Simulating Particle Detection in Liquid Media Using Mitsuba3

MPhys Project Report

James Michael Patrick Brady
Department of Physics and Astronomy, University of Manchester
(Project researched in collaboration with Joseph Rodriguez, Olivia Borgstrom and Andrei Ghira)
(Dated: June 18, 2024)

Geant4 is a highly accurate particle physics simulation software, but is inefficient when simulating phton transport. Based on a previous proof-of-concept paper simulating a Cherenkov detector with a gas radiator in a modified fork of Mitsuba 3, the aim of this project was to create an automated pipeline to export GDML markup to XML formats and to extend the proof-of-concept to liquid radiators. A script to convert from surfaces from the GDML format to the OBJ format was successfully created but currently lacks the ability to export materials from the GDML format. Using a basic test to emulate a surrogate of a Cherenkov ring, Mitsuba 3 was able to recreate a comparable ring using a spherical mirror. However, a Cherenkov ring could not be directly simulated from data as in the previous project, likely due to the custom emitter not functioning correctly, so it will require modification to allow for exact comparisons to Geant4.

1. INTRODUCTION

Geant4 is the current standard for simulating particle physics interactions, however the single-threaded approach it uses notably slow at handling processes with minimal self-interactions, such as the propagation of photons. However, in the computer graphics industry, physically-accurate renderers are able to simulate the propagation of light through space in a fraction of the time owing to their highly parallel approach to simulating rays of light and ability to use consumer hardware to accelerate the calculations required. Mitsuba 3 is an example of a physically-based raytracing renderer, and has previously been used in a proof of concept to show that a Cherenkov detector with a gas-based radiator (based on RICH1 at LHCb) could be accurately simulated within Mitsuba in a fraction of the time it takes Geant4 to do so[1]. Whilst Mitsuba is reliant on Geant4 to provide the photons and ray directions that produce the Cherenkov ring, a case could be made to use Mitsuba as a replacement for Geant4's inbuilt rendering due to being far more efficient. However, this will require an adequate and automatic pipeline to export the detector markup to a form compatible with Mitsuba and it is currently unknown whether the similar accuracy and improved efficiency will be maintained for liquid radiators. In addition, some theoretical future detectors would benefit from the increased speed of simulation but are currently incompatible with the current capabilities of Mitsuba. If the efficiency gains and accuracy of Mitsuba are maintained for multiple detector media and its limitations can be addressed, its use would be a great benefit as simulations would take far less time to conduct and the would allow usage of highly scalable and affordable consumer GPUs to accelerate simulations further.

2. OUTLINE OF THE PROJECT & THEORY

2.1. Why Use Mitsuba 3?

Geant4's [2] simulation is accurate but notably slow when handling photon propagation, however a raytraced physically-based renderer such as Mitsuba 3 [3] would be appropriate for a static and consistent process such as photon propagation. Photons also not interact with each other, so the problem is embarrassingly parallel, therefore it makes sense to use multi-threaded computation or GPU core computation to simulate photon transport - which is something a Physically-Based Renderer that uses ray-tracing (i.e. Mitsuba 3) is exceptionally efficient at.

A previous proof-of-concept [1] [4] demonstrated that Mitsuba 3 simulates photon transport in a basic gas-filled Cherenkov detector (consisting of a transport medium, two mirrors and a detection plane) to a high degree of consistency with Geant4 - with some discrepancies based on the plane where each toolkit detects photons. These discrepancies were fixed by coordinate-shifting one of the results to the detection plane of the other, where the respective Cherenkov rings were shown to overlap. Mitsuba 3 also proved to render much faster than Geant4 for $\gtrsim 10^4$ photons, where Geant4 scaled linearly with the number of photons and Mitsuba 3 did not scale at all with the number of photons (found to be because the bulk of computation time was being used for kernel compilation rather than photon propagation). GPU and multi-core CPU rendering took the same amount of time as each other, both were a factor of ten faster than rendering using a single CPU thread.

The next step is to show whether this speed and accuracy is maintained for simulations of complex liquid detectors in the vein of Hyper-K and DUNE and more complex gas detectors such as LHCb's RICH1.

2.2. Why Not Use Blender/Cycles?

Blender [5] is a similar free-and-open-source toolkit that can be used to render scenes using an inbuilt PBR raytracing renderer, Cycles, it differs by acting as a self-contained program with a GUI interface and is designed to be a general-purpose 3D media suite. In contrast, Mitsuba 3 is only designed to render pre-defined scenes via command-line or Python script, but specialises in high-accuracy research rendering.

Whilst Cycles uses the same underlying physical principles as an unmodified version of Mitsuba,

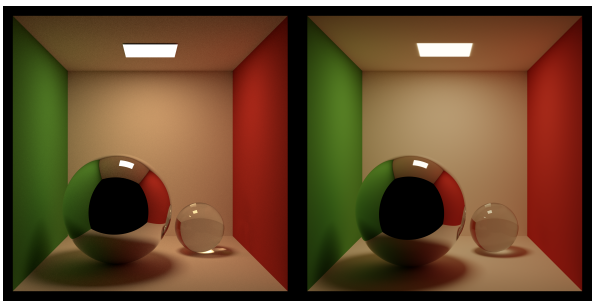

FIG. 1. A comparison between Mitsuba 3 (left) and Blender 3.6 Cycles (right) renders of a standard test scene, both at 2160x2160px, 590 samples/pixel without denoising,. Notably, glass caustics due to the area light can be seen on the red wall and the glass ball's shadow in the Mitsuba render, but is only approximated in the corresponding Cycles render.

