


# Real-Time Path-Guiding Based on Parametric Mixture Models

Mikhail Derevyannykh<sup>†</sup> 

Moscow State University, Eagle Dynamics



**Figure 1:** Frames of the Bistro scene (left) and the Sponza scene (right) with complex light scenarios are rendered with 1 spp using Path-Tracing (PT) and our Path-Guiding (PG) solution, and also with 2048 spp for reference. Notice the decrease in overall noise level for diffuse and specular surfaces, the ability of our algorithm to capture high-frequency indirect shadows just relying on 1 spp. Also, there is a slight performance boost. The images were rendered at 1080p resolution on a high-end desktop machine with RTX 2070 using the Falcor framework [NBW20].

## Abstract

Path-Guiding algorithms for sampling scattering directions can drastically decrease the variance of Monte Carlo estimators of Light Transport Equation, but their production usage was limited to offline rendering because of memory and computational limitations. We introduce a new robust screen-space technique that is based on online learning of parametric mixture models for guiding the real-time path-tracing algorithm. It requires storing of 8 parameters for every pixel, achieves a reduction of FLIP metric up to 4 times with 1 spp rendering. Also, it consumes less than 1.5ms on RTX 2070 for 1080p and reduces path-tracing timings by generating more coherent rays by about 5% on average. Moreover, it leads to significant bias reduction and a lower level of flickering of SVGF output.

## CCS Concepts

• Computing methodologies → Ray tracing;

## 1. Introduction

The main goal of path-tracing is to calculate the solution of the Light Transport Equation [Kaj86]:

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int_{H^+} L_i(x, \omega_i) f_r(x, \omega_i, \omega_o) \cos(\theta_i) d\omega_i \quad (1)$$

Here  $L_e(x, \omega_o)$  is the radiance emitted from a point  $x$  in a direction  $\omega_o$ ,  $L_i(x, \omega_i)$  is the incoming radiance from  $\omega_i$ ,  $f_r$  is the Bidirectional reflectance distribution function (BRDF) and  $\theta_i$  is the angle between the surface normal at  $x$  and  $\omega_i$ . Using, the Monte Carlo

simulation method for solving it [Vea98] leads to high variance. Usually, Importance Sampling techniques are used for sampling  $\omega_i$  direction to decrease the variance, but they only take into account the BRDF term of Eq. 1 and lead to unpleasant results in complex light cases. So, we need to guide the process of generating scattering direction based on the  $L_i(x, \omega_i) \cos(\theta_i)$  [VKv\*14] [MGN17] term, too, especially for real-time ray-tracing with a limited samples budget.

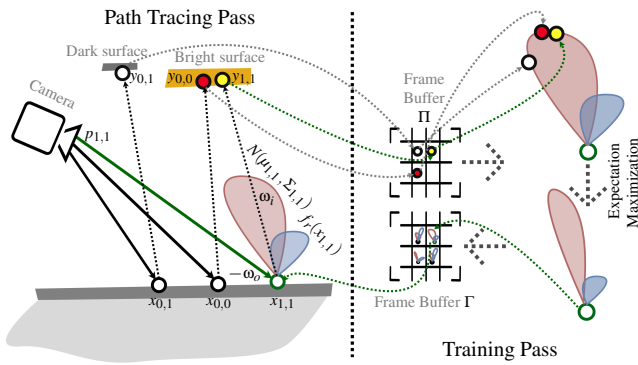
In this paper, we suggest storing parameters of parametric mixture models [VKv\*14] for every pixel in additional 2D textures  $\Gamma$  and using them for guiding. Mixture model  $D_{w,h}$  for a pixel  $p_{w,h}$  is based on the BRDF distribution of its surface  $x_{w,h}$  and one lobe 2D

<sup>†</sup> Chairman Eurographics Publications Board

Gaussian Distribution:

$$D_{w,h}(\omega_i) = \pi_{w,h} N(M^{-1}(\omega_i), \mu_{w,h}, \Sigma_{w,h}) + \pi_{w,h}^{-1} f_r(x_{w,h}, \omega_i, \omega_o) \quad (2)$$

where  $\mu_{w,h}$  is a mean,  $\Sigma_{w,h}$  is a covariance matrix,  $\pi_{w,h}$  is a mixing coefficient,  $\pi_{w,h}^{-1} = 1 - \pi_{w,h}$  and  $M$  is Shirley and Chiu area-preserving mapping [SC97] from a 2D unit square to a 3D unit hemisphere,  $N(x, \mu, \Sigma)$  is the probability density function of normal distribution. This technique ensures supporting arbitrary scenes, the ability to train models iteratively by re-projecting data from old frames, and robust adaptive Multiple Importance Sampling [VG95] for achieving results no worse than using default BRDF sampling. Our solution leads to a decrease in the overall level of noise (Figure 1) and does not impose any technical limitations on scenes materials, geometry and etc.



