Introduction

Project name: An Efficient Web Application for 3D Point Cloud Visualization and Down-Sampling

Github: https://github.com/try-one-try/3d-project#

Name: CHANG, Ruihe

Email: rchangab@connect.ust.hk

This project is a 3D point cloud visualization web application. It aims to offer intuitive, efficient point cloud processing and interactive visualization. The system uses a front-end/back-end separation architecture. The front end is built with Vue.js, and the back end uses Flask. Together, they let users upload, render, interact with, and down-sample point clouds right in the browser.

System Architecture

The system is built with a clear front-end/back-end split:

- **Front End:** Uses Vue.js and Three.js for a smooth user interface and high-performance 3D rendering.
- **Back End:** Uses Flask and Python libraries to handle point cloud data.
- **Communication:** The two sides talk via a RESTful API, forming a complete web solution for 3D point cloud visualization.

```
frontend/
    ├── public/              # Static assets
    ├── src/                 # Source code
    |   ├── components/      # Vue components
    |   |   ├── DownsamplePly.vue      # Point cloud downsampling component
    |   |   ├── FileUpload.vue         # File upload component
    |   |   └── PointCloudViewer.vue   # 3D visualization component
    |   ├── App.vue          # Main application component
    |   └── main.js          # Application entry point
    ├── package.json         # Dependencies and scripts
    ├── vue.config.js        # Vue configuration
    └── babel.config.js      # Babel configuration
backend/
    ├── app.py               # Main Flask application and API endpoints
    ├── downsample.py        # Point cloud downsampling algorithm
    ├── requirements.txt     # Python dependencies
    └── uploads/             # Directory for uploaded point cloud files
```

Front-end

- **Core Framework:** Vue.js + Vue CLI
- **Language:** JavaScript
- **3D Engine:** Three.js
- **Interaction Controls:** OrbitControls
- **HTTP Requests:** Fetch API, XMLHttpRequest
- **Code Style:** ESLint

Back-end

- **Web Framework:** Flask
- **CORS Support:** Flask-CORS
- **Language:** Python
- **Data Processing:** NumPy
- **Point Cloud Parsing:** PLYFile
- **Logging:** Python's logging module

Technology and Optimizations

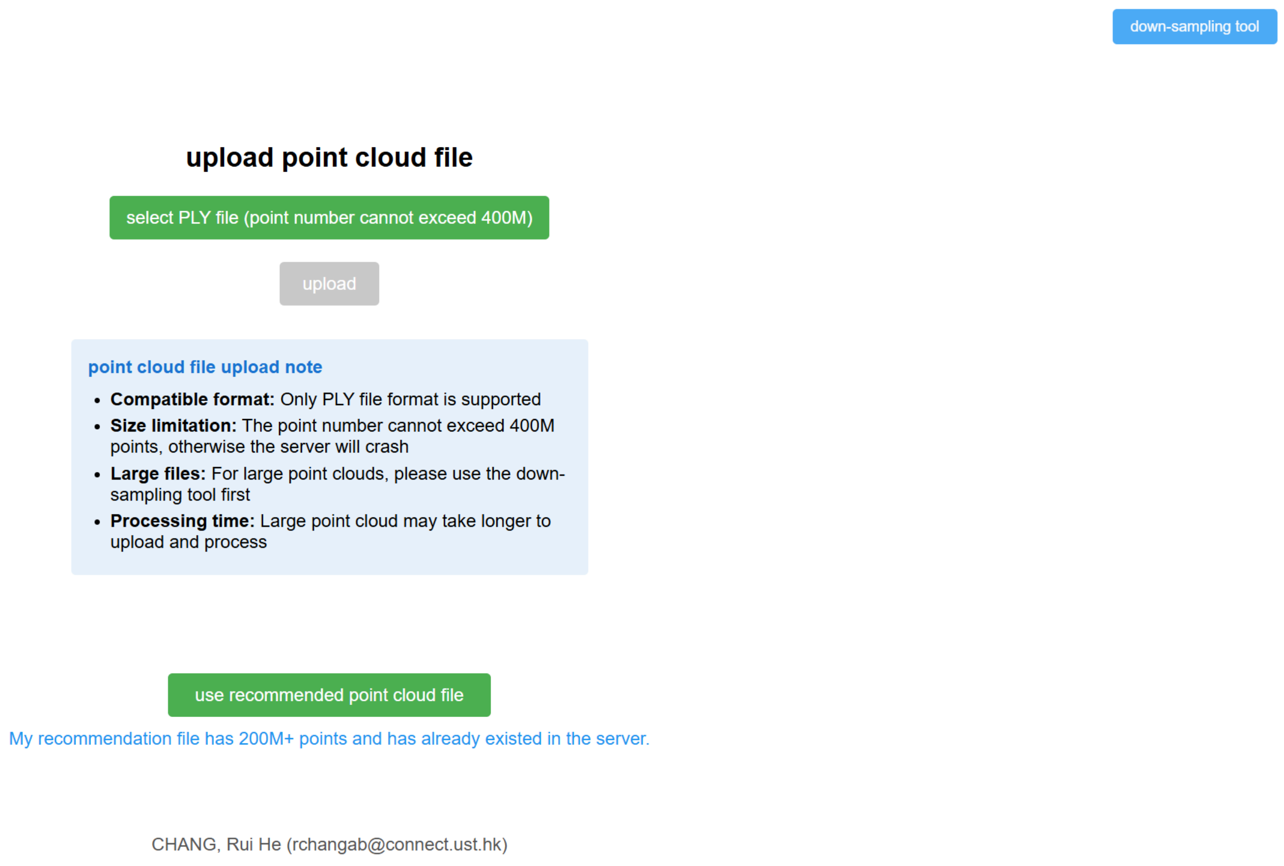
In building this 3D point cloud Visualization web application, I used several techniques to boost performance and user experience:

- **Spatial Hash and Random Sampling**
    - **Spatial Hash Sampling:** For large clouds (>1 million points), split 3D space into equal grid cells and sample each cell. This keeps the cloud's shape and key details.
    - **Random Sampling:** For smaller clouds, this method is fast and gives good visual results.
- **Large File Handling**

    - **Size Limit:** limit file size at 4 million points to protect server resources.
    - **Progress Display:** Users see live upload and processing progress bars for better feedback.
    - **Streaming Processing:** handle data in streams to lower memory use.
- **Rendering Optimizations**
    - **BufferGeometry:** Using Three.js's BufferGeometry to speed up rendering.
    - **High-Performance Mode:** enable WebGL's high-performance settings.
    - **Dynamic Material:** pick the best material based on whether the cloud has color data.
    - **Auto Camera:** The camera position is calculated and set automatically to fit the entire cloud.

Basic Features

Basic Features - Point Cloud Upload


down-sampling tool
select PLY file (point number cannot exceed 400M)
upload
use recommended point cloud file

upload point cloud file

point cloud file upload note

- **Compatible format:** Only PLY file format is supported
- **Size limitation:** The point number cannot exceed 400M points, otherwise the server will crash
- **Large files:** For large point clouds, please use the down-sampling tool first
- **Processing time:** Large point cloud may take longer to upload and process

My recommendation file has 200M+ points and has already existed in the server.

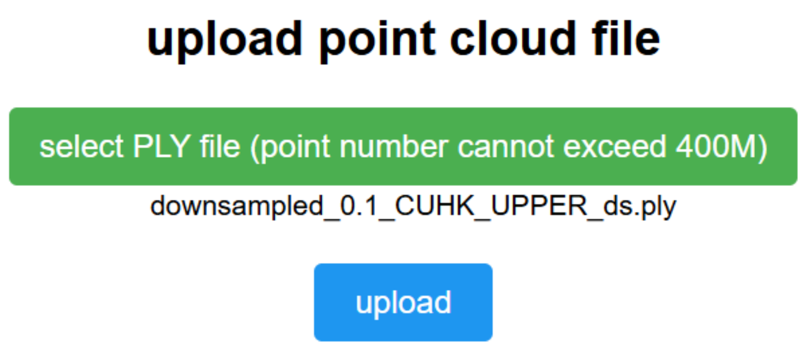
CHANG, Rui He (rchangab@connect.ust.hk)

- Upload ply point cloud file from web browser (frontend) to server (backend)
- API Interface
    - url: `/api/upload`
    - method: POST

Implementation process

Front-end

- Front-end provides a file selection interface through the FileUpload.vue component
- After the user selects a PLY file, upload button will be available



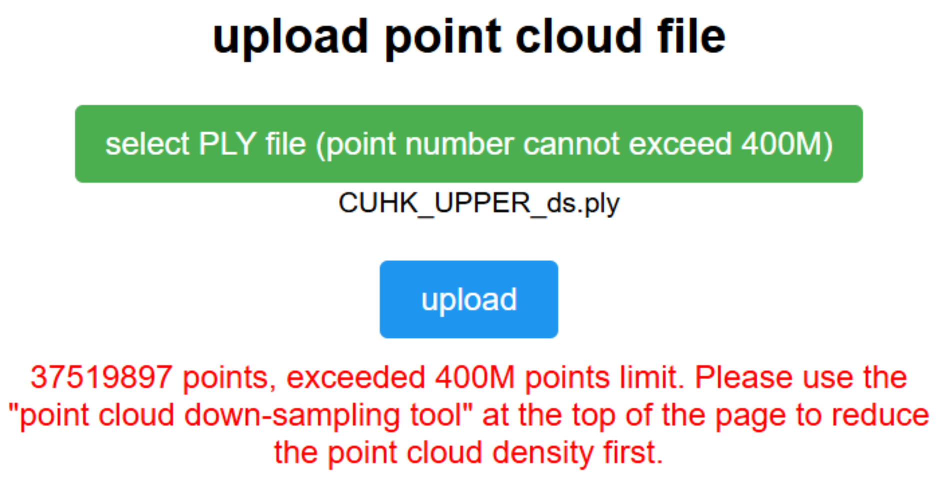
upload point cloud file

select PLY file (point number cannot exceed 400M)
downsampled_0.1_CUHK_UPPER_ds.ply
upload

