Noise-less Guarantees in Monte Carlo Path Tracing via Conformal Prediction Methods

Maya Gambhir

January 3, 2026

Abstract

1 Introduction

Monte Carlo path tracing (MCPT) is a powerful rendering technique that simulates realistic light transport in virtual environments. By tracing many light paths through a scene, MCPT can produce photorealistic images with global illumination, soft shadows, reflections, and other complex lighting effects. However, MCPT faces significant challenges in terms of convergence speed and noise reduction, especially at low sample counts. As rendering high-quality images often requires thousands of samples per pixel, there is a strong need for effective denoising techniques to produce clean results from noisy MCPT renders.

This paper explores a novel approach to MCPT denoising using conformal prediction methods. Conformal prediction is a statistical framework for constructing prediction sets with guaranteed coverage probabilities. By applying conformal prediction to MCPT sample data, we aim to develop an adaptive sampling and denoising technique that provides rigorous uncertainty quantification. Our key idea is to use partition learning conformal prediction (PLCP) to automatically identify informative subgroups in the image based on local complexity. This allows us to adaptively determine the required number of samples for each pixel to meet a specified noise threshold. The main contributions of this work are a formulation of MCPT denoising as a conformal prediction problem, allowing for principled uncertainty quantification and an adaptive sampling algorithm based on PLCP that learns to partition the image into regions of similar complexity.

This paper is organized as follows:

1. A review of Monte Carlo Path Tracing
2. A review of Conformal Prediction
3. A description of other tools used in the final algorithm
4. A description of the algorithm

2 Monte Carlo Path Tracing

The TLDR of Path Tracing

MCPT involves simulating the projection of many samples of light rays around a virtual environment to achieve a quality rendering, this can be sped up with deep learning techniques.

2.1 What is it?

The basics of MCPT are quite simple. MCPT is a rendering technique that aims to, among other things, simulate realistic light paths in virtual environments. The quantified definition of "Realistic Light" is grounded in the basics of physics, and represented by an equation discovered in 1986.

The Rendering Equation [1] denotes the *spectral radiance* of a wavelength λ directed outward along direction ω_o at time t, from a particular position x. It can be written as follows:

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int_\Omega f_r(x, \omega_i, \omega_o) L_i(x, \omega_i)(\omega_i \cdot n) d\omega_i$$

Where L_o is the outgoing radiance, L_e is the emitted radiance, f_r is the bidirectional reflectance distribution function (BRDF), L_i is the incoming radiance, ω_i and ω_o are the incoming and outgoing directions, n is the surface normal, and Ω is the hemisphere of incoming light directions.

While this equation can be solved, it is time and compute intensive, so MCPT is employed to approximate via sampling.

2.2 Paths & Path Tracing

What is a Path? A path is terminated by an eye E and a light L, and follows a series of *bounces*, or interactions with surfaces, that are either transmissions or reflections along with some notion of light scattering. MCPT follows an algorithm that involves repeated ray tracing through an initialized virtual environment.

The Perquisites: The path tracing algorithm requires some base knowledge of the virtual environment to execute properly. [2]

1. The **scene representation** includes the 3D geometric objects, surface normals, surface materials and light sources in the environment.
2. The **camera model** is a "pinhole camera" placed at some location in the scene.

The Path Tracing Algorithm [2] is executed by repeatedly simulating the shooting of light rays through the scene setup, redirecting the ray based on collisions with objects in the space, and finally averaging the contribution of all rays to each pixel to get the final image. We write a simplified version of the algorithm below.

Algorithm 1: Monte Carlo Path Tracing (Simplified)

Input: Scene setup S, camera C, number of samples per pixel N
Output: Final rendered image I
Start with a blank image I with the same size as the camera's view.
foreach *pixel in the image* **do**
    Set the pixel's color to zero.
    for *each sample out of N samples* **do**
        Create a random ray that passes through the pixel.
        Initialize the light collected for this ray to zero.
        Follow the ray through the scene:
        while *the ray hasn't left the scene* **do**
            Find where it hits something.
            if *there's no hit* **then**
                Add the background color and stop following the ray.
            Add light emitted and decide the ray's new direction based on material properties.
        Add this ray's contribution to the pixel's color.
    Average all the samples to get the final pixel color.
Return the completed image.