**Figure 2:** Firstly, we trace a ray from pixel  $p_{1,1}$  to scene with direction  $-\omega_o$  and get an intersection in  $x_{1,1}$  point respectively. Then algorithm loads parameters of mixture models from the frame buffer  $\Gamma(1, 1)$  for sampling first scattering direction  $\omega_i$ , traces a ray from  $x_{1,1}$  to  $\omega_i$ , and intersects some surface in  $y_{1,1}$  point. We save  $y_{1,1}$  and estimated  $L(x_{1,1}, \omega_i)$  as Virtual Point Light in the  $\Pi(1, 1)$  frame buffer. In the second training stage, we estimate the luminance of  $L_i(x_{1,1}, \omega_i) f_r(x_{1,1}, \omega_i, \omega_o) \cos(\theta_i)$  by accessing  $\Pi$ , use it and  $\omega_i$  for training mixture model of  $x_{1,1}$  by Expectation-Maximization (EM) algorithm in on-line fashion and save updated params in  $\Gamma(1, 1)$ . Also we use VPL of close pixels ( $p_{0,0}$ ,  $p_{0,1}$  in this example) for faster training.

## 2. Previous Work

The majority of previous works about path guiding were based on partitioning rendering scenes by some complex spatial structure on small cells and iteratively estimating spherical or hemispherical incoming radiance distribution for their internal surfaces by several approaches [MGN17] [VKv\*14] [DHD20] [RGH\*20]. The authors of [VKv\*14] rely on parametric mixture models for representing the incoming radiance, and they developed an algorithm for estimating model parameters in an online fashion with the support of weighted samples by radiance values. We use the same scheme with adaptive BRDF sampling and efficient implementation for real-time rendering based on screen space buffers for generating the first scattering direction [BMDS19]. Also, we suggest using Virtual Point Lights (VPL) of neighboring pixels for faster training [Kel97] as in the Restir GI algorithm [OLK\*21].

## 3. Path-Guiding Pipeline

We construct our solution based on the default path tracing algorithm with a subsequent training pass. It does not need any additional precomputations and complex volumetric data structures. Figure 2 illustrates the overall concept of our algorithm.

### 3.1. Path tracing pass

Algorithm inference default path tracing using rasterized G-Buffer, but with generating first scattering direction  $\omega_i$  based on the learned distribution of point  $x_0$ . Parameters for constructing this distribution are requested from the texture  $\Gamma$  with Nearest Neighbor filtering, based on current pixel coordinates  $w, h$  and motion vectors. For estimating mean and co-variance matrix in online fashion we only store and update  $E(X), E(Y), E(X^2), E(Y^2), E(XY)$  weighted by radiance's luminance moments of the sampled distribution, where  $[X, Y] = M^{-1}(\omega_i)$  and  $E$  is the expected value. In cases of invalid history (new visible geometry without trained distribution), we initialize new parametric models with  $\pi_{w,h} = 0.05$ ,  $k_{w,h} = 0$ ,  $k_{w,h}$  is a training epochs counter, and sample direction from BRDF.

For generating scattering direction, in case of sampling a uniform random variable  $\zeta < \pi_{w,h}$  we employ simulation of two  $N(0, 1)$  random variables by Box-Muller method and transforming them to  $N(\mu_{w,h}, \Sigma_{w,h})$  distribution with  $M$  mapping, in another we rely on the BRDF strategy. At the end of the pass, the algorithm also saves estimated  $L_i(x_{w,h}, \omega_i)$  and a position of next surface interaction  $y_{w,h}$  in additional texture  $\Pi$  for representing the VPL.

For achieving stable re-projecting mixture models parameters from an old frame to a new one we use some heuristical stopping weighting based on depth and normal differences, like in [SKW\*17] work. Also, it is important to take into account the difference between the hemispherical space of re-projected distribution and  $x_{w,h}$  surface space, and transform statistics. Exactly because of that reason we can not rely on hardware-accelerated bilinear filtering for sampling re-projected data.

