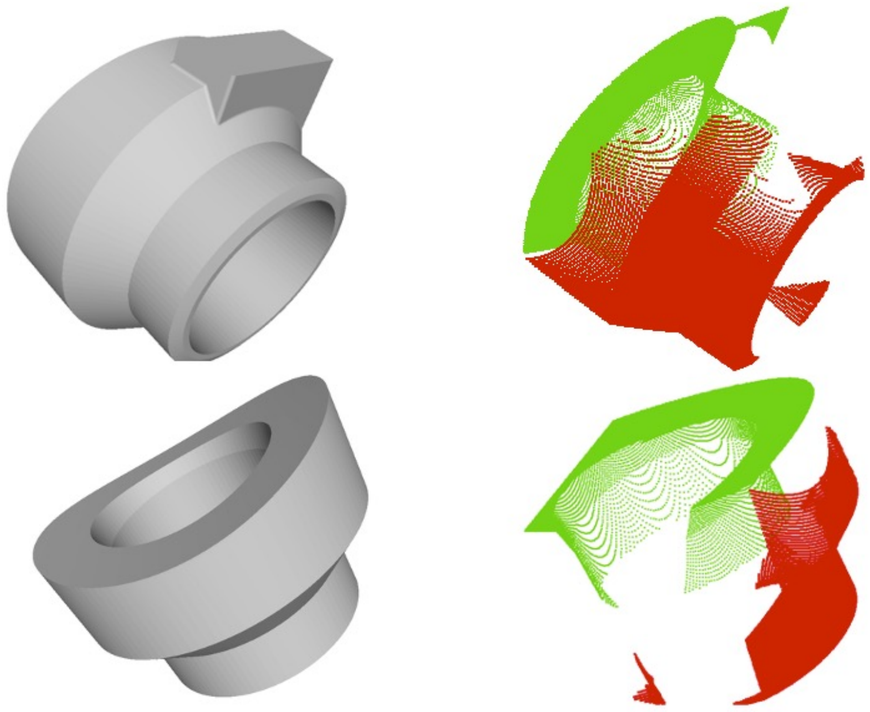
Scanning Simulation Speed Improvement in Robotic Nuclear Decommissioning System

Wonmook Jeong*, Hyoseok Lim, Sungmoon Joo, Ikjune Kim and Jonghawn Lee
Korea Atomic Energy Research Institute, Daedeok-daero 989-111, Yuseong-gu, Daejeon, Korea
**Corresponding author: jwonmook@kaeri.re.kr*

I Introduction

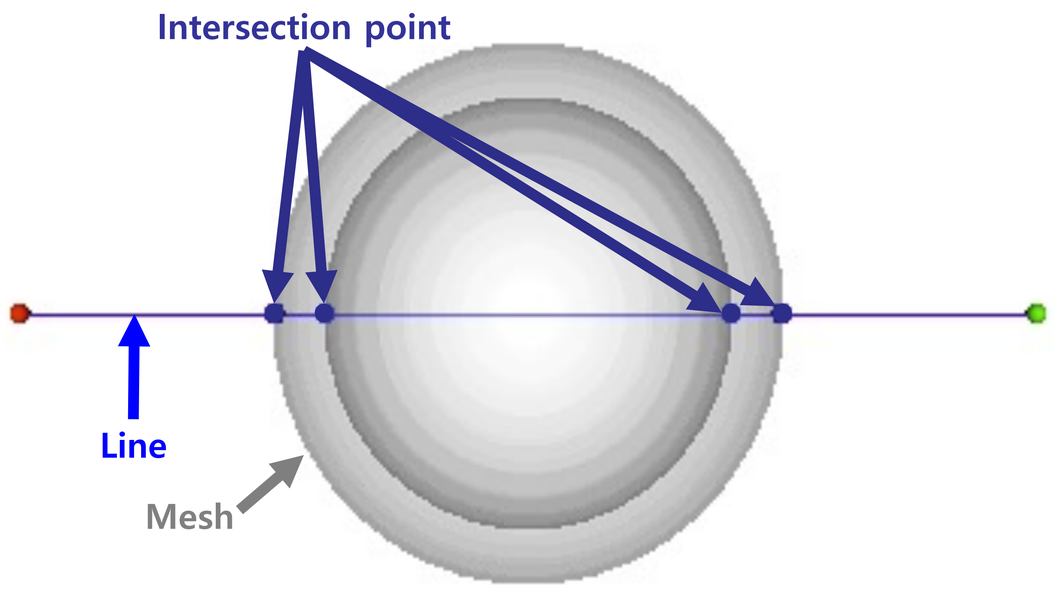
- Simulations generate synthetic point cloud data that is used to train deep learning models for classifying reactor parts.
- One way to obtain point cloud data is to use a scan model in a virtual environment.
- This study was conducted for the purpose of selecting an optimal algorithm for improving the runtime up to real-time scanning.