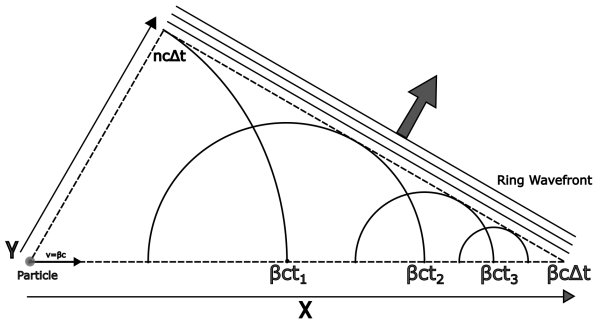

FIG. 2. An optical diagram showing Cherenkov emission. A charged particle travels at a speed $\beta c < c$, whilst the spherical wavefronts it emits travel at a speed $nc < \beta c$, constructively interfering to create planar wavefronts.

Cycles struggles to handle certain optical effects that Mitsuba can intrinsically recreate, notably glass "caustics" (focusing rays of refracted/reflected light through curved dielectric surfaces) and thin-film interference effects, which can be seen in Figure 1. [1]. These "missing" physical effects can be implemented into Cycles either by directly modifying the rendering pipeline's source files and recompiling Blender from source, or by defining the relevant optical equations in the BSDFs used to define the materials in a given scene. However, Blender is highly useful for manually creating mesh surfaces and can be compared to Mitsuba for basic optical effects such as reflection, so would have a use in this project.

The drawbacks of modifying the Blender source is that although it is built on C++ and free-and-open-source just like Mitsuba is, it has a far greater overhead and size compared to Mitsuba due to the additional non-render features in the program. Mitsuba already implements more features relevant to a proof-of-concept investigation such as interference by default, meaning that it requires less modification than Blender.

Overall, it would be possible but far more impractical to use Blender over a specialised package like Mitsuba given the increased program overhead and additional modifications required, but it has a place as a useful tool for checking that geometry outputs from Geant4 are correct. In addition, basic optical effects such as reflection and refraction could be compared to using simple surrogate tests to make sure that modifications made to Mitsuba work as expected.

2.3. Cherenkov Radiation

Cherenkov radiation is continuum radiation produced by charged particles moving through media at a speed greater than the local speed of light. Charged particles polarize materials due to introducing electric fields to the molecules within a material. However, the particle travels faster than the spherical wavefronts it emits, causing the wavefronts to interfere to produce a conical wavefront - leading to a biased polarization of the material and causing photons to be emitted along a cone instead of isotropically [6]. This leads to an angle of emission dependent on the refractive index and the speed of the particle as a fraction of the vacuum speed of light, β, as demonstrated in Figure 2:

$$\cos\theta_C = \frac{1}{n\beta} \Rightarrow \beta_{cr} = \frac{1}{n} \tag{1}$$

A differential describing the number of photons emitted per unit wavelength per unit path length travelled can be found [6]:

$$\frac{d^2N}{d\lambda dx} = \frac{2\pi\alpha}{\lambda^2}\left(1 - \frac{1}{(n\beta)^2}\right) \tag{2}$$

If the refractive index of a material, n, is constant, then this leads to an infinite amount of emission at short wavelengths which is not physical. However, n is not constant across all frequencies of emission, i.e. $n = n(\omega)$ as molecules act as oscillators, making the real value of the dielectric constant, $\eta(\omega)$, depend on frequency. There will be resonant frequency where $\eta(\omega)$ peaks and as $n = \sqrt{\eta(\omega)}$, n also peaks, after which it sharply drops [7]. This will lead to the requirement that $\beta_{cr} \geq 1$ to produce Cherenkov radiation, which is not possible, hence the number of photons emitted $N \to 0$ continuously at a minimum wavelength, below which emission stops.

2.4. Rayleigh Scattering

For water, the Cherenkov spectrum produced is within the visible spectrum and water molecules have an average size of 0.1 mnm, making Rayleigh Scattering highly dominant over Mie Scattering. This results in an approximately isotropic light scatter, meaning that Cherenkov rings can appear blurrier if the scattering is at a small angle, or dimmer if scattered away from the ring. The intensity of emission varies with wavelength λ, particle diameter d and angle θ [8]:

$$I = I_0 \frac{\pi^2\alpha^2}{\epsilon_0^2\lambda^4 d^2}\frac{1+\cos^2\theta}{2} \tag{3}$$

[1] As of Blender 4.0.1, caustics are far more accurate and comparable to Mitsuba's results, but still lack some accuracy

Using the definition of the scattering cross-section, σ_s, a mean free path Λ_{MFP} can be estimated using the particle number density of the material n_{mat}^-[8]:

$$\sigma_s = \frac{2\pi^5}{3}\frac{d^6}{\lambda^4}\left(\frac{n^2-1}{n^2+2}\right)^2 \Rightarrow \Lambda_{MFP} = \frac{1}{n_{mat}^-\sigma_s} \quad (4)$$

Calculating the cross-section of scattering in pure water yields a value of approximately 200m. In the Hyper-K detector, the maximum path length of a particle will be 100m, therefore on average half of the photons should scatter once, assuming that all photons will have scattered after travelling one mean free path and that there is a constant probability of scatter - so scattering remains significant but not necessarily frequent, especially for smaller water-based detectors.

