End Semester Evaluation Report

Point Cloud Segmentation Benchmark for College Campus

GitHub Repo: https://github.com/MJTheGreat3/Point-Cloud-Segmentation

Mathew Joseph

Roll No.: IMT2023008

Project Duration: August – December 2025

Academic Year 2025–26

Abstract

The project aimed to design, implement, and evaluate a scalable and reproducible semantic segmentation pipeline for a large-scale LiDAR point cloud covering the entire college campus.

The work involved registration of multiple zonal LiDAR scans, extensive preprocessing to handle noise and density imbalance, GPU-aware spatial partitioning, and semantic segmentation using Kernel Point Convolution (KPConv) via Open3D-ML. A significant portion of the project was dedicated to addressing real-world engineering challenges such as memory constraints, annotation scalability, and model generalization.

Multiple experimental strategies were explored, including inference using pre-trained models and custom training using manually refined campus data. While pre-trained models demonstrated strong transferability, custom-trained models failed to outperform them, revealing critical data-centric limitations. This report documents the complete pipeline, experimental findings, observed failure modes, and concludes with recommendations for future work emphasizing model-centric improvements over brute-force manual refinement.

Contents

Abstract i

1 Objective 1

2 Goals 1

3 Project Pipeline 1

3.1 Preprocessing and Registration . . . 2
3.2 Noise Filtering and Downsampling . . . 2
3.3 Partitioning Strategy . . . 2
3.4 Semi-Automated Segmentation . . . 3
3.5 Final Tiling and Representation . . . 3
3.6 Benchmarking and Evaluation . . . 3
3.7 Integrated Pipeline Summary . . . 3

4 Progress Till Mid-Semester 5

4.1 Literature Review and Tool Familiarization . . . 5
4.2 Registration and Early Preprocessing . . . 5
4.3 Memory Constraints and Pipeline Redesign . . . 5
4.4 Initial Segmentation Experiments . . . 6

5 Work After Mid-Semester 6

5.1 Pre-trained Model Inference . . . 6
5.2 Custom Training and Evaluation . . . 8
5.3 Attempted Dataset Expansion . . . 8

6 Final Project Directory Structure 9

6.1 Directory Organization and Purpose . . . 9

7 Observations and Recommendations 11

7.1 Observed Failure Modes . . . 11
7.2 Recommendations . . . 12

1 Objective

The primary objective of this project was to build a standardized benchmark dataset and processing pipeline for semantic segmentation of a large-scale LiDAR point cloud representing the college campus of IIIT Bangalore.

Specifically, the project sought to:

- Register and merge multiple zonal LiDAR scans into a unified spatial reference frame.
- Design preprocessing techniques that remove noise and redundancy while preserving critical geometric features.
- Establish a robust semantic segmentation baseline using state-of-the-art models such as KPConv.
- Identify real-world failure modes when deploying segmentation models on complex campus scenes.
- Create a reproducible workflow that can be extended for future dataset expansion and benchmarking.

This objective required balancing algorithmic rigor with practical constraints such as limited GPU memory, long processing times, and the high cost of manual annotation.

2 Goals

The project goals were defined to operationalize the above objective:

- Combine zonal LiDAR tiles into a unified campus-scale point cloud with accurate spatial alignment.
- Apply statistical filtering and voxel downsampling to ensure computational feasibility without excessive loss of detail.
- Experiment with modern deep learning architectures for point cloud segmentation, with KPConv as the primary baseline.
- Develop a modular and crash-resilient pipeline capable of processing hundreds of millions of points.
- Generate documentation, statistics, and code artifacts suitable for use as a benchmark.

These goals evolved iteratively as new constraints and insights emerged during implementation.

3 Project Pipeline

This section describes the complete end-to-end processing pipeline developed for the project, covering data ingestion, preprocessing, segmentation, refinement, and benchmarking. The

pipeline is designed as a modular sequence of stages, where each stage produces well-defined outputs that serve as inputs to subsequent stages.

3.1 Preprocessing and Registration

The raw dataset consists of multiple zonal LiDAR scans acquired independently over different regions of the campus. These scans exist in partially overlapping spatial frames and therefore require accurate registration before downstream processing can be performed.

Registration is carried out using CloudCompare through a two-stage approach. First, coarse alignment is achieved by manually identifying corresponding geometric features such as building corners, road edges, and vertical structures across overlapping tiles. This is followed by fine alignment using the Iterative Closest Point (ICP) algorithm to minimize residual errors between overlapping regions.