Original mesh data & Scanning data

II Simulator Development

- Simulators are commonly built by applying a ray-casting mechanism.
- The ray-casting mechanism uses the functions of the pycaster library and the VCG library as shown in the figure below to find the intersection points of a mesh and a line segment.

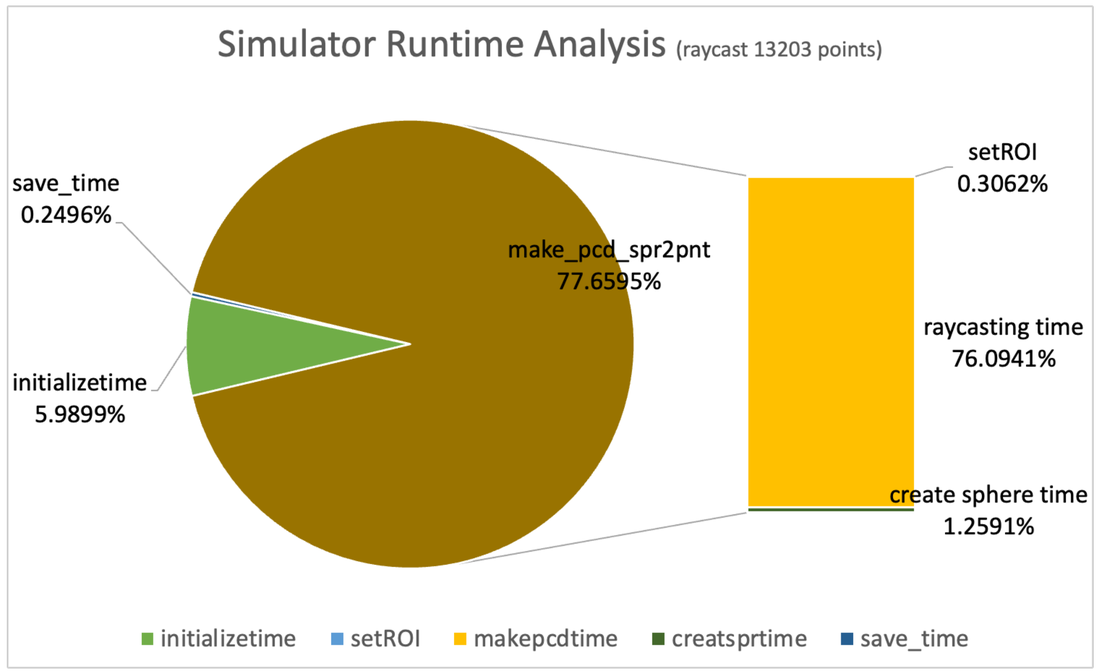


Intersection Point by Ray-casting Method

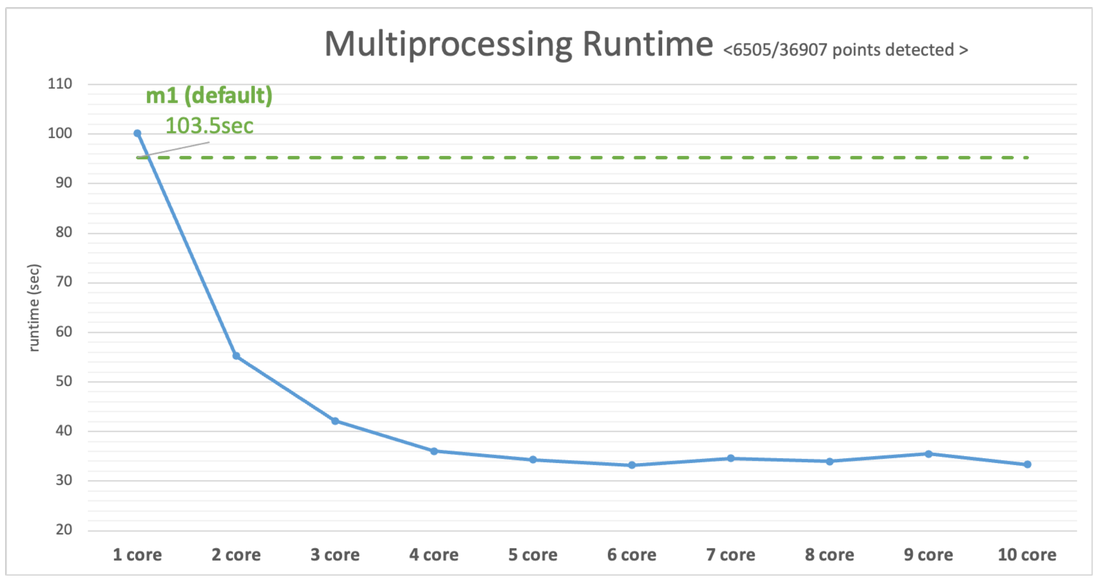
- To achieve optimal performance, we have created a simulator with scripts in various languages.
  - Pure C++(VCG lib) : Using the Find Intersection function, which is one of the functions of VCGlib.
  - Pure Python(pycaster) : Using the Find Intersection function, which is one of the functions of pycaster.
  - Binded Python(pybind11) : Pybind11 is designed to take advantage of Python's user-friendly environment and C++'s fastest computational speed. Pybind11 allows high-speed code written in C++ to work in Python.