2.3 Utility

The value of such an algorithm comes from its ability to effectively simulate a variety of different lighting effects including, but not limited to global illumination, soft shadows, color bleeding, caustics, indirect lighting, diffuse and specular reflections, refractions. These are all lighting techniques that are difficult to recreate by hand in applications such as animation.

The utility of this algorithm applies in any case where accurate rendering with more effective compute resources is needed. This includes computer animation and visual effects rendering, MCPT is widely employed for creating photorealistic imagery in movies and animations [3]. There are also a variety of applications in game development, scientific visualization [3] and architectural visualization [4].

2.4 Challenges

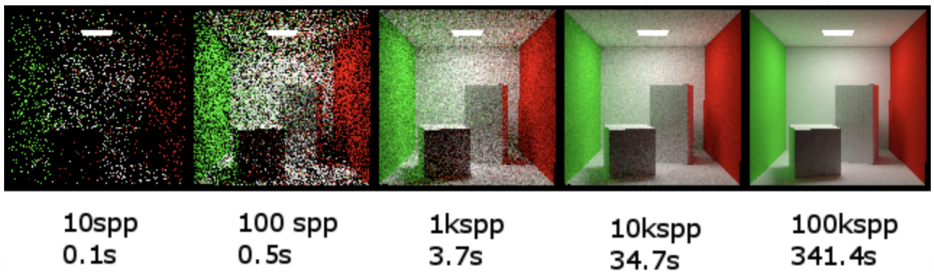

Figure 1: An example of the results of MCPT for a given number of spp(samples per pixel) and its corresponding compute time.

MCPT is, however, not without its challenges. Most notably, the algorithm faces problems of convergence, noisy outputs, and high compute requirements (although it is faster than the alternative of solving the rendering equation).

Convergence refers to the number of samples necessary to produce a useful and relatively noise free output. Note that variance decreases with $\frac{1}{\sqrt{N}}$ where N is the number of samples [5]. Since we are looking for a noise free image, low variance is a direct measure of output quality. With this, the more complex the lighting scenario, the more strict the acceptable bounds on variance become, as the user

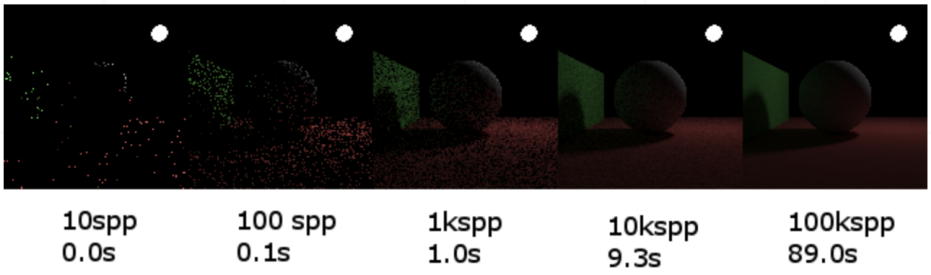

Figure 2: A second example of the results of MCPT for a given number of spp(samples per pixel) and its corresponding compute time.

desires a complex, but clean final output. Balancing convergence speed and render time involves trade-offs between image quality and computational efficiency. Faster renders may produce noisier results, while achieving noise-free images often requires significantly longer render times. The randomness in Monte Carlo integration leads to variance in light transport paths, resulting in grainy or speckled appearances in the rendered image. This effect is especially noticeable when scattered rays don't hit light sources after multiple bounces. Note the two examples given in 3 and 2, both take over a minute to converge to a reasonable output which could prove intractable with the introduction of many frames and more complex environments.

