EIC-Opticks and its application to the pfRICH detector at the Electron Ion Collider

Gabor Galgoczi*[1], Kolja Kauder[1], Maxim Potekhin[1], Sakib Rahman[1], Dmitri Smirnov[1], and Torre Wenaus[1]

[1] *Brookhaven National Laboratory, Upton, NY 11973, USA*

February 2025

Abstract

The lion's share of time spent in the simulation of Cherenkov and other scintillation detectors is spent on optical-photon transport, i.e. *ray tracing*, a task that GPUs are uniquely qualified to perform. We present EIC-Opticks, a fork of Opticks [1], which uses event aggregation to drastically accelerate photon transport simulation for low-to-moderate photon yield experiments. During the full Geant4 Monte Carlo simulation of a given detector, optical photon simulation is performed on GPU(s) using the NVIDIA OptiX framework.

We validate this approach using the ePIC pfRICH [2] detector. We find GPU and CPU simulations in excellent agreement. We characterize numerical tolerances relevant to GPU ray tracing and show that they are negligible compared to uncertainties from optical-physics modeling. For 5×10^4 electrons with a momentum of $p = 5$ MeV/c in the test case of the pfRICH detector, EIC-Opticks shows an order-of-magnitude speedup over multi-threaded Geant4, and a factor of up to 148±9 over single-threaded execution. Event aggregation further reduces the per-photon simulation time from $\sim 60\,\mu$s for single events to ~ 20 ns with batching, a factor of ~ 3000.

In order to make EIC-Opticks easily installable, we authored a Spack package that makes it possible to install it with a single command. Additionally, a Docker container[1] is provided for users with EIC-Opticks installed. EIC-Opticks provides guardrails for common pitfalls (e.g. nested volume conversion, ray tracing setting optimization).

*Corresponding author: ggalgoczi1@bnl.gov

[1] https://github.com/BNLNPPS/esi-shell

1 Introduction

Optical photon simulation is a critical, yet extremely time-consuming part of scintillator and Cherenkov photon detector simulations in both high-energy (HEP) and nuclear physics (NP). In these fields, the most widely used simulation tool is Geant4 [3, 4], which executes optical processes on the CPU. To give an example for the JUNO detector, 99% of simulation time was spent solely on the simulation of optical photons [1]. This limits the size of simulated samples, slows design iterations, and complicates end-to-end studies that must be repeated across many geometries. Furthermore, it limits the size of output data that can be used for Machine Learning applications.

At the same time, specialized Graphics Processing Units (GPUs) have gained massive popularity in entertainment and professional applications. Billions of dollars have been spent since the start of the millennium with no sign of slowing down. The key technologies are massive vectorization and since ca. 2018 dedicated ray tracing hardware. However, algorithms natively available for GPUs do not support processes such as Rayleigh scattering. The Opticks software [1, 5] was built Nvidia's ray-tracing engine OptiX [6]. Opticks also incorporates pertinant physics like Mie scattering, Rayleigh scattering, etc. Monte Carlo methods are used to introduce the randomness needed in HEP and NP simulations. Opticks was originally developed for the JUNO experiment, where single events generate hundreds of millions of optical photons. These photons occupy the entire RAM of the GPU. Thus, overhead is minimal. However, most high-energy and nuclear physics experiments involving optical photons yield only a few hundred photons per individual event. Therefore, naïve per-event offloading can be slower than CPU-only simulation because the fixed overhead per offload dominates the simulation time.

To address this issue, we introduce EIC-Opticks[2], a fork and extension of Opticks. Although developed mainly for Electron Ion Collider (EIC) [7] detector studies, it lends itself readily to many low-to moderate photon-yield applications, including medical imaging simulations. EIC-Opticks adds an event aggregation mode that batches thousands of Geant4 events into a single GPU call. This design retains physics fidelity while removing per-event overheads that otherwise erase GPU advantages at low photon counts.

The framework supports the standard optical processes required for Cherenkov detectors (e.g. wavelength-dependent refractive indices and absorption, Rayleigh scattering, and configurable surface reflectance/roughness models), and it integrates with common HEP/NP workflows. For usability and reproducibility, we provide containerized distributions and a Spack [8] package, enabling turnkey deployment on hosts with modern NVIDIA GPUs.