2.5. Absorption of Photons and EM Showers

For pure water, absorption is generally insignificant as the vibrational and rotational transitions are at energies within the IR & Microwave ranges as given by the HITRAN database [9]. The first excitation energy of atomic Hydrogen is $\sim$ 10.2 eV (estimating from the Rydberg energy), corresponding to 122 nm. As Cherenkov spectra in water are expected to fall within the short-visible spectrum, absorption should not contribute significantly to the absorption of photons produced by Cherenkov processes.

However, depending on the particle emitting Cherenkov radiation, electromagnetic showers can be highly prevalent. As particles can emit a continuum of photons via bremsstrahlung, photons with an energy of $\geq 2m_ec^2 \approx 1$ MeV can pair produce $l\bar{l}$ pairs, most frequently $e\bar{e}$ pairs [6]. Electrons emit via $l\bar{l}$ pairs frequently due to their low mass. Therefore, an electron with a very high momentum can produce a cascade of photons and other leptons, which themselves can produce more photons, making a Cherenkov ring almost impossible to discern. An EM shower could be initiated by an electron with a Lorentz factor of 3, corresponding to a β of 0.943, However, leptons of higher masses, such as muons, scatter far less frequently, making EM shower production far less frequent despite requiring lower values of β to initiate [8]. Therefore, it would be preferable to simulate muons to test Cherenkov detectors.

Given that the typical collision energy of the LHC is 10 TeV (i.e. high momentum muons are likely to be produced), it is reasonable to simulate high-momentum muons to test Cherenkov emission. Therefore Rayleigh Scattering becomes the dominant source of error within water simulations, and must be accounted for as accurately as possible within Mitsuba 3.

3. PROGRESS & RESULTS

At the beginning of the project, tasks were split between Joseph and myself (the Physics students) and Olivia and Andei (the Computer Science students). Joseph and I were to focus on the physics required to understand the simulations in this project and thus perform the data analysis and draw conclusions. Olivia and Andrei were asked mainly to handle the I/O processes required to export data from Geant4 to Mitsuba and to extend the functionality of Mitsuba as necessary. In the subsections below, any work relating to simulating curved detector surfaces and the pipeline between Geant4 and Mitsuba was carried out by Olivia and Andrei, but is required in this report to provide a general overview of all work completed so far. However, both of our pairs were advised to work closely with each other regardless and some of our work has overlapped, so whilst I am allowed to discuss their work, it is important that I note what each person contributed to the project.

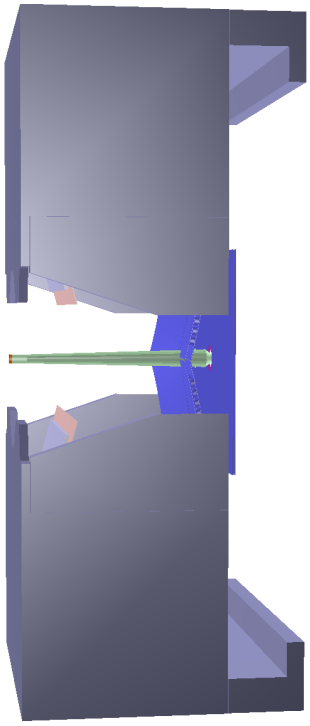

FIG. 3. An example of an OBJ file converted from GDML using Olivia's script, in this case the RICH1 detector at LHCb.

3.1. Geometry, Material & Source Conversion

Olivia was able to create a script that converts from GDML detector geometry, to OBJ files which can be used in most 3D software suites. The script was tested by using the geometry of the RICH1 detector at the LHCb experiment, as shown in 3. Conversion to OBJ allows for inclusion of the detector geometry (with materials) into Mitsuba's XML-based scene format and is highly modular - as parts of a detector with the same purpose will be constituted of the same material or similar materials, separation by material is a highly effective way to isolate the components of a detector from each other and hence discriminate what parts are necessary for an accurate render. For example, if the shielding of a detector increases the time required to render but does not participate significantly with the transport of a Cherenkov ring, it can be easily excluded.

After using Blender to check that the OBJs had been generating properly when using the script, an

error was found where detector components were loaded at a scale of 1000x real size - which is likely due to Geant4 working in mm as its base unit, rather than in m as Mitsuba and Blender do. Because of this, the scale should be reduced at generation. Another issue has been that every individual element within the generated OBJ files has a unique material assigned to it - rather than one unified material - however, loading OBJ files as Mesh shape objects in Mitsuba allows for all materials within a file to be overridden by a single material. Given that all mesh elements within a single file should have the same material, this is not an issue for our application of this project - but it will become an issue if benchmarks to other renderers are needed.

Geant4 materials are not directly compatible with the BSDF standard (Bidirectional Scattering Distribution Functions), algorithms that determine how light interacts with surfaces and volumes that Mitsuba and many other physically-based renderers use. Geant4 relies on proportions of elements/materials to calculate physical and optical properties as a part of its GDML standard of surfaces and materials. Unlike Geant4's GDML implementation, BSDF materials rely on average macroscopic properties (e.g. refractive index and surface roughness) to define materials + has different types of defined boundary properties (diffuse reflection, internal scattering etc.).