Registration quality is verified qualitatively by inspecting continuity across structural elements, including façade edges, poles, and ground surfaces. The output of this stage is a unified campus-scale point cloud that serves as the base input for all subsequent processing steps.

3.2 Noise Filtering and Downsampling

Raw LiDAR data contains isolated noise points and significant density variations arising from scan geometry and overlap between tiles. To address this, statistical outlier removal is applied using neighborhood-based filtering to eliminate points with anomalous local distributions.

Following noise removal, voxel-based downsampling is performed to enforce density uniformity and reduce computational overhead. The voxel size is selected empirically to balance geometric detail preservation against memory and runtime constraints. While downsampling reduces per-tile file sizes substantially, spatial overlaps across tiles motivate further restructuring of the dataset.

3.3 Partitioning Strategy

Processing the unified campus point cloud as a single entity exceeds available RAM and GPU memory resources. Even inference-only execution using pre-trained segmentation models is not feasible under such conditions.

To ensure stable execution, spatial partitioning is introduced as a core design principle. Using a custom partitioning pipeline, the campus point cloud is subdivided into smaller, spatially contiguous partitions, each constrained to approximately 1 GB or less in file size. This results in a total of 108 partitions, each of which can be processed independently.

Partitioning enables stable execution of preprocessing, inference, and training workflows, and allows incremental processing and crash recovery without compromising pipeline integrity.

3.4 Semi-Automated Segmentation

Semantic segmentation is performed using the Kernel Point Convolution (KPConv) architecture implemented through the Open3D-ML framework. As an initial baseline, a pre-trained KPConv model trained on the Paris–Lille dataset is used to perform inference on the campus partitions.

This stage follows a semi-automated paradigm, combining automated inference with selective manual inspection. The pre-trained model generates preliminary semantic labels, which are then visually examined to identify systematic misclassification patterns and regions requiring refinement.

This approach enables rapid generation of labeled data while minimizing the extent of manual annotation required.

3.5 Final Tiling and Representation

The ground-truth point cloud is then spatially re-partitioned into smaller, structured tiles. This tiling is performed purely for representational and computational purposes, ensuring compatibility with model input constraints such as those imposed by KPConv. The tiling strategy preserves spatial coherence and label consistency while enabling flexible reuse of the same dataset across inference, training, and evaluation pipelines.

This representation-centric tiling allows the dataset to scale efficiently and supports systematic experimentation without altering the underlying semantic annotations.

3.6 Benchmarking and Evaluation

Benchmarking is conducted by leveraging the finalized ground-truth campus dataset to evaluate the performance of existing and future semantic segmentation models.

This benchmarking framework establishes a foundation for systematic comparison of segmentation models and positions the campus dataset as a reusable testbed for future research and model development.

3.7 Integrated Pipeline Summary

Figure 1 presents a consolidated view of the complete processing pipeline. The pipeline integrates preprocessing, partitioning, semi-automated segmentation, refinement, and benchmarking into a unified workflow.

This integrated design represents the final operational pipeline and serves as a reproducible foundation for future extensions of the campus-scale point cloud segmentation benchmark.