We validated EIC-Opticks using the *pfRICH* detector [9] as a representative EIC subsystem. In a study with 5×10^4 electrons at $p = 5$ MeV/c, we compare observables between GEANT4 and EIC-Opticks and find agreement. In the same configuration, batching reduces runtime dramatically: relative to single-

[2] https://github.com/BNLNPPS/eic-opticks

threaded GEANT4 we observe speedups approaching three orders of magnitude in the batched case, while outperforming multi-threaded GEANT4 by a factor of ten for typical pfRICH workloads. These gains make previously prohibitive optical studies practical at scale and open the door to systematic scans over geometry, surface treatments, and operating conditions, not to mention the magnitude of order more training data for Machine Learning applications.

The paper is structured as follows: Section 2 describes EIC-Opticks and the batching strategy. Section 3 summarizes the pfRICH setup used in this work. Section 4 presents physics validation against GEANT4. Section 5 reports performance benchmarks and configuration studies. Section 6 discusses precision limits and mitigation strategies. We conclude with a summary.

2 EIC-Opticks

Opticks is a GPU-accelerated framework that couples to GEANT4 and offloads optical-photon transport to NVIDIA's OPTIX ray-tracing runtime while leaving all non-optical physics on the CPU. Building on this model, EIC-Opticks retains the hybrid execution workflow and introduces an event–aggregation mode that batches thousands of low yield GEANT4 events into a single GPU simulation. This removes per-event launch and transfer overheads without changing the underlying physics and makes configurations such as pfRICH efficient even when individual events produce only $\mathcal{O}(10^2$–$10^4)$ photons per event.

To protect users from crashes due to failed geometry conversions, EIC-Opticks analyzes constructive solid geometry (CSG) upon conversion from the input GDML [10] files and emits warnings when it detects deeply nested boolean operations (unions, subtractions, intersections). Since CSG is represented as full binary trees in the accelerator architecture (GPU representation of the geometry), such nested boolean operations can yield a binary tree size of millions of leafs. In such cases, we suggest modifications to the volume in question.

For usability and reproducibility, we migrated the most important runtime controls from shell environment variables into a versioned configuration file. Parameters include the maximum number of boundary reflections, and the "propagation epsilon" [11, 12], probably the most important user-defined setting in the NVIDIA OptiX framework (see Section 6 for further details).

EIC-Opticks is distributed as a container image for zero-install execution and provides a SPACK package for managed builds, ensuring consistent deployment across developer workstations and computer clusters. As an additional option EIC-Opticks can be installed with the following single command:

```
spack install eic-opticks ^optix-dev@7.9.0
```

We also provide geometry specific tools to optimize crucial settings. First, the maximum number of reflections controls when a photon is discarded. If the setting is too large, a small portion of photons that do not contribute significantly to the hit numbers use up excessive GPU time. If it is too small, a significant number of relevant hits may be missing. To optimize this number,

simulations can be automatically run with several settings until the number of hits converges. This analysis is needed for each geometry, since the expected mean path and detector topology varies case by case. In Section 3 we illustrate this trade-off specifically for the pfRICH geometry.

Secondly, the ray-tracing "epsilon" is a parameter of the underlying NVIDIA OptiX simulation. It defines by how much each step is shifted on purpose to avoid self intersection. If epsilon is too large, then photons may teleport through a boundary surface. If it is too small, a photon may get very close to a boundary and again teleport through, this time due to rounding errors. We will demonstrate details of this trade-off again in the context of the pfRICH in Subsection 6.

3 pfRICH detector at the Electron Ion Collider

The proximity-focusing Ring-Imaging Cherenkov Detector (pfRICH) is currently being constructed as part of the Electron-Proton/Ion Collider (ePIC) [9] detector system slated to start begin operations in the mid-2030s Designed to provide pion-kaon separation up to 7 GeV/c [13], the pfRICH detector subsystem consists of a thin radiator layer, a 40 cm proximity gap, a photosensor plane, and two mirrors to increase acceptance. The optical design prioritizes compactness and low material budget while maintaining the high pion-kaon separation, which makes it ideal to stress-test the EIC-Opticks software.

