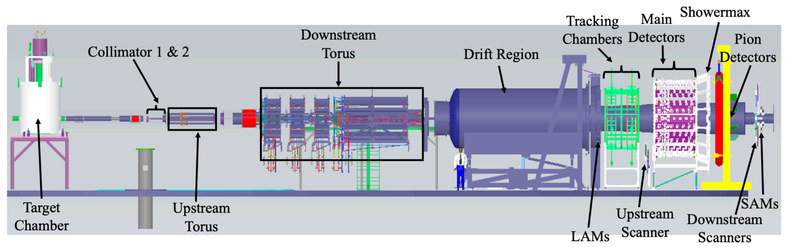
Program to Identify Secondary Background Sources in the MOLLER Experiment

L.A. Barrett, J. Mott, K.S. Kumar

UMass Amherst Department of Physics

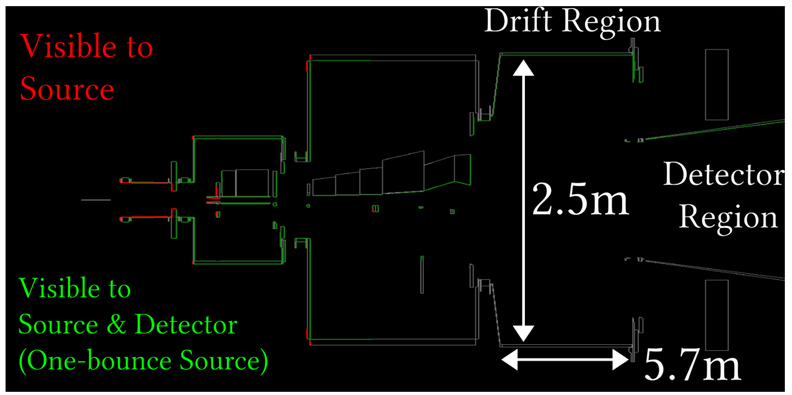
I. The MOLLER Experiment

- MOLLER seeks to measure the parity-violating asymmetry of Møller scattering A_{PV} using the 11 GeV beam in Hall A at Jefferson Laboratory (JLab)
- Will improve on previous measurements by a factor of five, providing the most precise measurement of the weak mixing angle $\sin^2\theta_W$ to date,



II. Twobounce Conditions & Twobounce 2D

- Electrons accelerated to 11 GeV are incident on a liquid hydrogen target.
- The incident beam causes high energy photons to scatter from the target, which could collide with other surfaces within the experimental chamber, consequently creating an additional undesired source of secondary particles



- Collimation system is designed to eliminate all one-bounce sources from the detector region

III. Twobounce 3D Goals & Approach

We have developed *Twobounce 3D*, aiming to extend Twobounce into three dimensions.

- There could be paths not accounted for in 2D - while the apparatus largely has rotational symmetry, it is not perfect.

This problem resembles a concept in computer graphics: global illumination.

- We can treat the hydrogen source and detector as light sources, and consider what areas are illuminated by both
- To use standard computer graphics techniques, the geometry must first be exported and triangulated, as triangles provide the fastest intersection algorithms
- The package *pyg4ometry* is able to support exporting the Geant4 GDML to the standard GLTF format. This functionality, along with the standard 3D software Blender, was used to triangulate the geometry and map a texture file to it.

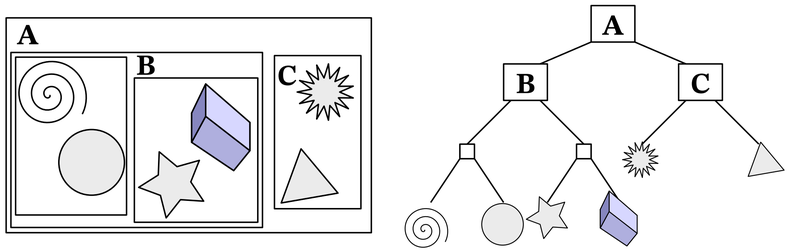
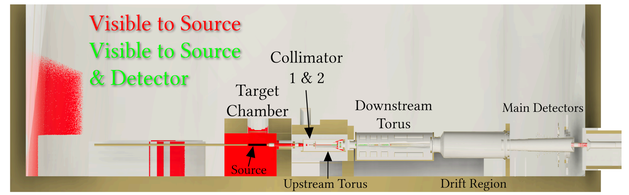


Figure 3: Construction of BVH tree

- The geometry consists of 265,000 triangles. This would make naive ray-checks extremely slow
- By organizing the geometry into a bounding-volume heirchy tree (BVH tree, Figure 3), the geometry can quickly be sorted through (ideally $O(n) \rightarrow \sim O(\log_2 n)$, up to 10,000x faster!)
- It was programmed in Rust, a langauge similar to C++ focusing on memory management and speed, allowing for inherently faster computations than Twobounce in Python
- Supports multithreading for an additional speedup

IV. Results & Conclusions



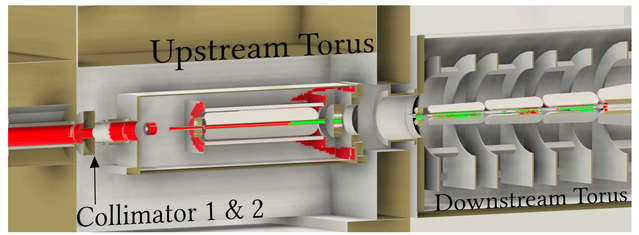


Figure 6: Drift and Main Detector Region

The above figures show the model output by Twobounce 3D, visualized in Fusion 360. They all show a side-cut view.

- In the drift and detector regions (Figure 6), we don't see any new signs of onebounce sources

These results validate the twobounce collimation design

V. Acknowledgements

I would like to thank Prof. Stewart Boogert from the University of Manchester, whose work on maintaining and improving pyg4ometry greatly eased working with our geometry and ensured an accurate translation. https://pyg4ometry.readthedocs.io/

This research was partially funded by the DOE and NSF