- After user click upload button, the front-end uses FormData class to send the file to the back-end

Back-end

- backend receives the file and saves it to the uploads directory
- backend uses PlyData library to analyze PLY files and extract point and color information
- backend Check if the number of points exceeds the limit of 4 million points
- If the number of points exceeds, then throw error information to frontend, and frontend will not continue to call rendering function



upload point cloud file

37519897 points, exceeded 400M points limit. Please use the "point cloud down-sampling tool" at the top of the page to reduce the point cloud density first.

- If upload successfully, then backend return success and file details (name, point count, color info)

```python
return jsonify({
    'success': True,
    'filename': filename,
    'total_points': num_points,
    'has_colors': has_colors
})
```

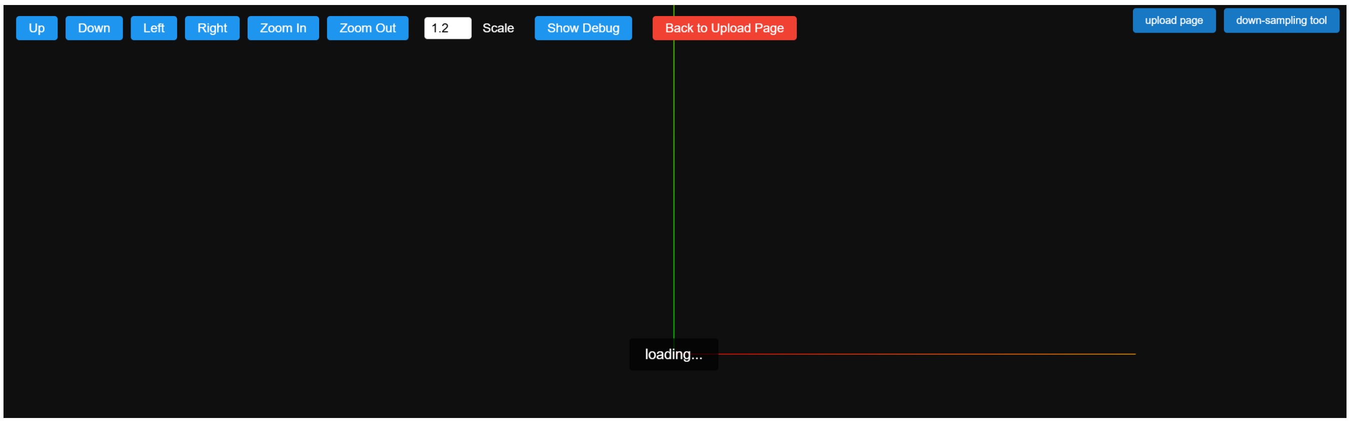
Basic Features - Point Cloud 3D Scene

Switch View from upload page to 3d scene

- When frontend receive upload success from back-end, the `FileUpload` component emits an event.
- The `App` component catches this event and updates its state.
- By watching the state change, the app switches the web view from upload page to 3d scene.

Initialize 3D Scene

- When `PointCloudViewer` mounts, it loads Three.js automatically.
- It creates a scene, camera, axes helper, renderer, and controls.
- It sets up lights and helper tools.



Fetch Point Cloud Data from Backend

- API Interface
  - url: `/api/pointcloud/<filename>`.
  - method: GET
  - function: reads the PLY file, extracts coordinates and color data, and returns the data in JSON format.
- Frontend sends a GET request to back-end with the file name of the uploaded file
- Backend will read the PLY file from uploads file folder and returns the data in JSON format.