The collaboration's original GDML file defining the pfRICH detector consists of 126 volumes. For simulation studies in this work, we use a minimally simplified geometry that preserves all optically relevant elements: aerogel radiators (with wavelength-dependent refractive index and Rayleigh scattering), inner/outer mirror assemblies with defined optical surfaces, the photosensor window and detection planes. Volumes that are irrelevant to photon transport or that rely on Geant4 solids currently unsupported (trapezoids) in our analytic GPU geometry description are substituted with optically similar constructs that match surface normals and refractive/reflective interfaces.

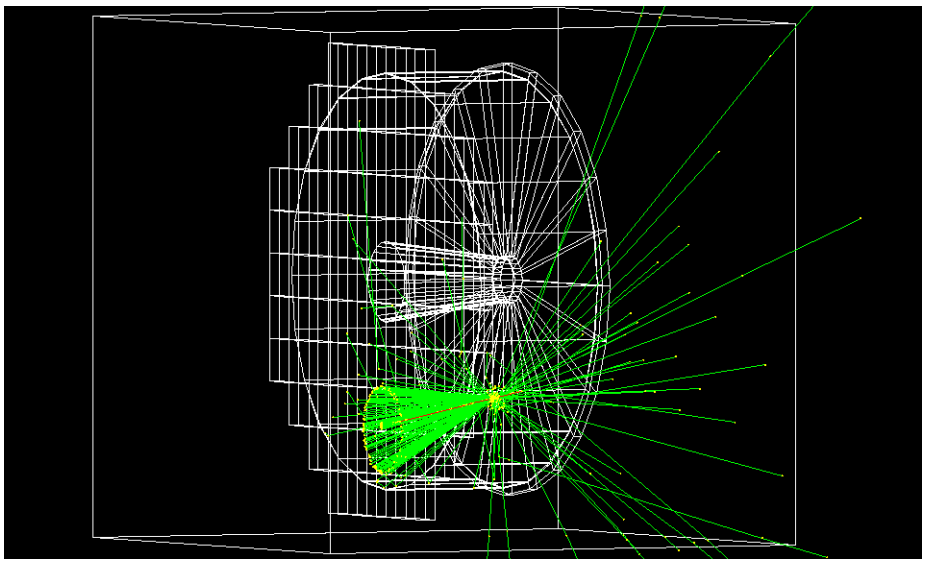

Figure 1: The Geant4 simulation of a single 5 MeV electron (red track) crossing the pfRICH detector, creating Cherenkov photons (green tracks) mainly in the aerogel tiles.

4 Validation

In order to validate EIC-Opticks we have used the exact same geometry both in a standalone Geant4 simulation (version 11.2.1) and in a simulation where the optical photons were offloaded to EIC-Opticks to be simulated on GPU. We used the GDML file of the pfRICH geometry, with the only modification being the approximated trapezoid volumes mentioned above. Note that for the purposes of this comparison, the standalone Geant4 simulation uses the same geometry as EIC-Opticks. The number of maximum reflections was set to 24 in EIC-Opticks; this choice is explained below in Section 5.

50,000 electrons with a momentum of 5 MeV/c were simulated in individual events. In the case where all particles were simulated on Geant4 (CPU) 13,840,638 ± 3720 hits were recorded. When optical photons were offloaded onto the GPU to be simulated by EIC-Opticks 13,840,566 ± 3720 hits were recorded.

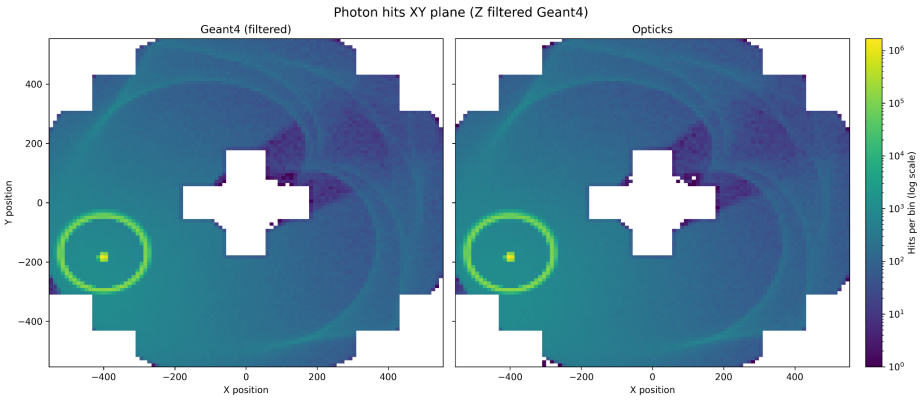

Figure 2: 50,000 electrons with momentum 5 MeV/c shot in as a pencil beam. Hit maps simulated by Geant4 (CPU) validation on the left and EIC-Opticks (GPU) on the right. Event batching introduced in EIC-Opticks was utilized. The identical (within statistical errors) hit maps prove that EIC-Opticks yields the same results as Geant4.