### 3.2. Training pass

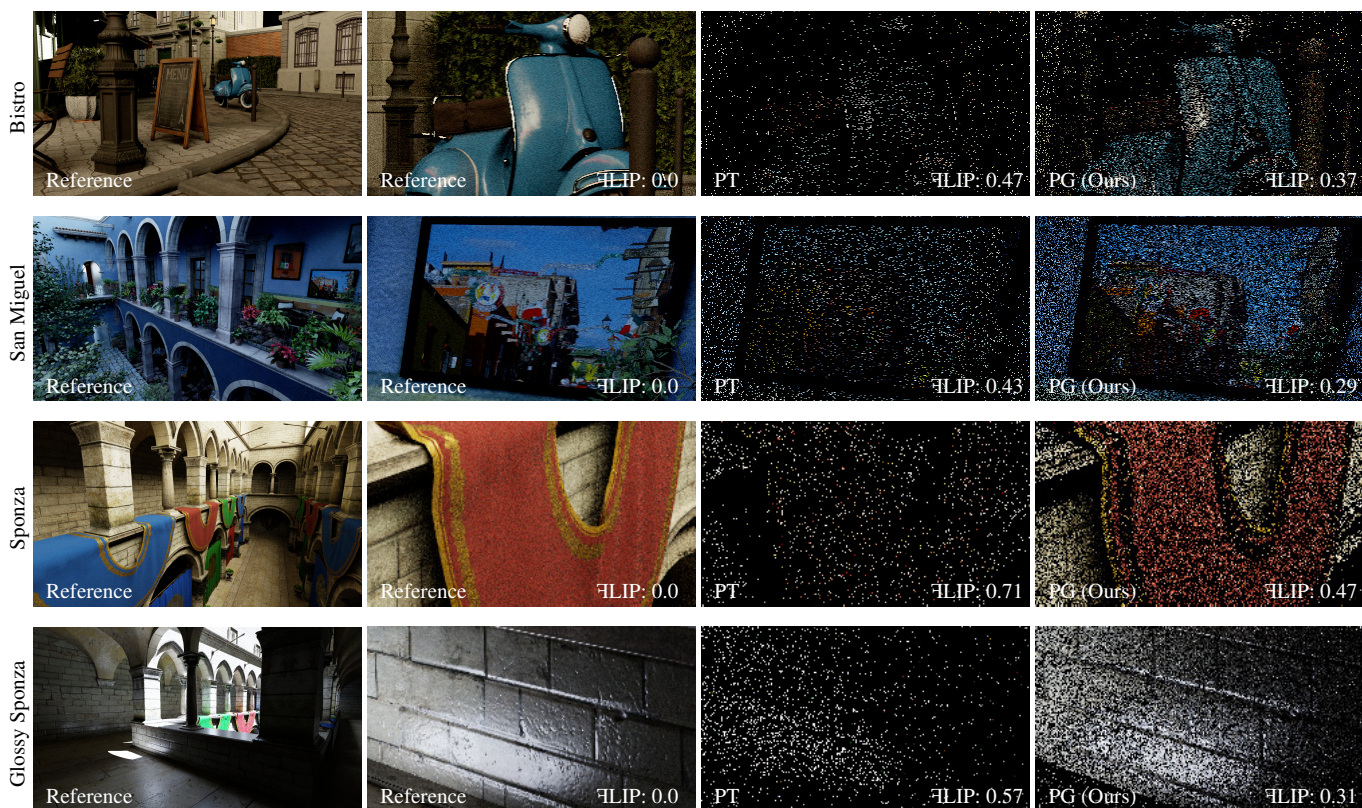
Expectation-Maximization (EM) algorithm iteratively trains parametric models to estimate distribution

$$D_{w,h}(\omega_i) \approx L_i(x, \omega_i) f_r(x, \omega_i, \omega_o) \cos(\theta_i) \quad (3)$$

for every pixel  $p_{w,h}$  based on incoming radiance  $L_i(x_{w,h}, \omega_i)$  from  $y_{w,h}$  and current fixed  $\omega_o$ . For running E-Step it is supposed to fetch material parameters from G-Buffer and estimate posterior probability  $p$  of sampling  $\omega_i$  from BRDF and from  $N([x, y], \mu_{w,h}, \Sigma_{w,h})$  distribution where  $[x, y] = M^{-1}(\omega_i)$ . In M-step algorithm calculates a new estimation of moments and other parameters employing temporal exponential smoothing with a smoothing factor:

$$\alpha_{w,h} = 0.7^{k_{w,h}} \quad (4)$$

Also, we suggest using  $N_{w,h}$  VPLs (see Eq. 5) from neighboring pixels inside a 5x5 pixels window for accelerating convergence, because it does not lead to any bias, where  $N_{w,h}$  is a number of sampled VPLs for a pixel  $p_{w,h}$ . For training distribution we only use samples generated by the BRDF strategy, because relying on all samples leads to  $\pi_{w,h} \approx 1.0$  and biased results in our experiments.