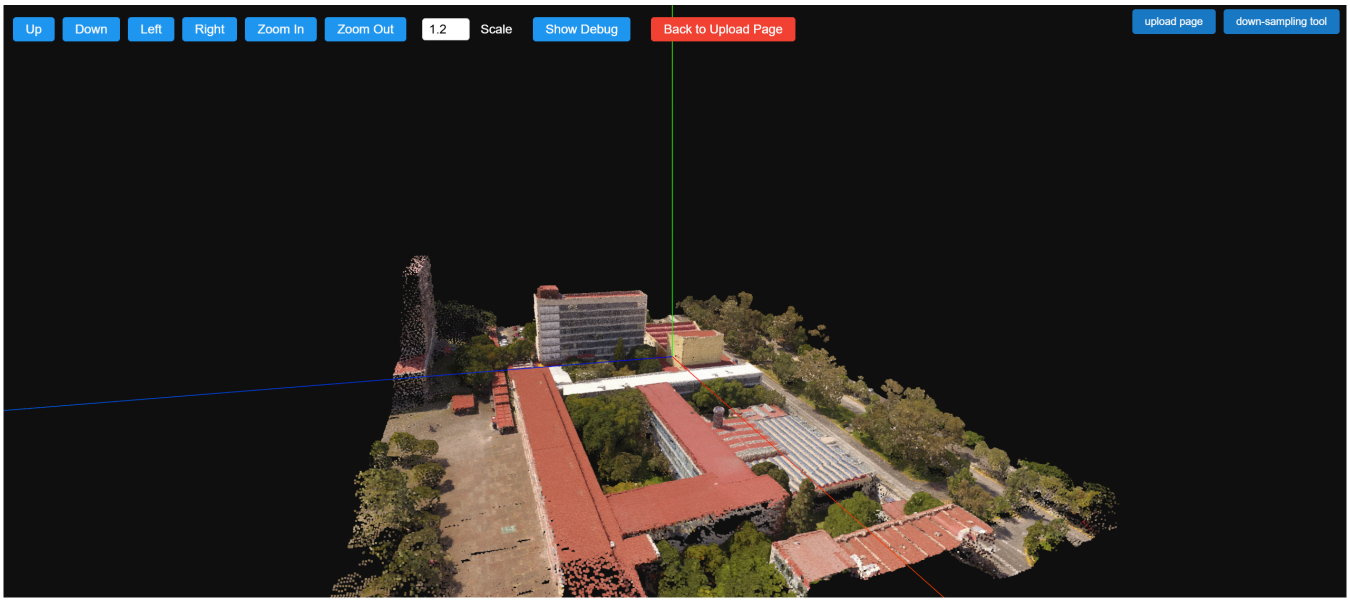
```
result = {
    'file': filename,
    'format': plydata.format,
    'version': plydata.version,
    'elements': elements,
    'vertex_samples': samples if samples else None
}

return jsonify(result)
```

Render the Point Cloud

After getting the point cloud data, the system needs to display it in the 3D scene:

- **Create Geometry**
  - The system makes a 3D geometry to hold all point positions.
  - It sets the X, Y, and Z coordinates from the backend into this geometry.
- **Handle Color Data**
  - If the point cloud has color info, the system reads each point's RGB values.
  - It applies those colors so the cloud shows its original colors.
  - If there's no color data, it uses a default green for all points.
- **Set Point Display Properties**
  - It sets each point's size to 0.01 (or auto-adjusts based on cloud size).
  - It chooses between one uniform color or each point's own color.
- **Create the Point Cloud Object**
  - It combines the geometry and display settings into one point cloud object.
  - It adds this object to the 3D scene for display.

