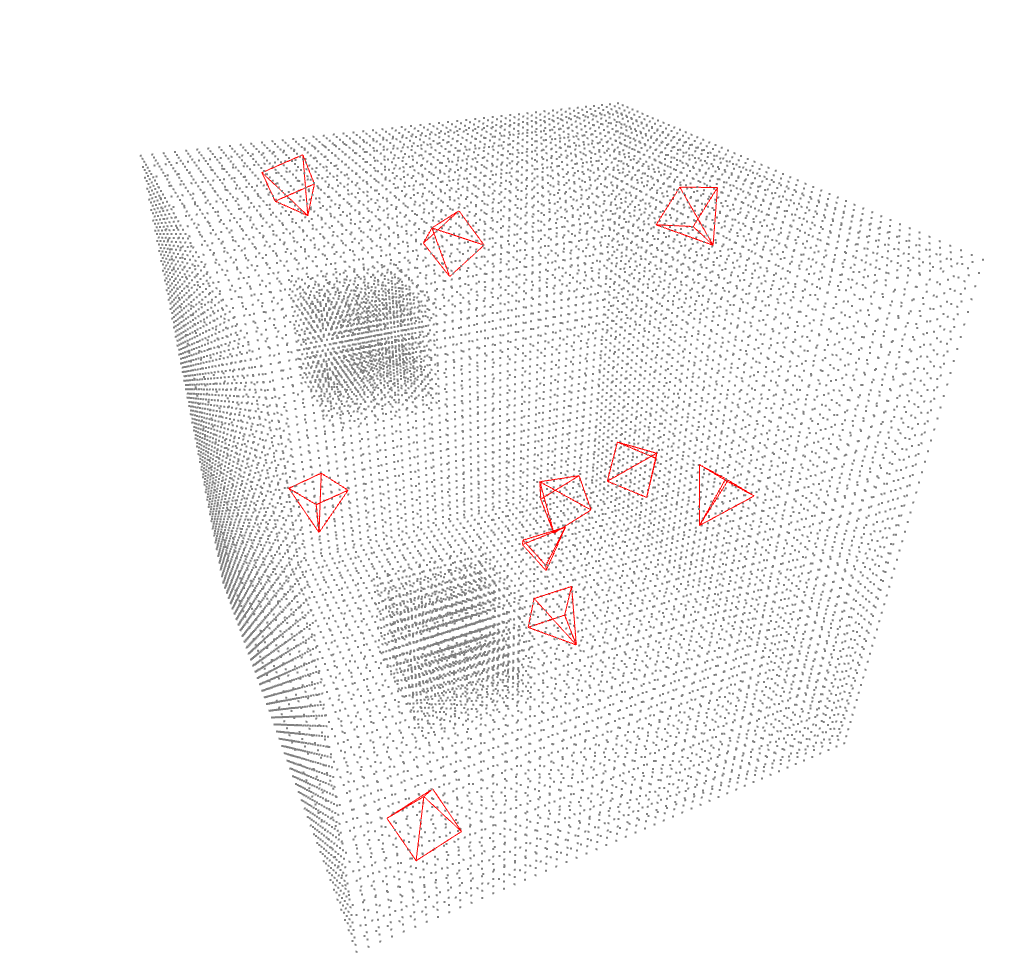
### Technical Case Study: Asset Visual Coverage Analysis

## Introduction

We are inspecting a 3D space for a client. The sensor stack we are using is a camera and lidar combination. The lidar data is used to produce a 3D point cloud as well as a trajectory information. In addition, we have timestamped images that are taken while moving through the space. As we are doing the analytics on the images, corrosion detection for example, we would like to know how much of the space we can see through the images provided.

For this case study, we have produced a mock dataset to test the algorithm on. A 5x5 space with some cubes inside, see the image below. The red pyramids represent the camera positions and viewing angles.

## Objective

Your task is to analyze the dataset containing:

1. A **point cloud** representing the asset in 3D space.
2. A **trajectory file** containing position and orientation data as a function of time.
3. **Timestamped images** captured during the trajectory. For this mock dataset the images are just empty frames, for this exercise the timestamp and camera FOV is enough.

The goal is to write a program that computes the **visual coverage** of the asset. Specifically, determine what percentage of the asset (as represented by the point cloud) is covered by the images based on the trajectory and camera parameters.

The intent is to limit the time spent to **3 hours** (for both the coding and putting a simple document together). It is important to keep yourself to that limit. If you can’t reach the point of a fully functional solution, please write out a short paragraph that outlines any further steps you would have taken to reach the final solution. One of the evaluation criteria, as outlined later in the document, is time management.

Provided Materials

A dataset folder containing:

* asset.pcd: The point cloud file in .pcd format.
* trajectory.csv: A CSV file with columns [timestamp, x, y, z, roll, pitch, yaw].
* A folder images/ with timestamped image files (e.g., image\_0001.jpg).
* camera\_intrinsics.json: A JSON file specifying the camera’s intrinsic parameters, including:
  + fx, fy: Focal lengths.
  + cx, cy: Center pixels
  + hor\_fov, ver\_fov: Horizontal and vertical field of view [degrees]
  + distortion\_coefficients: Lens distortion parameters (optional).

### Assumptions

To simplify the task, you may assume the following:

1. **Point Cloud**:
   * The point cloud is a fixed representation of the asset in a local coordinate frame.
   * All points in the cloud are equally significant for coverage calculation.
2. **Trajectory and Images**:
   * The camera position and orientation relative to the lidar frame can be assumed to be identity. No transformation is required from the trajectory file.
3. **Camera Model**:
   * Use a pinhole camera model for projection.
4. **Coverage Calculation**:
   * A point is considered "covered" if it projects within the boundaries of any image.

### Deliverables

1. **Code Repository**:
   * A Git repository containing:
     + Your implementation in Python
     + A README file explaining how to set up and run your code.
2. **Summary Document**:
   * A short document summarizing:
     + Your approach to the problem.
     + Assumptions you made.
     + The calculated percentage of the asset covered visually.

### Task Requirements

1. **Input Handling**:
   * Load the point cloud, trajectory data, and camera parameters.
   * Read the timestamped images.
2. **Projection and Visibility**:
   * Use the trajectory, camera parameters, and timestamps to project the point cloud into image space.
   * Determine which points from the point cloud are visible in each image.
3. **Coverage Computation**:
   * Calculate the fraction of the asset’s points that are visible in at least one image.
   * Output the result as a percentage.
4. **Efficiency and Clarity**:
   * Write clear, maintainable code.
   * Document your assumptions and limitations.

### Evaluation Criteria

Your submission will be evaluated on:

* **Functionality**: Does the code produce correct and meaningful results?
* **Clarity**: Is the code easy to understand and run? Are assumptions documented?
* **Approach**: Is the methodology sound and appropriate for the problem?
* **Time Management**: Did you scope your solution effectively to fit within the allotted time?

### Suggestions for Success

* Focus on the core functionality (projection, visibility, and coverage calculation).
* Use libraries like Open3D, or NumPy to simplify image and point cloud processing.
* Make reasonable assumptions to handle ambiguities in the data or problem statement.
* Don’t over-optimize; prioritize producing a working solution.
* Keep an eye on the time. If you are close to running out, think about reprioritising the report.

Good luck! We look forward to seeing your approach and insights.