**Figure 3:** These comparisons demonstrate the effectiveness of our path-guiding algorithm for different scenes with zoomed render patches relying on 1 spp with trained mixtures. It decreases the variance of LTE (1) estimator for specular materials with normal maps (Bistro and Glossy Sponza), scenes with low-frequency environmental lighting (San Miguel), diffuse materials with high-frequency local indirect shadows, and complex sampling directional lighting (Sponza). Also, we show FLIP [ANAM\*20] metric error for numerical comparison.

## 4. Results and Discussion

We evaluate our new path-guiding algorithm on a set of different scenes: Sponza, Bistro Exterior, SanMiguel, Glossy Sponza based on the Falcor rendering framework [NBW20].

### 4.1. Comparison with Path-Tracing

We compare our solution only to default path-tracing because all other Path-Guiding techniques do not support real-time GPU-based rendering and require additional precomputations. As can be seen in Figure 3 our algorithm drastically decreases the level of noise for all testing scenes. It allows us to distinguish indirect local shadows, high-frequency specular reflections which are caused by employing normal maps, only by relying on 1 spp. Also, we want to notice that our technique decreases the variance of the MC estimator for low-frequency environmental lighting in the San Miguel scene, too.

Real-time rendering developers usually apply SVGF denoiser [SKW\*17] for achieving stable rendering results. We suggest employing the path-guiding technique, because, as can be seen in Figure 4, it drastically improves the robustness of the SVGF algorithm. Also, our technique decreases the level of flickering and recover lost indirect shadows and lighting. For a demonstration of this combination of techniques please watch the videos from supplementary

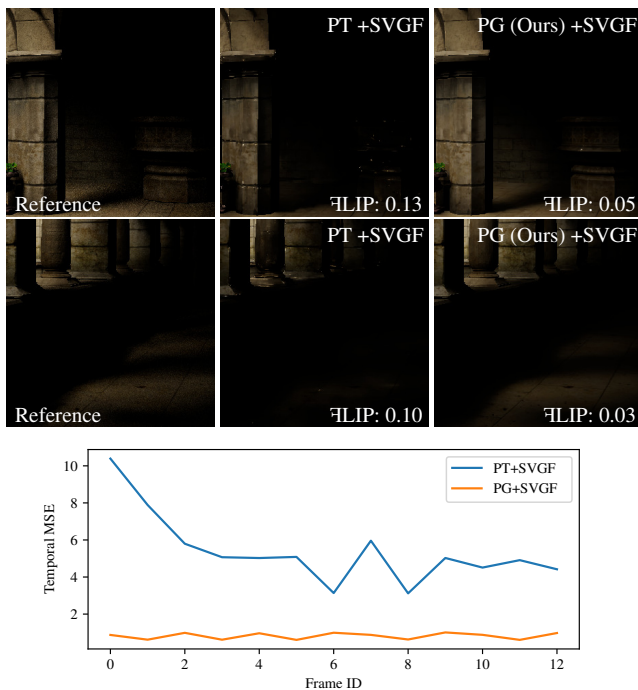
materials. But we rendered several frames of the static scene with disabled jittering of pixel centers using PT+SVGF and PG+SVGF, constructed plots with estimated temporal error for proving our statements about flickering level, see Figure 4.

### 4.2. Performance

Because we rely on the screen space technique the performance is fully dependent on the resolution of frame buffers. For improving it our algorithm preloads VPLs in shared memory buffers in the second training pass, which allows us to sample VPLs of neighboring pixels with better efficiency. The value of  $N_{w,h}$  influences the final performance and it introduces the trade-off between the speed of convergence of training and load on computing resources, for this reason, we propose to estimate it adaptively:

$$N_{w,h} = (1.0 - \min(k_{w,h}/kMax, 1.0)) * 15 + 5 \quad (5)$$

, where  $kMax = 10$  is the maximum number of training epochs. The final results of our performance measurements are about 1.5-1.75 ms per frame on RTX 2070 at 1080p resolution as additional overhead compares to the default path-tracing technique. But at the same time, path-guiding promotes the generation of more coherent rays, so it leads to an overall performance gain of about 5% compared to default path-tracing on average in our experiments.



**Figure 4:** As can be seen in this comparison for the Sponza scene, applying real-time SVGF can achieve robust results by using our path-guiding technique. Relying only on the denoiser with default path-tracing signal leads to temporal-flickering, fireflies, over-darkening the final renders, and loss of high-frequency indirect lighting or shadows. Also, we attach the plot for visualizing the level of flickering, which is captured by rendering several frames of the static scene.