In Mitsuba, some BSDF parameters are set for physically accurate optical transport (i.e. conservation of energy) by default, leaving the most important properties to be set - Roughness (0 for fully specular reflection, 1 for fully diffuse reflection) and η and k (the real/imaginary parts of the index of refraction, respectively). Roughness in Mitsuba is defined as the root mean square of the slope of microscopic details within the Beckmann-Spizzichino model [10], which modifies how light will reflect from a smooth surface.

Unlike other raytracing algorithms such as Blender's Cycles, metals/conductors (opaque materials that reflect at the surface), dielectrics (translucent materials with reflection & refraction) and plastics (dielectric materials with internal scattering) are specified as separate predefined material types alongside the Disney Principled BSDF (which is the industry standard for artists, but is not designed for physical units and experiments such as this). Having predefined material types by default rather than having to combine several simpler material BSDFs (such as glossy/diffuse surfaces) and manually implement physical effects such as thin-film effects and Fraunhofer & Fresnel diffraction is highly convenient for this experiment, as it provides a well-established physically-accurate basis to work from.

Geant4 must be able to calculate the index of refraction internally in order to accurately simulate photon transport, so there may be a way to extract it or calculate the frequency-dependent index of refraction outside Geant4. Otherwise, the best alternative is to use reference values around the wavelengths being propagated within translucent media. It is also important to consider whether any materials used in a detector polarize/delay rays or cause anisotropic effects, as these can be accounted for using Mitsuba's material system. Therefore, the next steps should be to find a way to translate the material definitions in Geant4 to Mitsuba-compatible BSDF materials as part of the parsing to XML and OBJ files and ensure that the scale is fixed during the parsing from GDML. If possible, it would be convenient if all elements combined in a single OBJ file had the same unified material, but this is not necessary if they are only to be used in Mitsuba, as previously stated.

3.2. Rendering of Curved Detector Surfaces

Mitsuba provides two types of sensors that can be used to directly measure power incident to points - the radiance meter, which measures the power in W per unit area per steradian along a given direction and the irradiance meter, which measures the power in W per unit area incident over all the triangles of the mesh it attaches to. However, both of these detectors are currently unsuitable for use in our applications, as both only work with 1x1 films, the radiance meter only measures along a single ray direction (the documentation states that it acts as the limit of a perspective camera as the field of view/FOV tends to 0°) and the irradiance meter averages the irradiance over a whole surface. These limitations makes using unmodified version of these sensors highly inconvenient, so rendering from perspective sensors is the currently the best method of recording the data. This rules out simulating detector experiments such as Hyper-Kamiokande (also referred to as Hyper-K) for the time being, especially as Hyper-K has a mostly curved detection surface.

It is possible to batch render several sensors focused on a single point and correct for spherical coordinates to create a superposition image - Andrei has has been able to implement and demonstrate this with a textured cube. Theoretically, the process could be reversed to render multiple cameras facing radially out from a single point, and stitch together individual film renders to create a single flat image, which would be similar to the presentation of data from experiments such as Hyper-K.

Another avenue would be to implement a method to record the light values at surface of a detector to a UV-mapped texture (also known as "baking"), which would remove the requirements to use multiple sensors and also remove a source of distortion via the use of perspective sensors. However, this method is more within the domain of fully-featured 3D development programs (such as Blender or 3DS Max)/specialist material creation suites (such as Adobe Substance) rather than ray-tracing algorithms, so this is likely to be far more complicated than stitching together several camera renders. It might also be possible to modify the irradiance meter to output to a film larger than 1x1 pixels and to record values per quad face (as they are far easier to tessellate in a way that maps well to a uniform pixel output) to produce a similar output. However, this will require the detector planes to be meshed in such a way that each sensor in a physical detector has a exactly one quad face corresponding to it, matching the area they cover - so precisely defining detector surfaces in Geant4

would be crucial.

3.3. Simulation of Cherenkov and Scintillation Photon Transport

In order to avoid EM showers but check Cherenkov radiation detection for high-energy experiments, it is ideal to use muons instead of electrons given that their production will be somewhat frequent, electrons tend to have multiple scattering (which introduces noise in the propagation - which not ideal for testing accuracy) whereas the higher mass of highly suppresses the production of showers and the scope of the project does not include EM showers.

A basic Cherenkov ring detector was created by Joseph in Geant4 to test transport within liquids: it consisted of a 1x1x1.5 m cuboid of pure water ($\Re(n) = 1.333$) centred on the point (0,0,0), a muon with $\beta = 0.995$ travelling along the long edge of the detector (defined as the z-axis, moving in the positive direction) from the centre and a flat 1x1m detection plane located at (0, 0, 0.49). A Cherenkov ring will therefore be emitted with an angle of 41.07°. Within the produced simulation using the optical physics libraries, scattering was not significant due to the reduced scale, no EM showers were initiated and absorption effects were not seen, meaning that the assumptions made previously are likely accurate to some degree. Using the script created by Olivia, this was translated into an XML file using the generated OBJ files for the detection plane and and water medium. The OBJ files were not provided with the XML when I received it, so the detector was redefined in the XML easily using the primitives and transforms provided in Mitsuba's framework. Using the detector in Geant4, Joseph produced data for Cherenkov emission - as both a circle of concentric, overlapping rings by allowing the muon to travel uninterrupted and as a single ring by destroying the muon after a single step.