III Multiprocessing

- The problem that always accompanies Python-based scripts is that they are slower than C++.
- About 76% of simulators are capable of multiprocessing.
- According to Amdahl's Law, the using 8-core runtime performance improvement is calculated to be approximately 3x.



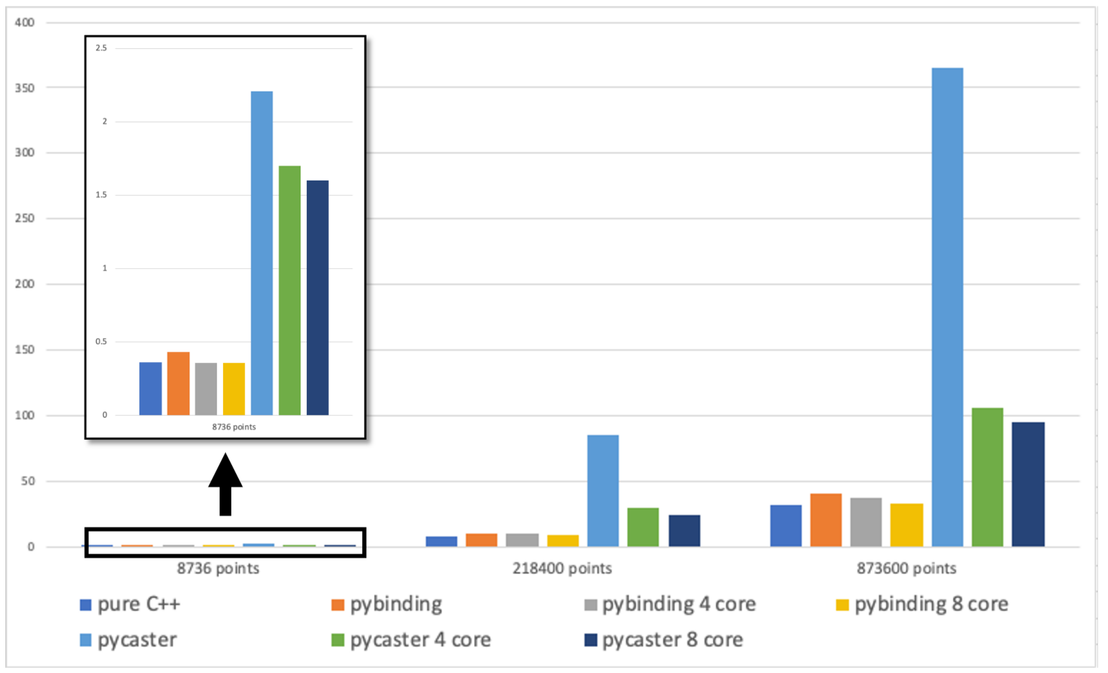
Simulator Runtime Analysis



Multiprocessing Runtime per Core

IV Conclusions

- In almost all cases, pure C++ code performed the best, and the results with Binded Python showed similar performance.
  - Pure C++ : 6~10x time Faster than pure python
  - Binded Python : 5~9x time Faster than pure python
  - Pure Python : About 3x time improvement by multiprocessing



Computation Speed Comparison (sec)