2.5 ML in MCPT

Given the difficulties listed above, a variety of machine learning techniques can be employed to improve the quality of outputs and the speed of convergence. Variations on CNNs take in a few noisy outputs and denoise them into a noise free final image/rendering, whether it be through reinforcement learning, auto-encoding or standard deep learning methodologies.

Supervised learning for denoising MCPT outputs A natural progression is to use employ deep neural networks, or CNNs to denoise the images. Rather than the traditional method of approximating the distribution via direct sampling and averaging to approximate the rendering equation, this method [6], takes a series of under-sampled images and uses a neural network to approximate the noise free version.

Reinforcement learning for Monte Carlo sampling An alternative method is to guide path exploration via reinforcement learning methods [7]. This method assumes each ray has eight actions at each bounce, and learns online to take the best action at each bounce. Such an implementation noted specific speedup in convergence to a noise free image.

Generative models for enhancing image quality Another proposed method is reconstructing noisy images via a recurrent auto-encoder [8]. This method considers a large pixel neighborhood in the convolutional steps and introduces fully convolutional recurrent blocks after every encoding stage. The auto-encoder aspect encodes relationships between pixels, and utilizes depth and normal information without user guidance.

3 Conformal Prediction

The TLDR of Conformal Prediction

Conformal prediction is a statistical prediction method that aims to produce an uncertainty set with marginal guarantees of correctness over the distribution of inputs.

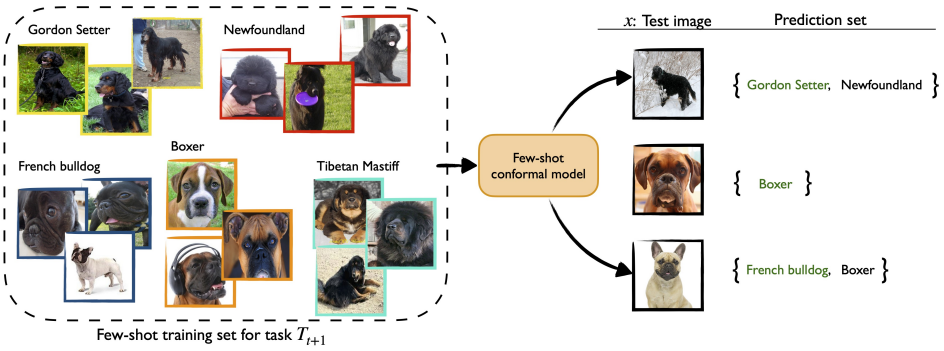

Figure 3: An example of how conformal prediction may do image classification via https://ar5iv.labs.arxiv.org/html/2102.08898

3.1 What is conformal prediction?

Perhaps more closely related to course material is work on the statistical technique known as conformal prediction. Note, the majority of this section is derived from the very thorough explanation of CP in [9]. The aim of the conformal prediction process is to generate prediction sets on the outputs of any given model that capture the correct value with some marginal probability guarantee. This is done by calibrating a threshold for some heuristic score computed on each potential output.

3.1.1 The Standard Conformal Prediction Algorithm

Given some set of input output pairs (x, y) and a trained model $\mathcal{M}$, steps 1-3 of the algorithm define the calibration portion of CP. We first define some heuristic association between the input x and label y. We then use this score, defined over all calibration examples, to calculate a quantile that will serve as a threshold for future examples. The algorithm is formally outlined below.