A major issue with attempting to simulate photon transport in Mitsuba was that the modifications made to the renderer, namely the photon emitter, were not documented in any available form [4]. As a result, we examined the files included in the "MPhys" directory of the Manchester repository branch of Mitsuba 3, corresponding to testing done on the modifications as part of the previous project, in an attempt to reverse-engineer the workings of the custom emitter. The commits made to the GitHub repository also provided some clarification as to where changes had been made. The examination of the relevant scripts and the source file for the photon emitter was partially successful - they indicated that a photon emitter needs to be created at runtime in a Python script and how the data needed to be passed to the emitter's constructor.

In this case, data was passed a Numpy array floats with dimensions [1, 1, 6N + 1] corresponding to N photons, which is constructed as a VolumeGrid object in Mitsuba when passed to the photon emitter's constructor. The element of this array along at [0, 0, 0] is equal to N - used for parsing how many photons need to be created - and the elements in each block [0, 0, i:$i + 6$] ($1 \leq i \leq N$) correspond to the data required for the i^{th} photon, the first three are the positions relative to the origin and the last three are the normalised directions of momentum. This data can be extracted from Geant4 via an output to CSV. However because the data provided is not scaled from mm to m within the emitter's constructor and uses standard Mitsuba transforms, it is implied that either the geometry needs to be scaled, as in some of the example tests (which is not physically accurate for highly dispersive media) or that the data needs to be scaled to m before being used.

Photon emitters are constructed as spotlights with a very narrow angle, which effectively constrains all emissions to a small beam pointing away from the photon origin position in the direction the photon moves in when simulated in Geant4 and prevents isotropic emission - so this becomes a very good substitute for sampling single photons without having to modify the default RGB path tracing. Light falloff is also disabled for the photon emitter so that the intensity does not decrease as the area of the spotlight increases at far distances.

However, there is a major limitation to using this constructor - Mitsuba does not allow scenes that are loaded from either XML files or Python dictionaries to have objects added to them once instanced - which means that in order to use the photon emitter, the entire scene must be defined as a dictionary and then instanced. Relying on Python dictionaries is not ideal, as they are more difficult to format and understand than XML files and need to be rewritten or commented out if another scene needs to be loaded. Therefore, it would be wise to rewrite the constructor so it can accept a text or CSV file (as Geant4 defaults to outputting particle data as a CSV) at construction, using a header such as ¡fstream¿ to parse the data and allowing files to be referenced in the XML description. Specifically, the runtime dependence should be moved to the C++ source file for the photon emitter to maintain consistency with the rest of Geant4, especially as OBJ files and bitmap textures are parsed in this way and allow for the use of XML files.

Another issue with attempting to simulate the photon transport has been that the previously mentioned files used for testing no longer function - indicating that there has been an undocumented change to the emitter source file made at some point after testing. As a result, limited information could be gained by examining the Python scripts from the previous project as the simulations could not be directly reproduced from these tests. However, given that the method of data input had been found, the updated emitter should theoretically work.

Attempting to directly image light that is emitted from a single point - such as a point light or spotlight - is impossible in Mitsuba as these emitters do not have geometry, so must be detected via reflection from and transmission through surfaces. As virtual perspective sensors propagate rays from pixels to object surfaces in a backwards chain to an emitter, there must be at least one surface involved in detecting emission. Given that photons act as

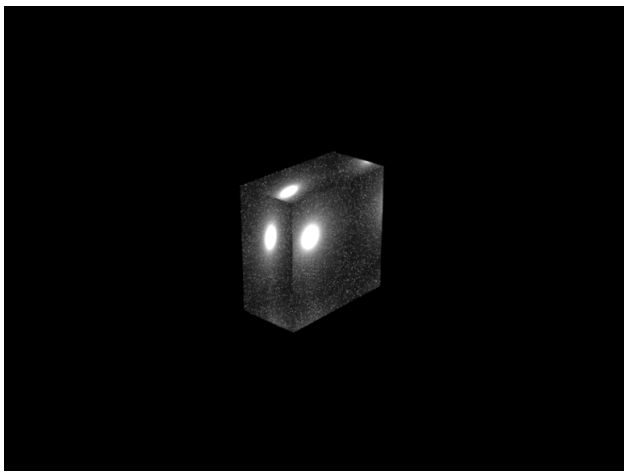

FIG. 4. A demonstration of emission from the photon emitter inside the water medium being reflected by a diffuse plane. Internal reflection and refraction are also observed, indicating that light is being transported accurately

geometry-less spotlights, replacing the detection plane with a single perspective camera will not work, so a mirror surface needs to be imaged to produce visible rings.

This produces a unique problem for the simple detector used in this test as there are no mirrors involved, unlike most real detectors such as RICH1, which use mirrors to reflect the rings on to a detector. In this case, using a mirror will require the photons to travel a further distance through the medium in Mitsuba than the photons detected in Geant4, introducing the possibility of further scattering. Whilst it may be possible to image the 1x1m square face of the water volume in the positive z direction, this would introduce possible errors through internal scattering and/or transmission out of the volume.