In Fig. 2 the hit histograms of generated photons are shown, the left panel was generated by regular Geant4 on a CPU, and the right one with EIC-Opticks. One can see that the histograms are identical, and the Cherenkov ring and caustic curves generated by the reflection of the optical photons on the outer mirror are visible in both.

5 Performance

In order to compare the simulation of EIC-Opticks to Geant4 optical photon propagation, 50,000 electrons, each with a momentum of 5 MeV/c were simu-

lated yielding 44 million optical photons. The time required to simulate the photons in Geant4 was measured as the time difference between the full simulation run with and without optical photon propagation. In the case of EIC-Opticks, we included all the overhead: copying the would-be simulated data to GPU, ray-tracing, and copying the results back to memory. In Fig. 3 the speedup using EIC-Opticks is plotted with respect to a multi-threaded Geant4 optical photon propagation. The GPU simulation is 148±9 times faster compared to a single thread of Geant4 and 9.3±0.5 times faster than a multi-threaded Geant4 simulation with 20 threads. The GPU utilized was an NVIDIA GeForce RTX 4090, and the CPU was an Intel Xeon w7-3445.

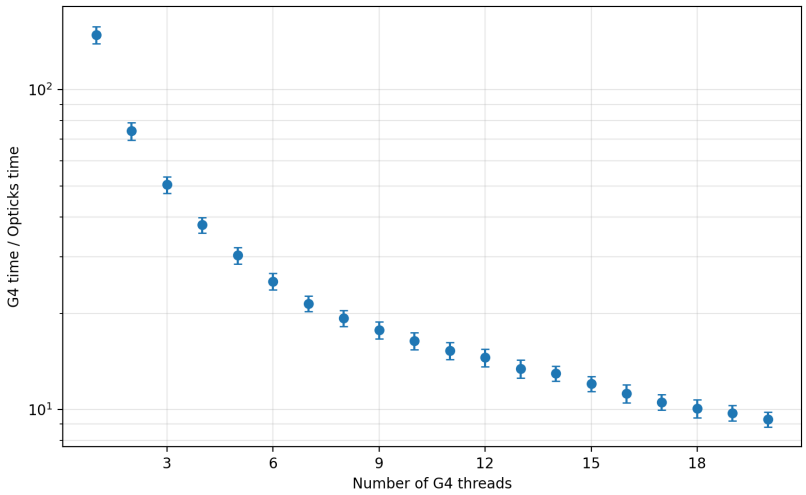

Figure 3: Speed increase of the GPU simulation compared to propagating the photons with multi-threaded Geant4 (CPU) for different number of Geant4 threads.

The pfRICH's photon occupancy is relatively small, meaning that overhead plays a significant role on a per-event basis, and furthermore that the GPU is not used to its full potential. We can achieve an additional dramatic speed-up by batching multiple Geant4 events into a single GPU launch. In the left panel of Fig. 4, the average time required to propagate a single photon is plotted for different numbers of Geant4 events batched together on the GPU. When a single event with ∼100 photons is simulated, it takes ∼60 μs on average to propagate one photon. Meanwhile, a batch of 10,000 events reduces this number to ∼ 20 ns. On the right of Fig. 4, the speedup is shown for different numbers of events batched together compared to no batching. With the GPU's memory thus fully utilized, a speedup of ∼ 3000 is reached.

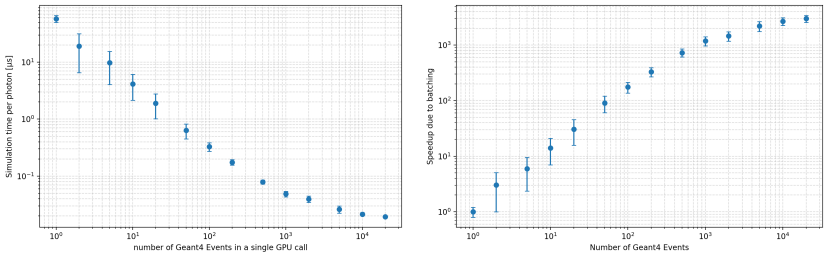

Figure 4: Left: Average time spent on the simulation time for one photon with EIC-Opticks with different event batch sizes in a single GPU call. Right: Speed increase of the GPU simulation due to batching compared to no batching. Simulating 10,000 events in a single GPU call yields a speedup of three orders of magnitude in the case of the pfRICH detector.