Algorithm 2: Conformal Prediction

1: **Input:**
   Pre-trained model $\mathcal{M}$ that predicts y given x
   Calibration dataset $\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$
   Desired confidence level $1 - \alpha$ ($\alpha \in [0, 1]$)
   New input X_{test} to make predictions on
2: Define a score function $s(x, y)$ such that higher scores indicate worse agreement between x and y.
3: Compute calibration scores for the given dataset:
   $s_i = s(X_i, Y_i)$ for all $i = 1, \ldots, n$
4: Compute the $(1 - \alpha)$ quantile $\hat{q}$:
   $\hat{q} = \text{Quantile}_{1-\alpha}(s_1, \ldots, s_n)$
   Use the empirical distribution: $\hat{q}$ is the $\lceil (n + 1)(1 - \alpha) \rceil / n$ quantile.
5: For the test input X_{test}, construct the prediction set:
   $C(X_{\text{test}}) = \{y : s(X_{\text{test}}, y) \leq \hat{q}\}$
   Interpretation: Include all y values for which the score function indicates sufficient agreement with X_{test} compared to the calibration threshold $\hat{q}$.
6: **Output:** Prediction set $C(X_{\text{test}})$ for the new test input

Correctness Guarantees Provided we meet the necessary preliminaries, which are outlined in the theoretical section below, we get the following guarantee for a chosen factuality level α

$$\mathbb{P}(Y_{\text{test}} \in C(X_{\text{test}})) \geq 1 - \alpha,$$

Predicting C on New Inputs We further outline the how to process a new input after calibration is concluded. Given a threshold q, a new input x and a set of possible outputs $y_1, ..., y_n$, we calculate $s(x, y_i) \forall i \in [1, n]$ and include in $C(X_{test})$ all y_i such that $s(x, y_i) < q$. Essentially, we threshold possible outputs by our nonconformity metric, rejecting outputs that are extremely non conformal given what we saw in calibration.

Heuristic Notions of Uncertainty There are a variety of uncertainty measures that can be defined on (x, y) pairs. These score functions also often make use of $\mathcal{M}(x)$ and its proximity to the true label y. Some simple methods include:

1. Residual-based: $s(x, y) = |y - \hat{y}|$, where $\hat{y} = \mathcal{M}(x)$
2. Likelihood-based: $s(x, y) = -\log P(y|x)$, if $\mathcal{M}$ is probabilistic

Note that while the quality of the score function does not infringe on our statistical guarantees of correctness it a poor heuristic *does* make for extremely large and relatively low-utility prediction sets.

3.2 Theoretical Analysis

One benefit of conformal prediction is that it relies on relatively weak distributional assumptions. The method only requires **exchangebility** of the distribution of inputs, which is notably weaker than the i.i.d. assumption. It also requires consistency of the scoring function, as use of the same threshold throughout. We outline the main theoretical guarantee and its associated proof below.

3.2.1 Proof of Correctness Guarantee

Statement: Given exchangeable calibration data $\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ and a test point $(X_{\text{test}}, Y_{\text{test}})$, the prediction set $C(X_{\text{test}})$ constructed using conformal prediction satisfies:

$$\mathbb{P}(Y_{\text{test}} \in C(X_{\text{test}})) \geq 1 - \alpha,$$

where $\alpha \in [0, 1]$ is the significance level.

Proof. Let the score function $s(X, Y)$ assign a real-valued score that reflects the agreement between X and Y. Assume we have n calibration examples $\{(X_i, Y_i)\}_{i=1}^n$ and a test example $(X_{\text{test}}, Y_{\text{test}})$, all drawn exchangeably from the same distribution.

1. **Exchangeable Scores**: Under exchangeability, the scores $s_1, s_2, \ldots, s_n, s_{\text{test}}$, where $s_i = s(X_i, Y_i)$ and $s_{\text{test}} = s(X_{\text{test}}, Y_{\text{test}})$, are also exchangeable.

2. **Quantile Definition**: The prediction set $C(X_{\text{test}})$ is defined as:

$$C(X_{\text{test}}) = \{y \in \mathcal{Y} : s(X_{\text{test}}, y) \leq \hat{q}\},$$

where $\hat{q}$ is the $\lceil (n + 1)(1 - \alpha) \rceil / n$ quantile of the calibration scores $s_1, \ldots, s_n$.