Using a flat, perfectly reflective mirror may mean that a Cherenkov ring may not be detected as it must only reflect the Cherenkov ring at its incident angle. In this case, the rays produced by the photons will be unable to reflect backwards and so cannot be detected directly by a perspective sensor. However, taking inspiration from real Cherenkov detectors and the optics of radio telescopes, it would make sense to use a concave spherical mirror with a radius of curvature centered on the origin (where the sensor is placed) so that the image formed is constituted of of parallel rays perpendicular to the sensor. Subsequently, this allows for rays to be traced from the camera backwards to a photon whilst maintaining the reflection angle, so an image can be produced. This may not work for a perfectly smooth mirror, as sensors cast rays at an angle from a central point through each pixel, but the mirror would need to be far less diffuse than a flat mirror to allow for valid rays to be cast.

A mistake made early on in testing the liquid medium is the usage of a fully diffuse plane as the detection surface, which effectively blurs out any formed image across the plane and so would not be appropriate for accurately recreating the emission patterns seen. However, this provided evidence that the photon emitter was emitting light and doing so from the correct origin when scaled to m, as seen in figure 4.

Whilst Blender is not as physically accurate due to aforementioned missing optical effects, the basics of ray reflection and scattering should be very similar to Mitsuba. Hence, it was decided to check the photon emitter's results against a similar surrogate in Blender at the same scale, also providing an opportunity to model and test a spherical mirror mesh which could then be imported into the Mitsuba detector scene in place of the flat detection plane.

The surrogate to the Cherenkov ring photons used in Blender was a single spotlight partially blocked by black diffuse disc at a distance along its spotlight's axis, both of which were behind the sensor (located at the scene origin). As a result, a uniform ring of light emitting at a fixed range of angles would be produced, analogous to a Cherenkov ring. The spotlight and disc had to be located behind the camera, as the diffuse circle is a renderable surface and would obscure part of the mirror, so moving the spotlight system to the origin (as the photons are emitted relative to the origin in the Mitsuba scene) would produce an invalid comparison.

The spherical mirror is centred on the origin and has a radius of 0.6484m, so that the light of a Cherenkov ring emitted from the origin will reflect from the points of the mirror located in the plane z = 0.49m, therefore producing a ring of the same size as the one seen by the flat detection plane in Geant4. Because the rays are reflected parallel to each other by this mirror, the size of the image remains constant between the mirror and sensor. Though there is still a small probability for photon rays to scatter between the mirror and sensor, this can be mitigated by using a high number of samples/pixel to produce a more precise render. Within Blender, the mirror was given a metallic material (analogous to Mitsuba's conductor BSDF) with an adjustable Beckmann roughness. This mirror was then exported from Blender to an OBJ file which was loaded into the Mitsuba scene and assigned the mirror material as previously discussed

As the spotlight needs to be placed behind the camera in Blender, it was decided that it would be placed at (0, 0, -0.51) so that there would be a 1m separation between the spotlight and plane's position, and the disc would be placed at (0, 0, -0.01) so that it would be halfway between the spotlight and the detection plane.

As the Cherenkov ring is emitted at an angle of 41.07°at a distance of 0.49m from the detection plane, the spotlight needs to emit at an angle of 23.12°to produce a ring of the same size on the mirror, and the disc can have a maximum radius of $\tilde{0}$.21m before fully blocking the spotlight. A sensor with a 1:1 aspect ratio requires a horizontal field of view of 91.16°when located at the origin, so that it only observes the area the corresponding to the detection plane.

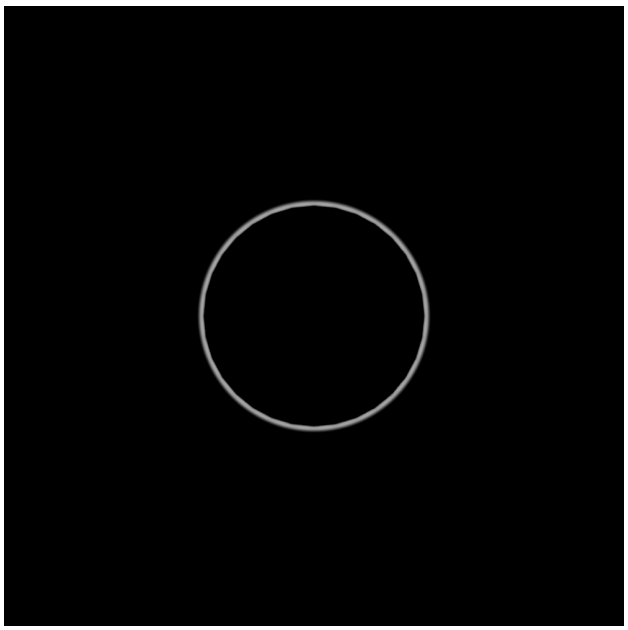

FIG. 5. A test Cherenkov ring produced in Blender using a spherical mirror, a spotlight and a black circular plane to obscure part of the spotlight.

The spotlight was given a light emission radius of 0.005m to ensure that the the ring has some thickness.