The maximum number of reflections on surfaces is one of the most important settings in EIC-Opticks, described in Section 6. We simulated 50,000 electrons with momentum of 5 MeV/c with maximum reflection settings from 5 up to 32 in steps of 1. In Fig. 5 the number of hits can be seen for different reflection settings. The number of hits starts to plateau around $\sim$24 reflections.

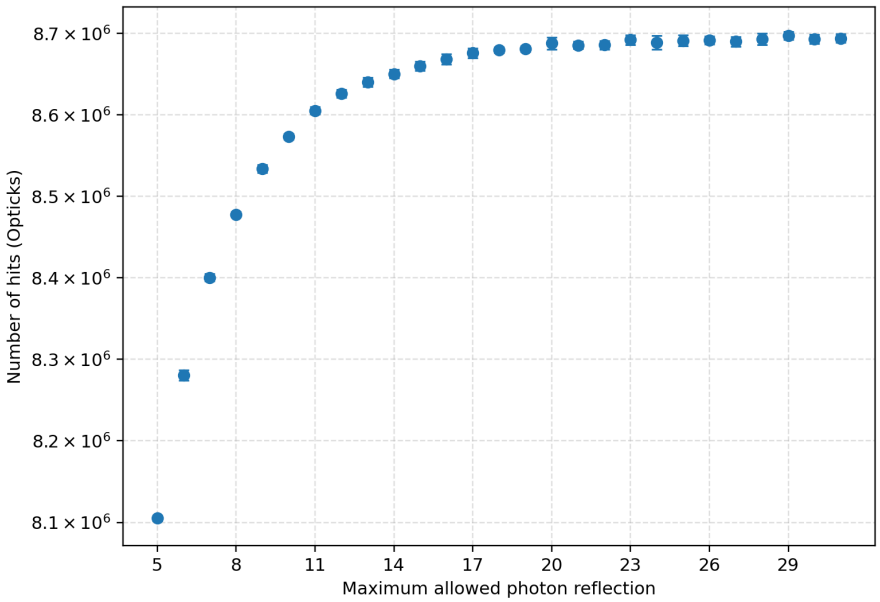

Figure 5: Number of hits in the pfRICH geometry upon simulating 50,000 electrons 5 MeV/c with maximum reflection settings from 5 up to 32. One can see that the number of hits plateaus around ≈24, in which case the number agrees with the baseline CPU Geant4 simulation within a statistical error.

In order to properly measure the run time and see whether there is a significant difference for specific settings, 20 simulations each were run for maximum reflection values of 8, 16, 24, and 32. Table 1 shows the simulation time for each setting. The time variation for different settings is on the order of 10%. However, for specific use cases, e.g. elongated scintillator bars where photons can be reflected hundreds of times, warp—GPU core group that shares instructions—divergence still has to be investigated.

Table 1: GPU simulation time required to simulate 50,000 electrons with a momentum of 5 MeV/c for multiple maximum reflection settings

Maximum Reflections	**Time [s]**
8	0.737 ± 0.008
16	0.83 ± 0.01
24	0.86 ± 0.01
32	0.86 ± 0.01

6 Photon leakage

EIC-Opticks uses ray-tracing provided by the underlying NVIDIA OptiX framework. When the track of a photon crosses a boundary, naïvely the next step should start at the boundary. However, OptiX utilizes single precision numbers, and when calculating the origin of the next step, there is a chance that it will lie still outside the volume. Thus, instead of being reflected on the surface in the second step, the ray will intersect the boundary from the outer side and thus escape from the volume, in other words, "leak" out. This happens due to the reflection process that still changes the direction of the ray.

In order to prevent photons from this leakage (or teleport), a small offset is added for each step call along its current direction. As shown in Fig. 6, even though the origin of the second step is incorrectly in the other volume, the offset pushes it back.