3. **Coverage Analysis**: To ensure $Y_{\text{test}} \in C(X_{\text{test}})$, we need:

$$s(X_{\text{test}}, Y_{\text{test}}) \leq \hat{q}.$$

Since the scores $s_1, s_2, \ldots, s_n, s_{\text{test}}$ are exchangeable, the rank of s_{test} among all $n+1$ scores is uniformly distributed:

$$\mathbb{P}(\text{rank}(s_{\text{test}}) \leq \lceil (n + 1)(1 - \alpha) \rceil) = 1 - \alpha.$$

4. **Key Insight**: By construction, the quantile $\hat{q}$ ensures that $(1-\alpha)$ proportion of the scores $s_1, \ldots, s_n$ are less than or equal to $\hat{q}$. Due to exchangeability, the same holds for s_{test}. Thus:

$$\mathbb{P}(s_{\text{test}} \leq \hat{q}) \geq 1 - \alpha.$$

5. **Final Coverage**: Since $s_{\text{test}} \leq \hat{q}$ implies $Y_{\text{test}} \in C(X_{\text{test}})$, we conclude:

$$\mathbb{P}(Y_{\text{test}} \in C(X_{\text{test}})) \geq 1 - \alpha.$$

□

4 Applying Conformal Prediction to MCPT

4.1 Motivation

An interesting question that may arise in the process of MCPT analysis of the samples is: d**o all pixels need the same number of samples?** It is clear that in a given rendering some areas of the image are much more complex than others, but classic sample collection techniques take the same number of samples for each pixel.

4.2 A Formal Setup

In an effort to introduce conformal techniques into the monte carlo denoising process, we first formalize our framework with Gharbi et al. [10]. Their methodology takes in an unordered set of MCPT samples and use a splatting approach along with a neural network to output a denoised final image.

This problem is cast as a *supervised learning problem*, with the input being a set of noisy examples for each pixel, with some contribution(to other pixel values) and additional features drawn from the entire image. The actual network architecture is not central to our methodology. In fact, the only thing we rely on is the per pixel analysis and computation on each pixel relative to the construction of the final image.

4.3 Setting up our Toolbox

4.3.1 Measures of Complexity

Three important complexity measures for assessing pixel neighborhoods in Monte Carlo path tracing are sample variance, perceptual variance, and gradient magnitude. These metrics provide valuable information about the local complexity and uncertainty in rendered images.

Sample variance is a fundamental measure that quantifies the spread of sample values within a pixel neighborhood. It was utilized in the Population Monte Carlo Path Tracing (PMC-PT) algorithm [11] to adaptively allocate more samples to high-variance regions. For a pixel with N samples and radiance values L_i, the sample variance is calculated as:

$$\text{Var}(L) = \frac{1}{N-1} \sum_{i=1}^{N} (L_i - \bar{L})^2$$

where $\bar{L}$ is the mean radiance. Higher variance indicates greater complexity and potential for noise, making it useful for guiding adaptive sampling and denoising efforts. This measure is especially useful when a scene has a variety of illumination levels or material properties. It also does well at identifying extremely bright or dark outliers that result from sampling noise.

Perceptual variance extends the concept of sample variance by incorporating human perception factors. The PMC-PT algorithm normalized sample variance using the threshold-versus-intensity (TVI) function to account for the non-linear relationship between physical light intensity and perceived brightness. This approach can be expressed as:

$$\text{PerceptualVar}(L) = \frac{\text{Var}(L)}{\text{TVI}(\bar{L})}$$

By adjusting for perceptual importance, this metric helps focus computational resources on areas where noise is more noticeable to viewers, potentially improving the perceived quality of the final image, performing well when we care more about the more visible noise in the image.

Gradient magnitude measures the rate of change in pixel values across the image, it is a common metric in image processing and computer vision. For a 2D image with pixel values $I(x, y)$, the gradient magnitude can be approximated using finite differences:

$$\text{GradMag}(x, y) = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$$

High gradient magnitudes often correspond to edges, textures, or other visually important features. In the context of Monte Carlo denoising, this metric can help identify regions that require more careful treatment to preserve important details and edges and avoid over-smoothing.