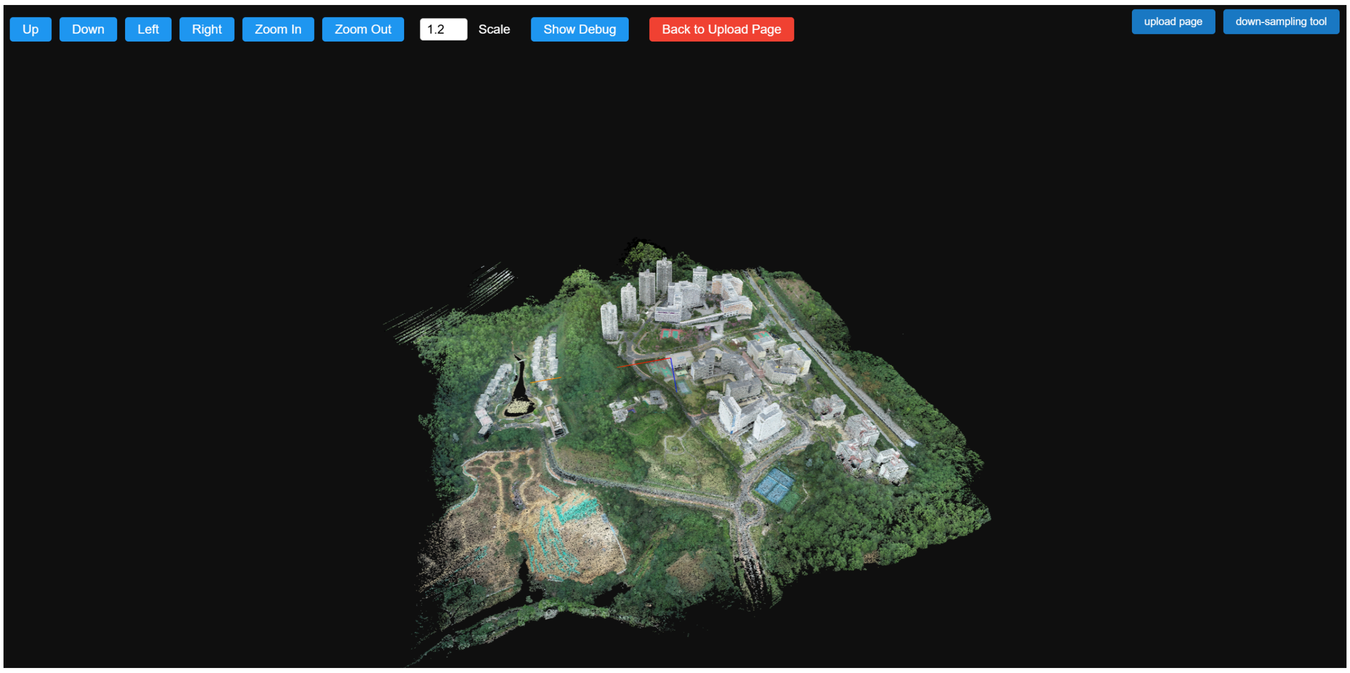


Adjust the View

To make sure users see the whole model, the system auto-adjusts the camera:

- **Compute the Bounding Box**
  - The system finds the smallest cube that contains all points.
  - It determines the cloud's overall size and range.
- **Center the Cloud**
  - It moves the cloud's center to the origin (0, 0, 0).
  - This makes it easy for users to rotate around the model.
- **Auto-Set Camera Distance**

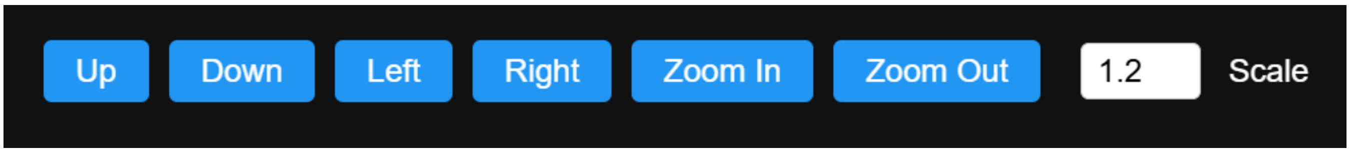
- Based on the cloud size, it calculates a suitable camera position for a clear view.
- It ensures the initial view shows the entire cloud.



User Interaction Controls

The system offers several ways to interact with the 3D point cloud:

- **Mouse Controls**
  - Left-click drag: rotate the model
  - Right-click drag: pan the view
  - Scroll wheel: zoom in and out
- **Touchscreen Controls** (for tablets and phones)
  - One-finger drag: rotate the model
  - Two-finger drag: pan the view
  - Two-finger pinch: zoom in and out
- **UI Button Controls**
  - Arrow buttons: rotate the model up, down, left, or right
  - Zoom buttons: move the camera closer or farther, according to the scale number set



Resource Management

When the user leaves the viewer or closes the page, the system cleans up:

- **Free Memory**
  - Release the memory used by the point cloud data

- Remove any 3D objects no longer needed
- **Remove Event Listeners**
    - Stop listening for window resize events
    - Remove mouse and touch event listeners
- **Remove Page Elements**
    - Delete the 3D canvas or rendering area from the page
    - Make sure no extra elements are left behind

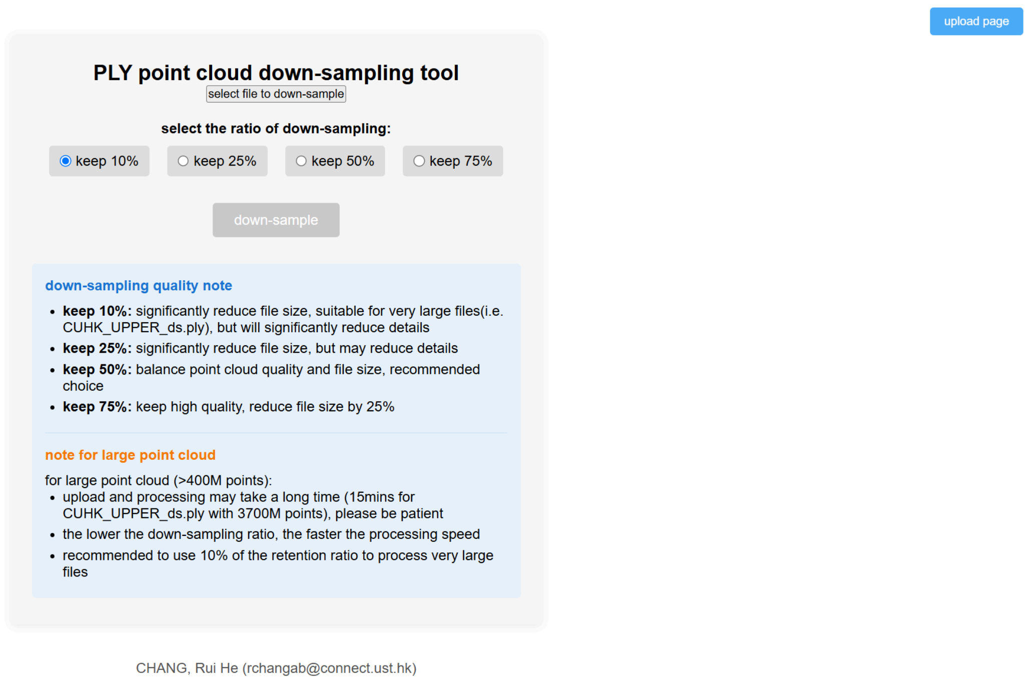
Advanced Features - Down-Sampling

Necessary Down-Sampling for web browser

Point cloud down-sampling is a key feature, especially for very large files.