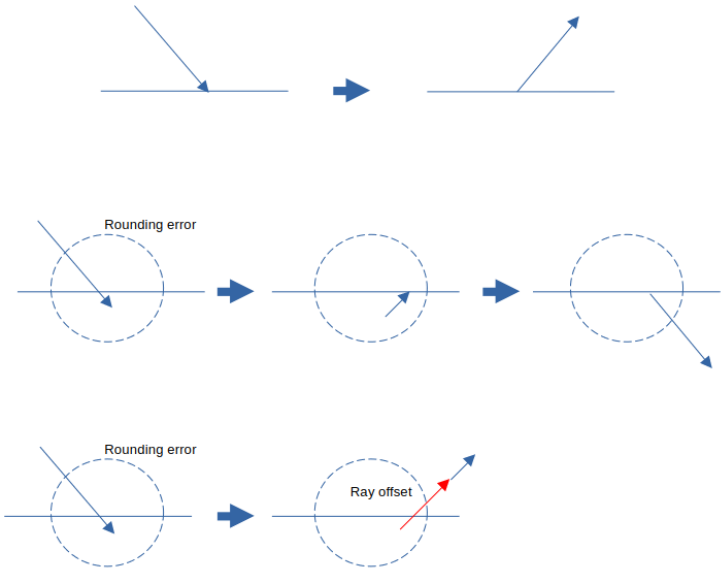

Figure 6: A typical reflection from a surface: first step ending on the surface and the second step originating from the surface (top). In ray tracing however (middle), rounding errors can cause the second step to start outside its originating volume and thus being scattered from the outer side of the surface, effectively teleporting through a surface. In OptiX, a ray offset is applied to each ray which pushes such escapees (and all rays) back into the volume (bottom).

Since EIC-Opticks utilizes the Monte Carlo method to include bulk physics, setting a proper ray offset becomes vital due to the following: in each step the OptiX ray tracing function is called, which returns the surface that photon

intersects. Afterwards the Monte Carlo method is utilized to decide which physical process happens to the photon. For example, it can bulk scatter instead of being propagated to the next surface. Yan and his colleagues reported the same phenomenon [14].

In the latter specific case, if the scattering occurs very close to a boundary, the ray offset can push the origin of the next step outside from the volume, as shown in Fig. 7. Thus, there is an optimal value for the ray offset, if it is too small the photon can leak due to rounding error, while if it is too large it can teleport due to bulk scattering.

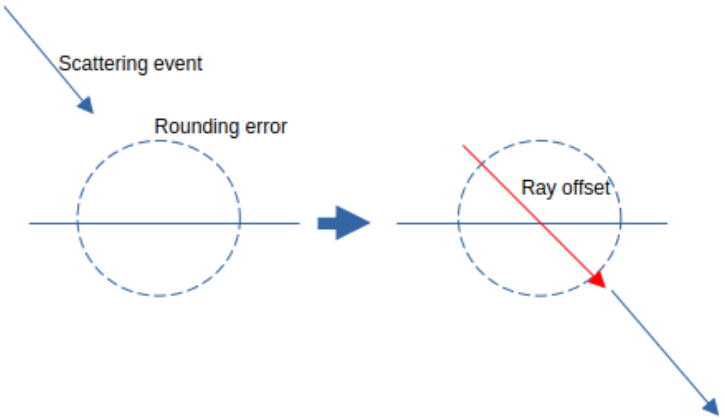

Figure 7: A ray that is being bulk scattered close to the surface (left) is pushed out from the volume by the ray offset. If the direction of the ray had changed, as in a typical surface reflection, this would not be an issue, since the origin of the next step is moved along the direction of the ray.

In order to quantify photon leakage in the case of EIC-Opticks, we created a simple geometry, consisting of two concentric spheres with radius 15 mm and 25 mm, respectively. The inner sphere is perfectly reflective on the inside, the outer one detects photons that escape from the inner one. We simulated 10^8 photons originating from the shared center. The maximum number of reflections were set to 32; the scattering length was 1 m. When setting the ray offset to the default value of 50 μm, we observed $\sim$ 30 leaks per 10^6 photons (30 ppm). In Fig. 8, the fraction of escaped photons when varying the ray offset setting is shown. One can read off an optimal setting for this specific geometry around 50 nm, for which leakage drops by a factor of 2000 to $\sim$0.015 ppm. Since leakage is geometry dependent, the optimal ray offset will vary. EIC-Opticks provides automated scan utilities so that users can determine and validate the best settings for their specific detector geometry.