4.3.2 Multi-calibration

Multi-calibration is an extension of traditional calibration methods that aims to address bias and ensure fairness across multiple subgroups. One approach is to ensure the guarantees hold not only marginally on the entire distribution, but on any set of groups that inputs in the distribution may be a part of [12]. We can also expand on the exchangebility guarantee by dynamically adapting our threshold in a sequential learning setup [13]. In our case, we require calibration across varying sample sizes for a given pixel. If we did not calibrate on these groups, we would lose the exchangebility assumption required to attain guarantees, as we are no longer drawing at random from the set of inputs, but conditional on some current sample size.

4.3.3 Partition Learning Conformal Prediction

While group-wise guarantees are beneficial in conformal prediction, sometimes groups are not easy to define. If we aim to calibrate our sample amount to the complexity of the area surrounding the pixel, we need a way to turn complexity values into distinct features. Rather than choosing them arbitrarily we can employ Partition Learning Conformal Prediction or PLCP [14]. The method described in the paper constructs prediction sets for supervised learning tasks with enhanced conditional validity by learning uncertainty-guided features from calibration data. It dynamically partitions the covariate space into groups where prediction uncertainties are similar, improving upon traditional methods that assume predefined structures or rely solely on marginal guarantees. PLCP leverages machine learning models to learn these partitions, offering better conditional coverage, more precise prediction intervals, and adaptability to complex datasets. The framework is computationally efficient and can handle both regression and classification tasks while maintaining theoretical guarantees for both finite and infinite sample sizes. While PLCP can partition the covariate space arbitrarily, we limit it to partition on the complexity of a given pixel, allowing us to optimally use to complexity as an indicator of the needed threshold. Additionally, this ensures that we do not just decrease overall, but on both complex and simple areas of the image.

4.3.4 Imposing a Limit on Noise

Finally, to get guarantees of non-noisiness, we must impose an allowable limit of noise for the final image that we output. Our measure of this is dependent on if we have the ideal denoised as part of our calibration set.The ideal value to choose for a given situation is dependent on factors of the input distribution and should be determined by the needs of the individual user.

If we know the ideal image we can use a simple measure like mean squared error or structural dissimilarity [15]. If we do not know the ideal, we can refer again to the complexity measures above, which do a good job at indicating noise across an entire image as well as in a given area.

4.4 The Algorithm

Due to time constraints, I'll describe the algorithm in more informal terms, leaving the formal definition as an exercise for the reader. Just kidding, I'll do it eventually just not right now (its due in 5 hours).

Algorithm 3: Threshold Calibration and Path Sampling with Conformal Prediction

Input: Set of images $\{I_1, I_2, \ldots, I_n\}$ with Monte Carlo paths $\{p_{ij}\}$ for each pixel j in image I_i
Output: Threshold function $\tau(s, c)$ based on sample size s and complexity c
foreach *image* $I_i \in \{I_1, I_2, \ldots, I_n\}$ **do**
  foreach *pixel* j *in* I_i **do**
    Calculate group membership based on sample sizes s_j and on complexity c_j and PLCP.
    Calculate the final sample size s_j^* for pixel j.
    Calibrate the threshold for every possible sample size seen, $\tau_j^* = \tau(s_j^*, c_j)$ for pixel j.
foreach *pixel* j *in a new image* I_{new} **do**
  repeat
    Sample paths $\{p_j^{(t)}\}$ for pixel j.
    if $\hat{y}_j^{(t)} \geq \tau_j^*$ **then**
      Stop: Threshold τ_j^* exceeded, prediction is valid.
    if $t > s_j^*$ **then**
      Stop: Sample size t exceeds maximum threshold, no unique threshold.
  until
return Threshold function $\tau(s, c)$.

We also informally state that theoretical guarantees should follow from [14] as well.