According to my experiment, when a point cloud has too many points (e.g., over 4 million), not only upload and processing will slow down and 3D rendering suffers, but also the web browser will crash.

Therefore, Down-sampling cuts the point count to keep the overall shape while greatly reducing file size and speeding up processing.



PLY point cloud down-sampling tool

select the ratio of down-sampling:

(●) keep 10% ( ) keep 25% ( ) keep 50% ( ) keep 75%

down-sampling quality note

- **keep 10%:** significantly reduce file size, suitable for very large files(i.e. CUHK_UPPER_ds.ply), but will significantly reduce details
- **keep 25%:** significantly reduce file size, but may reduce details
- **keep 50%:** balance point cloud quality and file size, recommended choice
- **keep 75%:** keep high quality, reduce file size by 25%

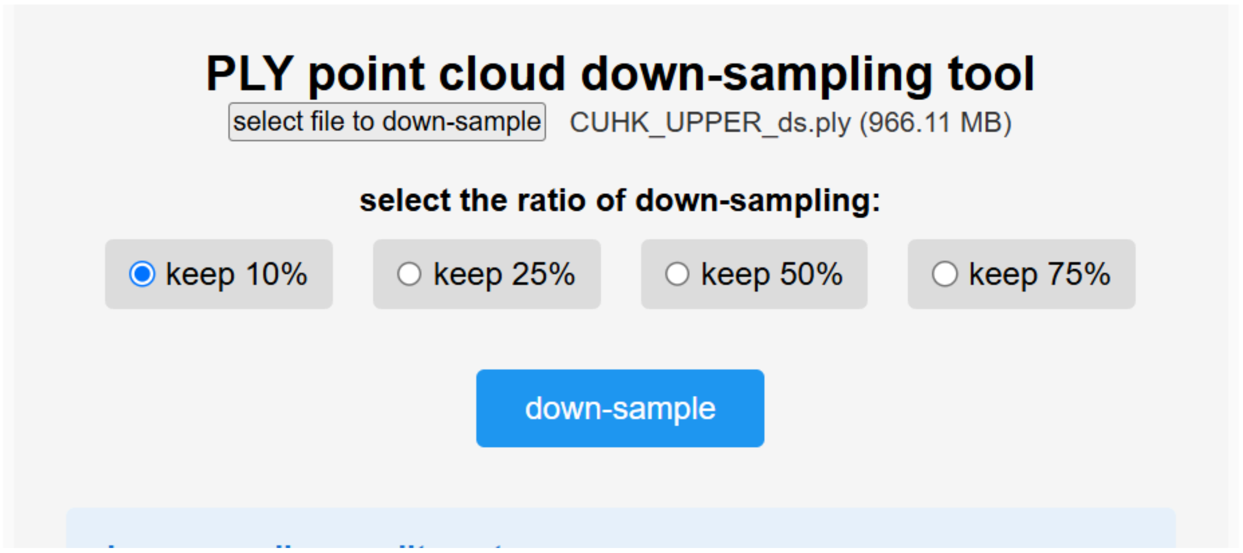
note for large point cloud

for large point cloud (>400M points):
- upload and processing may take a long time (15mins for CUHK_UPPER_ds.ply with 3700M points), please be patient
- the lower the down-sampling ratio, the faster the processing speed
- recommended to use 10% of the retention ratio to process very large files