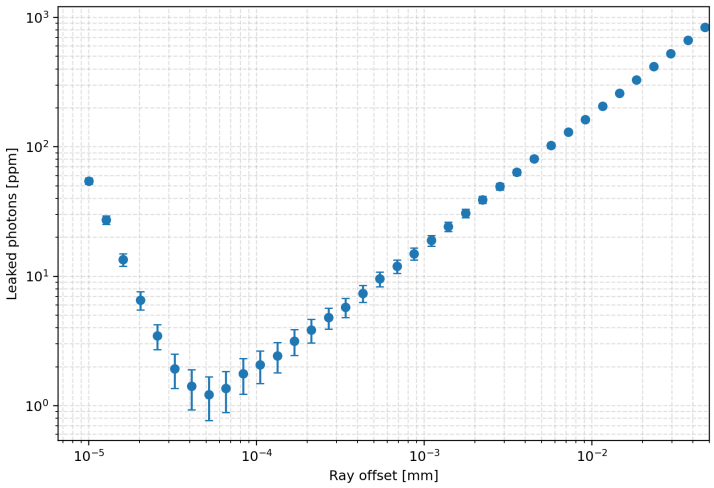

Figure 8: Fraction of optical photons leaking through a perfectly reflective spherical surface for different ray offsets. The reason for the increase at low offsets is rounding errors, while for large offsets it is caused by bulk scattered events. The optimal ray offset depends on the geometry.

7 Summary

We introduced EIC-Opticks, a GPU backend for optical-photon transport simulation that extends the Opticks framework to low- and moderate-yield detectors. That was achieved by aggregating photons from thousands of Geant4 events into a single GPU launch. Additionally, we introduced guardrails (e.g. CSG depth checks, scene-epsilon scans) in order to automatize the choice of geometry specific settings. With these tools, users can safely import complex geometries and minimize photon leakage.

We validated EIC-Opticks by simulating electrons entering the ePIC pfRICH detector using Geant4. Comparisons for optical photon transport between EIC-Opticks offloading and standard Geant4 CPU workflow were performed. Agreement in output was confirmed, and a speedup of 148 ± 9 over single-threaded Geant4 and 9.3 ± 0.5 over a 20-thread Geant4 configuration was shown.

The event aggregation first implemented in EIC-Opticks reduced the average transport time per photon from ~ 60 μs (no batching) to ~ 20 ns simulating 10,000 events in one batch, a $\sim 10^3$ throughput gain.

Furthermore, we showed how important is to set the maximum reflections in EIC-Opticks, depending on the specific detector geometry in question. In the case of pfRICH, we showed that a setting of $\sim$24 is optimal. It yields the same number of hits as the Geant4 validation simulation while minimizing the GPU runtime as much as possible. EIC-Opticks provides the tools to carry out this optimization for arbitrary geometries.

We quantified the limitations of EIC-Opticks with respect to tracking error and "photon leakage". The latter is a common issue of ray tracing in the NVIDIA OptiX framework. We concluded that the tracking error is far lower than the typical uncertainty in the underlying modeling (e.g., non-perfect mirrors). We additionally showed how photon leakage can be minimized, and we provide scripts in EIC-Opticks so that users can optimize ray tracing's settings for their specific geometry.

By making GPU optical transport efficient in the low-yield regime, EIC-Opticks enables fast, large-scale optical studies for EIC detectors, accelerates geometry/surface scans, and unlocks the production of much larger training sets for machine-learning workflows. Although demonstrated for pfRICH, the approach generalizes to other Cherenkov and scintillator systems—including medical-imaging simulations—where per-event photon counts are modest. EIC-Opticks can be installed via Spack or used directly from provided container images.

8 Acknowledgement

We gratefully acknowledge Simon Blyth, the developer of Opticks, for his valuable help with this work and for many insightful discussions. We furthermore thank Alexander Kiselev and the pfRICH collaboration for providing us the geometry of the pfRICH detector. This work was supported by the Laboratory Directed Research and Development (LDRD) program under project 26794 of the Brookhaven National Laboratory.