5 Conclusion

In this paper, we presented a novel approach to Monte Carlo path tracing denoising using conformal prediction methods. By formulating denoising as a conformal prediction problem and leveraging partition learning techniques, we developed an adaptive sampling algorithm that can automatically identify regions of varying complexity in rendered images. This allows our method to efficiently allocate samples and provide rigorous uncertainty quantification.

While our method is an interesting idea, there are several avenues for future work. Clear future work is the formalization of the algorithm and the associated theoretical guarantees that stem from the other papers. Additionally, an implementation of this method would allow us to compare both runtime and quality of outputs to alternative methods, notably the other machine learning applications described in the section 2.

We can also include exploring more sophisticated partitioning strategies, incorporating additional image features beyond simple complexity measures, and extending the approach to handle animated sequences. Additionally, further investigation into the theoretical properties of PLCP in the context of image data could yield insights for improving both efficiency and accuracy. Overall, this work represents a step towards principled, adaptive denoising for Monte Carlo rendering. By bridging the gap between statistical learning theory and computer graphics, we can give some mathematical formalization to the sometimes more heuristic nature of image generatino and renderings.

References

[1] James T. Kajiya. "The Rendering Equation". In: *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '86)*. 1986, pp. 143–150. DOI: 10.1145/15922.15902.

[2] URL: https://graphics.stanford.edu/courses/cs348b-01/course29.hanrahan.pdf.

[3] Tianyu Huang et al. *Path Guiding for Monte Carlo PDE Solvers*. 2024. arXiv: 2410.18944 [cs.GR]. URL: https://arxiv.org/abs/2410.18944.

[4] Najda Villefranque et al. "A Path-Tracing Monte Carlo Library for 3-D Radiative Transfer in Highly Resolved Cloudy Atmospheres". In: *Journal of Advances in Modeling Earth Systems* 11.8 (2019), pp. 2449–2473. DOI: https://doi.org/10.1029/2018MS001602. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2018MS001602. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2018MS001602.

[5] URL: https://www.cs.rpi.edu/~cutler/classes/advancedgraphics/S08/lectures/17_monte_carlo.pdf.

[6] Xiwen Chen and Jianfei Shen. "Monte Carlo Noise Reduction Algorithm Based on Deep Neural Network in Efficient Indoor Scene Rendering System". In: *Advances in Multimedia* 2022.1 (2022), p. 9169772. DOI: https://doi.org/10.1155/2022/9169772. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1155/2022/9169772. URL: https://onlinelibrary.wiley.com/doi/abs/10.1155/2022/9169772.

[7] URL: https://github.com/repalash/CUDAPathTracerRL.

[8] Chakravarty R. Alla Chaitanya et al. "Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder". In: *ACM Trans. Graph.* 36.4 (July 2017). ISSN: 0730-0301. DOI: 10.1145/3072959.3073601. URL: https://doi.org/10.1145/3072959.3073601.

[9] Anastasios N. Angelopoulos and Stephen Bates. *A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification*. 2022. arXiv: 2107.07511 [cs.LG]. URL: https://arxiv.org/abs/2107.07511.

[10] Michaël Gharbi et al. "Sample-based Monte Carlo Denoising Using a Kernel-splatting Network". In: *ACM Trans. Graph.* 38.4 (2019), 125:1–125:12.

[11] Yu-Chi Lai and Charles Dyer. *Population Monte Carlo Path Tracing*. Tech. rep. University of Wisconsin-Madison Department of Computer Sciences, 2007.

[12] Christopher Jung et al. "Batch Multivalid Conformal Prediction". In: *International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=Dk7QQp8jHEo.

[13] Osbert Bastani et al. *Practical Adversarial Multivalid Conformal Prediction*. 2022. arXiv: 2206.01067 [cs.LG]. URL: https://arxiv.org/abs/2206.01067.

[14] Shayan Kiyani, George Pappas, and Hamed Hassani. *Conformal Prediction with Learned Features*. 2024. arXiv: 2404.17487 [cs.LG]. URL: https://arxiv.org/abs/2404.17487.

[15] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity". In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.