## 5. Limitations

One of the main problems of this algorithm is slow training of the mixture model parameters in cases of incorrect reprojection between pixels which has very different incident lighting distribution. We discovered that it is not sufficient to cut pixels for re-projection only by normal and depth differences, so the optimal strategy is an open question for future research.

Also, using the screen space technique allows us to guide the sampling process only for first surface interactions. Moreover, it leads to unstable results for new visible surfaces, but we want to notice that convergence in such cases is very rapid.

Because of supporting guiding in the hemispherical domain, we rely only on the BRDF distribution sampling for transparent materials and also for materials with roughness near zero.

In contrast to the work of [VKv\*14], we only train one Gaussian lobe per pixel and it may lead to worse results. This decision is motivated by the ease of the implementation and limitations of real-time path-tracers in terms of memory bandwidth (every additional lobe requires storing 6 parameters) and computational loads. Also, because parameters of a Gaussian mixture model make a set, which is order-independent, there would be a filtering problem.

## 6. Conclusion

In this paper, we have presented the novel real-time technique for path-guiding which decreases the overall noise level of rendering with a limited samples budget for different types of scenes and light conditions, and slightly improves the overall performance. Also, we discovered that our algorithm has additional benefits of combining it with SVGF, which leads to decreasing the bias of the final render and improving its temporal stability. We believe that our work will inspire the research community to explore new ways of applying path-guiding techniques for real-time ray tracing.

## References

- [ANAM\*20] ANDERSSON P., NILSSON J., AKENINE-MÖLLER T., OSKARSSON M., ÅSTRÖM K., FAIRCHILD M. D.: Flip: A difference evaluator for alternating images. *Proc. ACM Comput. Graph. Interact. Tech.* 3, 2 (aug 2020). 3
- [BMDS19] BAKO S., MEYER M., DEROSE T., SEN P.: Offline deep importance sampling for monte carlo path tracing. *Computer Graphics Forum* 38 (10 2019), 527–542. 2
- [DHD20] DITTEBRANDT A., HANIKA J., DACHSBACHER C.: Temporal Sample Reuse for Next Event Estimation and Path Guiding for Real-Time Path Tracing. In *Eurographics Symposium on Rendering - DL-only Track* (2020), Dachsbacher C., Pharr M., (Eds.), The Eurographics Association. 2
- [Kaj86] KAJIYA J. T.: The rendering equation. *SIGGRAPH Comput. Graph.* 20, 4 (aug 1986), 143–150. 1
- [Kel97] KELLER A.: Instant radiosity. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (USA, 1997), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., p. 49–56. 2
- [MGN17] MÜLLER T., GROSS M., NOVÁK J.: Practical path guiding for efficient light-transport simulation. *Comput. Graph. Forum* 36, 4 (jul 2017), 91–100. 1, 2
- [NBW20] NIR BENTY KAI-HWA YAO P. C. L. C. S. K. T. F. M. O. C. L., WYMAN C.: The falcor rendering framework, 2020. 1, 3
- [OLK\*21] OUYANG Y., LIU S., KETTUNEN M., PHARR M., PANTALEONI J.: Restir gi: Path resampling for real-time path tracing. *Computer Graphics Forum* 40 (12 2021), 17–29. 2
- [RGH\*20] RATH A., GRITTMANN P., HERHOLZ S., VÉVODA P., SLUSALLEK P., KŘIVÁNEK J.: Variance-aware path guiding. *ACM Trans. Graph.* 39, 4 (jul 2020). 2
- [SC97] SHIRLEY P., CHIU K.: A low distortion map between disk and square. *J. Graph. Tools* 2, 3 (dec 1997), 45–52. 2
- [SKW\*17] SCHIED C., KAPLANYAN A., WYMAN C., PATNEY A., CHAITANYA C. R. A., BURGESS J., LIU S., DACHSBACHER C., LEFOHN A., SALVI M.: Spatiotemporal variance-guided filtering: Real-time reconstruction for path-traced global illumination. In *Proceedings of High Performance Graphics* (New York, NY, USA, 2017), HPG '17, Association for Computing Machinery. 2, 3
- [Vea98] VEACH E.: *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford, CA, USA, 1998. AAI9837162. 1
- [VG95] VEACH E., GUIBAS L. J.: Optimally combining sampling techniques for monte carlo rendering. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1995), SIGGRAPH '95, Association for Computing Machinery, p. 419–428. 2
- [VKv\*14] VORBA J., KARLÍK O., ŠIK M., RITSCHER T., KŘIVÁNEK J.: On-line learning of parametric mixture models for light transport simulation. *ACM Trans. Graph.* 33, 4 (jul 2014). 1, 2, 4