References

[1] Simon C. Blyth. "Opticks: GPU Optical Photon Simulation for Particle Physics using NVIDIA OptiX". In: *Proceedings of CHEP 2018*. Preprint. 2018. URL: https://simoncblyth.github.io/env/report/opticks-blyth-chep2018-v1.pdf (visited on 09/24/2025).

[2] Youqi Song et al. "Particle Identification with the ePIC detector at the EIC". In: *arXiv* (2024). pfRICH overview: π/K up to $\sim$7 GeV/c, $\sim$40 cm gap. eprint: 2410.20410. URL: https://arxiv.org/abs/2410.20410 (visited on 09/24/2025).

[3] S. Agostinelli et al. "GEANT4: A Simulation Toolkit". In: *Nuclear Instruments and Methods in Physics Research A* 506.3 (2003), pp. 250–303. DOI: 10.1016/S0168-9002(03)01368-8.

[4] J. Allison et al. "Recent Developments in Geant4". In: *Nuclear Instruments and Methods in Physics Research A* 835 (2016), pp. 186–225. DOI: 10.1016/j.nima.2016.06.125.

[5] Simon C. Blyth. "Opticks: GPU Optical Photon Simulation via NVIDIA OptiX (7+)". In: *Proceedings of CHEP 2023*. Preprint. 2023. URL: https://indico.jlab.org/event/459/papers/11811/files/1068-opticks-blyth-chep2023-v0.pdf (visited on 09/24/2025).

[6] Steven G. Parker et al. "OptiX: A General Purpose Ray Tracing Engine". In: *ACM Transactions on Graphics* 29.4 (2010), 66:1–66:13. DOI: 10.1145/1778765.1778803. URL: https://research.nvidia.com/sites/default/files/pubs/2010-08_OptiX-A-General/Parker10Optix.pdf (visited on 09/24/2025).

[7] R. Abdul Khalek et al. "Science Requirements and Detector Concepts for the Electron-Ion Collider: EIC Yellow Report". In: *Nuclear Physics A* 1026 (2022), p. 122447. DOI: 10.1016/j.nuclphysa.2022.122447.

[8] Todd Gamblin et al. "The Spack Package Manager: Bringing Order to HPC Software Chaos". In: *Proceedings of SC '15: International Conference for High Performance Computing, Networking, Storage and Analysis*. 2015. DOI: 10.1145/2807591.2807623. URL: https://tgamblin.github.io/pubs/spack-sc15.pdf (visited on 09/24/2025).

[9] R. Abdul Khalek, A. Accardi, J. Adam, et al. "Science Requirements and Detector Concepts for the Electron-Ion Collider: EIC Yellow Report". In: *Nuclear Physics A* 1026 (2022), p. 122447. DOI: 10.1016/j.nuclphysa.2022.122447.

[10] Radovan Chytracek et al. "Geometry Description Markup Language for Physics Simulation and Analysis Applications". In: *IEEE Transactions on Nuclear Science* 53.5 (2006), pp. 2892–2896. DOI: 10.1109/TNS.2006.881062. URL: https://cds.cern.ch/record/1023367/files/cer-002682496.pdf (visited on 09/24/2025).

[11] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Managing Rounding Error (PBRT 3e, §3.9)*. 2018. URL: https://www.pbr-book.org/3ed-2018/Shapes/Managing_Rounding_Error (visited on 09/24/2025).

[12] N. Christensen et al. *Solving Self-Intersection Artifacts in DirectX Ray-tracing*. Concept applies equally to OptiX (ray_tmin/origin offset). 2023. URL: https://developer.nvidia.com/blog/solving-self-intersection-artifacts-in-directx-raytracing/ (visited on 09/24/2025).

[13] C. Chatterjee. "Particle Identification with the ePIC detector at the EIC". In: *Proceedings of Science* 469 (2025). 31st International Workshop on Deep Inelastic Scattering (DIS2024), WG6: Future Experiments. Pre-published on December 27, 2024; published on January 16, 2025., p. 266. DOI: 10.22323/1.469.0266.

[14] S. Yan and Q. Fang. "Accelerating Mesh-based Monte Carlo Photon Transport Simulation Using Ray-Tracing Hardware". In: *Proceedings of SPIE* 13308 (2025), PC1330806. DOI: 10.1117/12.3057023.