Testing in Blender, the ring was not visible for a Beckmann roughness between between 0 and 0.03, but was for roughnesses between 0.03 and 1 - showing that both reflective and diffusive materials could work when used on a spherical mirror. The higher the roughness value, the more light will bounce around the mirror's surface, increasing the background luminance across the mirror but also increasing the luminance of the ring. The background luminance seems to be roughly constant across the mirror given many samples ($\sim$ 128) so it is easy to subtract from the resulting render. A good compromise was found at a roughness of 0.04, where the ring appeared bright and the background remained relatively low, which is shown in Figure 5.

Using the same environment in Mitsuba and changing the disk material to a diffuse material with no reflectance (hence absorbing any light in its path), a similar ring to the one observed in Blender was produced from the spotlight. The ring produced by Mitsuba is very similar to the ring output by Geant4, as shown in Figure 6, providing good evidence that using a concave spherical mirror is the correct approach to image the Cherenkov ring. The spherical mirror behaves in a very similar way in both Mitsuba and Blender, but comparing Figure 5 and 6 shows that the ring is twice as wide in the image produced by Mitsuba despite maintaining the same field of view, indicating that Blender likely uses the half-angle to define the field of view, but Mitsuba uses the full angular distance to define it.

Attempting to render the photon data directly with

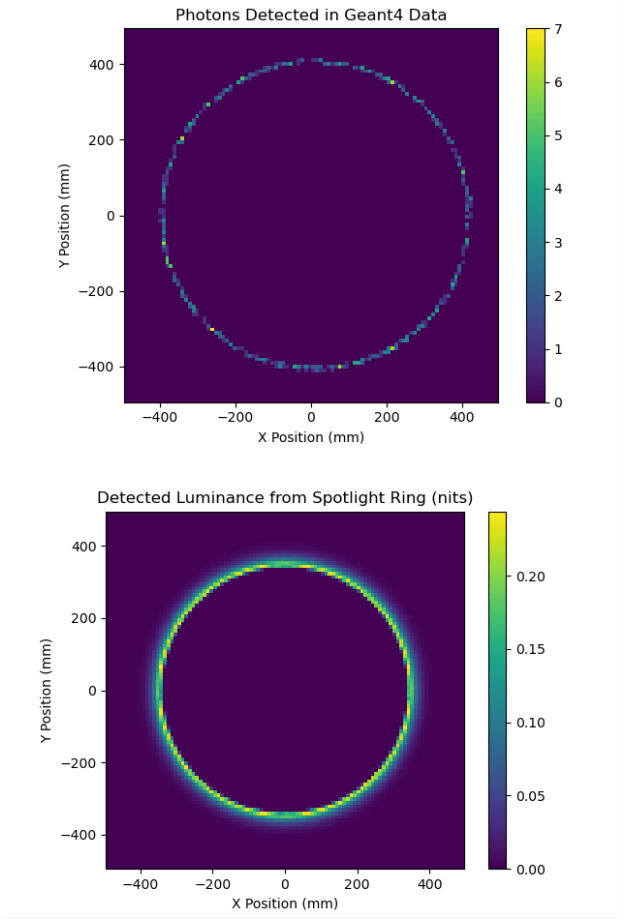

FIG. 6. A comparison between the ring produced by Geant4 (top) and the spotlight and spherical mirror in Mitsuba (bottom). The size is generally similar, providing evidence that the use of a spherical mirror could provide an accurate image of the Cherenkov ring simulated using the photon emitter.

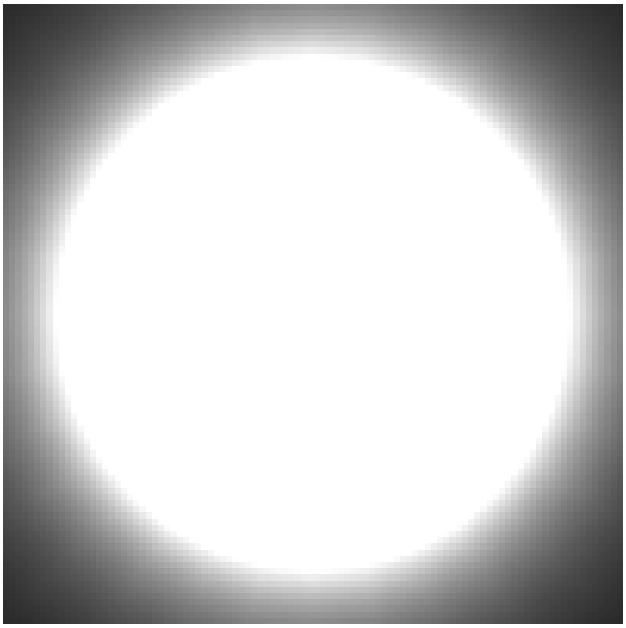

FIG. 7. A simulation of the Cherenkov ring in Blender using the photon emitter. Despite using the same roughness value, a diffuse circle is produced, indicating that the emitter is not functioning as desired.

the same detector (but with the spotlight and disc replaced for a photon emitter) produces a fully-white image, moving the camera away from the axis of the mirror by 5cm and reducing the roughness value from 0.04 to 0.01 produces a solid white circle with a diffuse edge, as demonstrated in 7. If a Cherenkov ring is being observed, this should not be seen - but the data was verified as a Cherenkov ring in the Geant4 simulations and the ring produced by the spotlight and disc is similar to the ring produced in Geant4. Overall, this suggests that Mitsuba is transporting the light produced by a ring at a fixed angle correctly and that the customised photon emitter is not working as intended. Specifically, the emission produced it resembles a narrow spotlight pointing along the positive z axis and quickly diffuses at low values of roughness due to the high intensity of the emission.