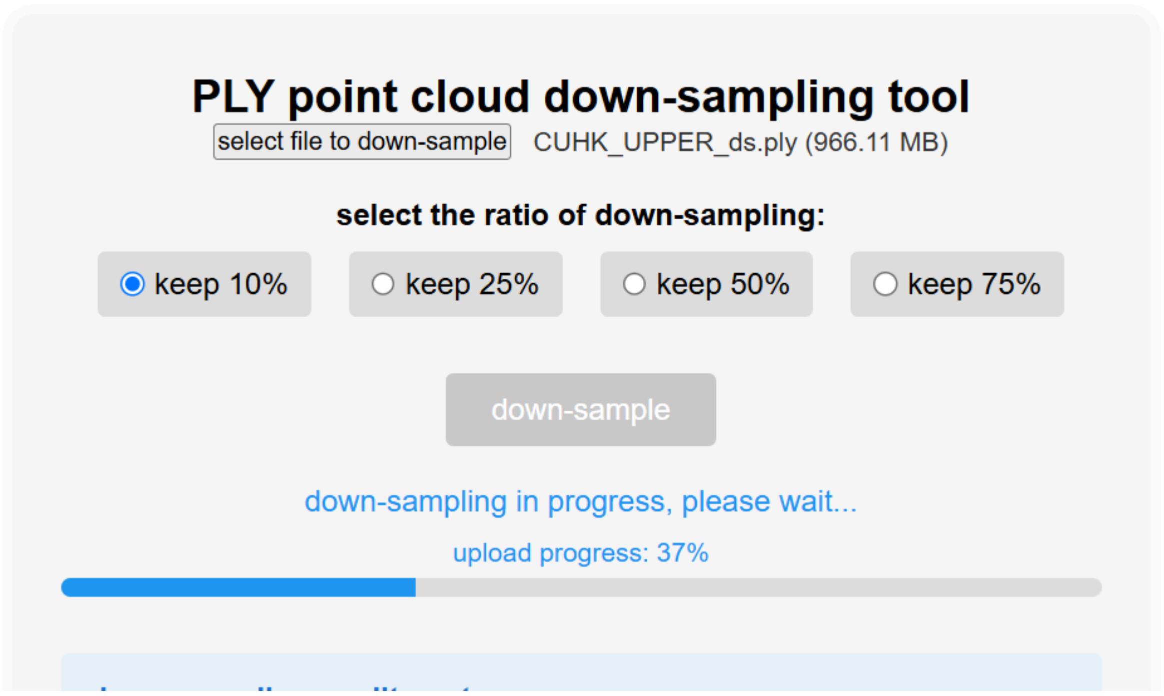
CHANG, Rui He (rchangab@connect.ust.hk)

Feature Description

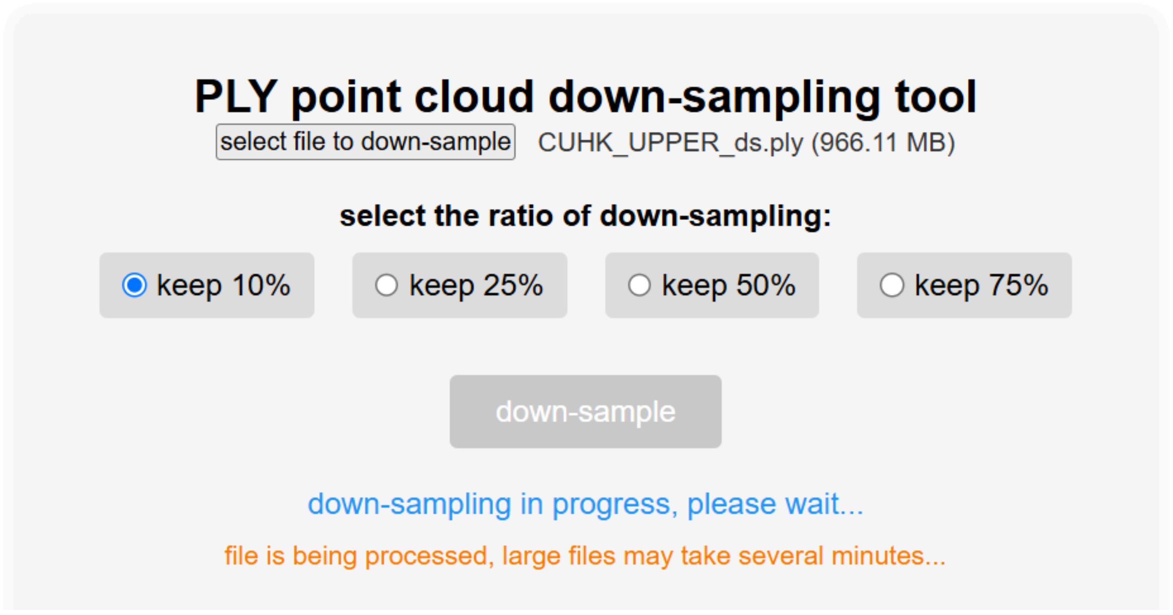
- **File Selection**
    - The user clicks the **"Select File to Down-sample"** button
    - A file picker opens and the user chooses a PLY point cloud file
    - The system shows the chosen file's name and size


PLY point cloud down-sampling tool
select file to down-sample
CUHK_UPPER_ds.ply (966.11 MB)
select the ratio of down-sampling:
keep 10%
keep 25%
keep 50%
keep 75%
down-sample

- **Choose Down-Sampling Ratio** The user picks a keep rate:
  - **keep 10%:** significantly reduce file size, suitable for very large files(i.e. CUHK_UPPER_ds.ply), but will significantly reduce details
  - **keep 25%:** significantly reduce file size, but may reduce details
  - **keep 50%:** balance point cloud quality and file size, recommended choice
  - **keep 75%:** keep high quality, reduce file size by 25%
- **Start Upload & Down-Sampling**
  - The user clicks the **"Down-sample"** button
  - The file uploads to the server, with a progress bar shown


