable guided filter,” in *CVPR*, 2018.
- [5] E. Veach and L. J. Guibas, “Optimally combining sampling techniques for monte carlo rendering,” in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 419–428, 1995.
- [6] E. P. Lafortune and Y. D. Willems, “Bi-directional path tracing,” 1993.
- [7] H. W. Jensen, “Global illumination using photon maps,” in *Eurographics workshop on Rendering techniques*, pp. 21–30, Springer, 1996.
- [8] E. Veach and L. J. Guibas, “Metropolis light transport,” in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 65–76, 1997.
- [9] A. Keller, “Instant radiosity,” in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 49–56, 1997.
- [10] T. Müller, M. Gross, and J. Novák, “Practical path guiding for efficient light-transport simulation,” in *Computer Graphics Forum*, vol. 36, pp. 91–100, Wiley Online Library, 2017.

- [11] S. Bako, M. Meyer, T. DeRose, and P. Sen, “Offline deep importance sampling for monte carlo path tracing,” in *Computer Graphics Forum*, vol. 38, pp. 527–542, Wiley Online Library, 2019.
- [12] F. Rousselle, M. Manzi, and M. Zwicker, “Robust denoising using feature and color information,” in *Computer Graphics Forum*, vol. 32, pp. 121–130, Wiley Online Library, 2013.
- [13] B. Bitterli, F. Rousselle, B. Moon, J. A. Iglesias-Gutián, D. Adler, K. Mitchell, W. Jarosz, and J. Novák, “Nonlinearly weighted first-order regression for denoising monte carlo renderings,” in *Computer Graphics Forum*, vol. 35, pp. 107–117, Wiley Online Library, 2016.
- [14] N. K. Kalantari, S. Bako, and P. Sen, “A machine learning approach for filtering monte carlo noise.,” *ACM Trans. Graph.*, vol. 34, no. 4, pp. 122–1, 2015.
- [15] M. Zwicker, W. Jarosz, J. Lehtinen, B. Moon, R. Ramamoorthi, F. Rousselle, P. Sen, C. Soler, and S.-E. Yoon, “Recent advances in adaptive sampling and reconstruction for monte carlo rendering,” *Computer Graphics Forum*, vol. 34, no. 2, pp. 667–681, 2015.
- [16] B. Xu, J. Zhang, R. Wang, K. Xu, Y.-L. Yang, C. Li, and R. Tang, “Adversarial monte carlo denoising with conditioned auxiliary feature,” *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2019)*, vol. 38, no. 6, pp. 224:1–224:12, 2019.
- [17] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele, “Joint bilateral up-sampling,” *ACM Transactions on Graphics (ToG)*, vol. 26, no. 3, pp. 96–es, 2007.
- [18] J. Chen, A. Adams, N. Wadhwa, and S. W. Hasinoff, “Bilateral guided upsampling,” *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, pp. 1–8, 2016.
- [19] Z. Wang, J. Chen, and S. C. Hoi, “Deep learning for image super-resolution: A survey,” *IEEE transactions on pattern analysis and machine intelligence*, 2020.
- [20] P. Bauszat, M. Eisemann, and M. Magnor, “Guided image filtering for interactive high-quality global illumination,” in *Computer Graphics Forum*, vol. 30, pp. 1361–1368, Wiley Online Library, 2011.

- [21] M. Boughida, T. Groueix, and T. Boubekeur, “Interactive monte-carlo ray-tracing upsampling,” in *Proceedings of the 37th Annual Conference of the European Association for Computer Graphics: Posters*, pp. 23–24, 2016.
- [22] H. Zimmer, F. Rousselle, W. Jakob, O. Wang, D. Adler, W. Jarosz, O. Sorkine-Hornung, and A. Sorkine-Hornung, “Path-space motion estimation and decomposition for robust animation filtering,” *Computer Graphics Forum*, vol. 34, no. 4, pp. 131–142, 2015.
- [23] B. Bitterli, “Rendering resources,” 2016. <https://benedikt-bitterli.me/resources/>.
- [24] R. Mantiuk, K. J. Kim, A. G. Rempel, and W. Heidrich, “Hdr-vdp-2: A calibrated visual metric for visibility and quality predictions in all luminance conditions,” *ACM Trans. Graph.*, vol. 30, July 2011.
- [25] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda, “Photographic tone reproduction for digital images,” in *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH ’02*, (New York, NY, USA), p. 267–276, Association for Computing Machinery, 2002.
- [26] swehrwein, “Python implementation of guided filter,” 2017. <https://github.com/swehrwein/python-guided-filter/>.