However, it has not been possible to determine why this emitter is failing to work as intended and why similar results to the previous experiment cannot be produced. As the scripts provided from the previous project actively attempt to render their included scenes, including the corresponding graphs showing Cherenkov rings, but they failed to render when run. This further supports the possibility that the emitter was modified at some point after the tests had been conducted, but the specific changes had remained undocumented and so there is no immediate way to understand how the emitter should function in its current state. Given that the source will need to be modified to allow for a file reference to be taken in XML and parsed when loaded, it would also make sense to look through the entirety of the source to find potential places in construction where this behaviour fails. The issue could possibly arise during use of the Dr. Jit library included with Mitsuba to create the kernel, as it handles the data provided to it differently to inbuilt C++ data types and objects and is not directly included in the source file.

4. CONCLUSION

The project has so far seen several successes and setbacks. Out of the main objectives set at the beginning of the project, a pipeline to directly convert GDML detector descriptions to OBJ files has been achieved and meshes surfaces consistently, allowing for detectors to be ported to XML scenes for Mitsuba without relying on an intermediate program. However, the conversion script is currently unable to include materials from a GDML file due to being higly different to the BSDF standard used in most 3D software suites, so the next step in creating a Geant4 to Mitsuba pipeline could be to find a way to extract the required refractive index and roughness from the GDML description.

Likewise, progress has been made on simulating detectors with curved surfaces, such as Hyper-K, in Mitsuba. Whilst the sensors included in Mitsuba designed for measuring light incident on surfaces are inadequate for use at large scales, it is possible to batch render and combine the renders from several perspective sensors at once, which is similar to the approach used in the previous project. A superposition image of a cube from several angles was produced, corrected for spherical coordinates, which could theoretically be used to create several renders of a single curved surface which could then be combined into a single image. If this approach works, then implementing curved detectors should not be much more complex than implementing detectors with flat detection planes. However, if this approach does not work, then an alternative that will allow the light incident to a curved surface to be baked to a texture should be implemented instead - though this will require detector surfaces to be UV mapped adequately and precisely, which would require more work.

Theoretically, Mitsuba 3 should produce accurate results for Cherenkov ring propagation in liquid media. The Cherenkov radiator and detector created in Geant4 used a flat plane to detect photons, which is not ideal in Mitsuba as a diffuse material is required to detect the ring due to conservation of reflection angles on perfectly reflective surfaces and further travel through a dispersive medium. However, using a test in Blender that produces a uniform ring of light radiating with the same angle as the simulated ring showed that a concave spherical mirror could be used to replace the flat plane. The main advantages of using a concave spherical mirror are that the image is reflected as parallel rays, so the size of the image remains constant and the angular deviation required to detect the image is lower so the detector material can be less rough than the material required for a flat plane. Using the same test in Mitsuba produced a ring that was highly comparable to the one produced in Geant4, meaning that Mitsuba will likely propagate a ring made of photons accurately. However, whilst the photon emitter created for the previous project does emit light, it does not accurately recreate a Cherenkov ring despite the passed data describing the same size of ring. Instead, the photons seem to render as a single small spotlight pointing along the axis of the detector, rather than several pointing at a fixed angle away from the axis. It is likely that the constructor of the photon emitters is not correctly setting the target positions of the spotlights used, though a lack of documentation of the modifications and a lack of working tests has made finding the exact error difficult. Given that part of the modification's source needs to be rewritten to be reference the data in an XML file rather than creating the emitters within the Python script at runtime, it would make sense to investigate and rewrite the other parts of the source file, then provide documentation if and when it works as desired.

[1] A. C. S. Davis *et al.*, Optical photon simulation with mitsuba3 (2023), arXiv:2309.12496 [physics.comp-ph].

[2] S. Agostinelli *et al.*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **506**, 250 (2003).
[3] W. Jakob *et al.*, Mitsuba 3 renderer (2022), version 3.1.1, available from: https://mitsuba-renderer.org.
[4] A. C. S. Davis *et al.*, Mitsuba 3, manchester fork (2023), as used by A. Davis et al., 2023.
[5] B. Foundation, Blender (2023), version 3.6.0, available from: http://www.blender.org.
[6] B. R. Martin and G. P. Shaw, Particle physics (Wiley, 2008) Chap. 4.4 - Experimental Methods, 3rd ed., sections 4.4.5 and 4.4.6.
[7] L. Fülöp and T. Biró, International Journal of Theoretical Physics **31**, 61 (1992).
[8] E. Hecht, Optics, global edition (Pearson, 2016) Chap. 4.2 - Rayleigh Scattering, 5th ed.
[9] I. E. Gordon *et al.*, Journal of Quantitative Spectroscopy and Radiative Transfer **277**, 10.1016/j.jqsrt.2021.107949 (2022).
[10] P. Beckmann and A. Spizzichino, *The scattering of electromagnetic waves from rough surfaces*, new ed. (Artech House Publishers, 1987).