PLY point cloud down-sampling tool
select file to down-sample
CUHK_UPPER_ds.ply (966.11 MB)
select the ratio of down-sampling:
keep 10%
keep 25%
keep 50%
keep 75%
down-sample
down-sampling in progress, please wait...
upload progress: 37%

  - After upload, the server runs the down-sampling
  - Down-sampling may take a long time (15mins for CUHK_UPPER_ds.ply with 3700M points)


PLY point cloud down-sampling tool

select file to down-sample CUHK_UPPER_ds.ply (966.11 MB)

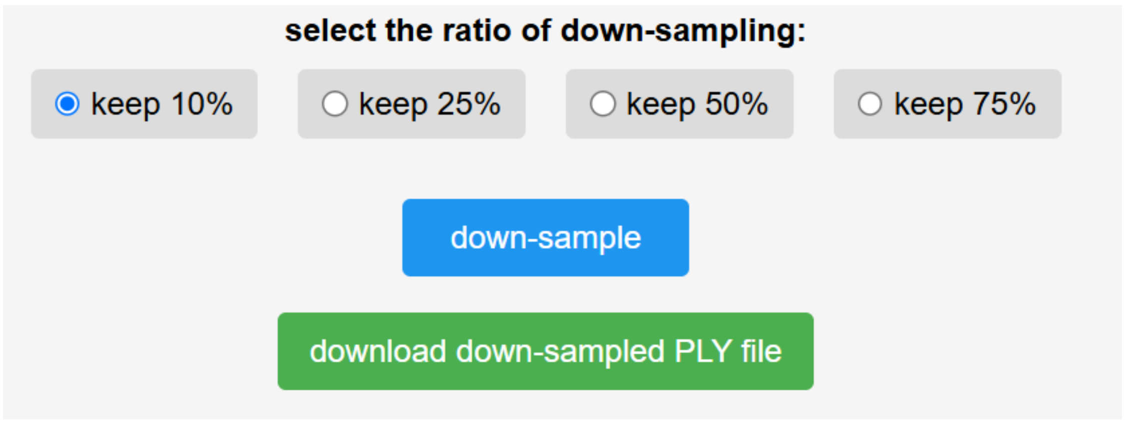
select the ratio of down-sampling:

[✓ keep 10%] keep 25% keep 50% keep 75%

down-sampling in progress, please wait...

file is being processed, large files may take several minutes...

- **File Download**
    - When down-sampling finishes, a download dialog appears with a **Download** button


select the ratio of down-sampling:

[✓ keep 10%] keep 25% keep 50% keep 75%

Feature Implementation

Front-End Implementation Details

The front end component, DownsamplePly.vue, handles user interaction and file upload:

- **Upload Tracking**
    - Use `XMLHttpRequest` instead of `fetch` so user can track upload progress in real time.
    - A progress bar shows upload status, which is important for large files.
- **Post-Upload Handling**
    - After the file uploads and down-sampling finishes, automatically creates a download link so the user can save the processed file.

Back-End Processing Flow

When the server gets a down-sample request, it does the following:

- **Receive and Validate**
    - Accept the uploaded PLY file and the down-sample ratio.

    - Check that the file format and parameters are valid.
- **Run Down-Sampling Algorithm**
    - Use Python's `PlyData` library to read the PLY file.
    - Calculate how many points to keep (original count × ratio).
- **Choose down-Sampling Strategy by Size**
    - For large clouds (over 1 million points): use **spatial-hash down-sampling**.
    - For small clouds: use **random sampling**.
- **Create and Return New File**
    - Extract all needed attributes (coordinates, colors, etc.) for the sampled points.
    - Save the down-sampled points into a new PLY file and send it back to the front end.

Down-Sampling Strategy

Spatial Hash Down-Sampling

Large point clouds (over 1 million points) often represent complex 3D shapes and usually have uneven point density. In simple random sampling, every point has the same chance to be picked, which can make small but important areas (like building edges) vanish after down-sampling. The spatial hash method ensures each region has points, preserving the cloud's overall structure and key features.

- **Core Idea**
    - Divide 3D space into an even grid of cells. From each non-empty cell, pick points in proportion to the desired down-sample rate. This keeps the spatial layout and avoids losing critical details.
- **Advantages**
    - **Preserves Structure:** By sampling each area, it keeps the cloud's overall shape and local details.
    - **Uniform Coverage:** Prevents over- or under-sampling in any region.
    - **Adaptive:** Maintains relative density in both sparse and dense areas.

Random Down-Sampling

Small point clouds (less than 1 million points) usually show simpler shapes with fairly even point distribution, so random sampling rarely loses essential features.

- **Core Idea**
    - Randomly pick the needed percentage of points from the original cloud to form the new one. It's simple and works well when point count is low and distribution is uniform.
- **Advantages**
    - **Fast:** No complex spatial grids needed, so it runs quickly.
    - **Low Memory:** Doesn't require extra data structures.