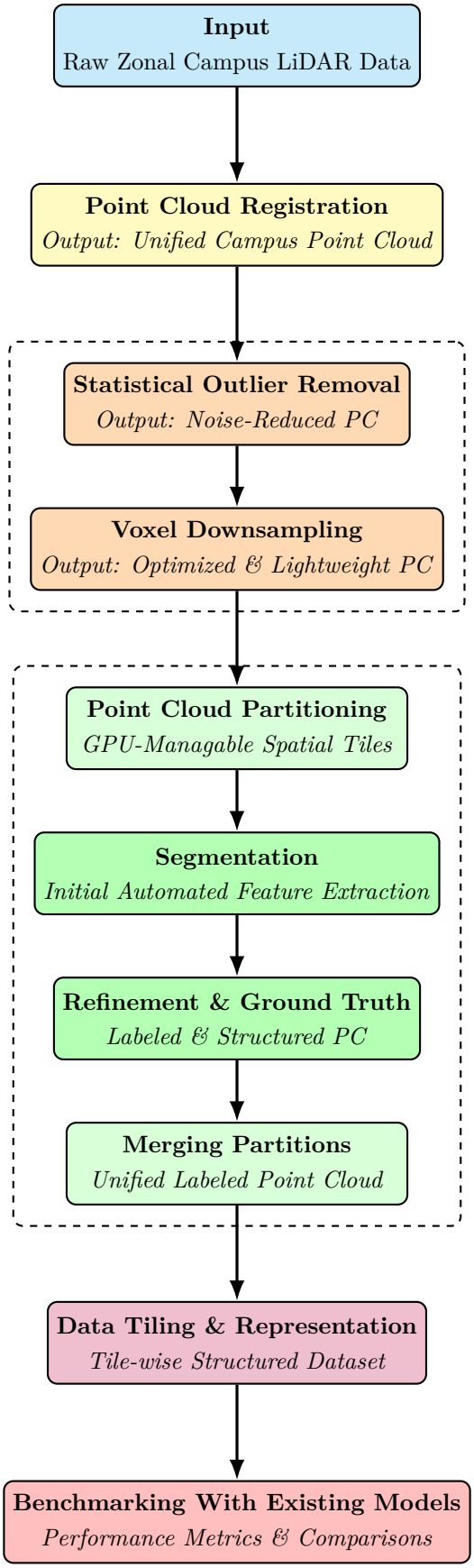


Figure 1: Modified end-to-end pipeline for campus-scale point cloud segmentation

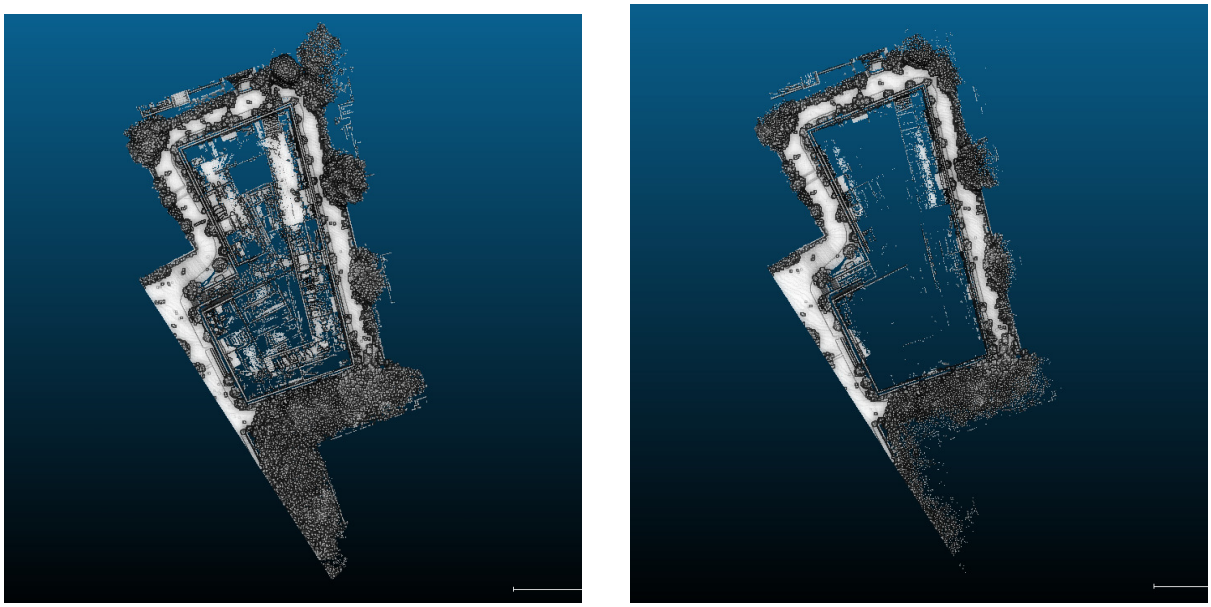
4 Progress Till Mid-Semester

4.1 Literature Review and Tool Familiarization

Initial phases involved studying LiDAR segmentation literature, including datasets such as DALES, to understand common class definitions and feature representations. Simultaneously, CloudCompare was used extensively to gain proficiency in manual point cloud operations.

4.2 Registration and Early Preprocessing

Manual registration of zonal tiles was carried out, followed by preliminary downsampling and statistical analysis of point distributions. Early automation attempts revealed severe memory bottlenecks when processing merged datasets nearing 80GB in size.



(a) Before Preprocessing (b) After Preprocessing

Figure 2: Comparison of a tile before and after preprocessing

4.3 Memory Constraints and Pipeline Redesign

Repeated crashes during Jupyter-based execution highlighted memory as a dominant constraint. This led to exploration of PDAL and chunk-based processing approaches, ultimately motivating a complete redesign of the preprocessing workflow.

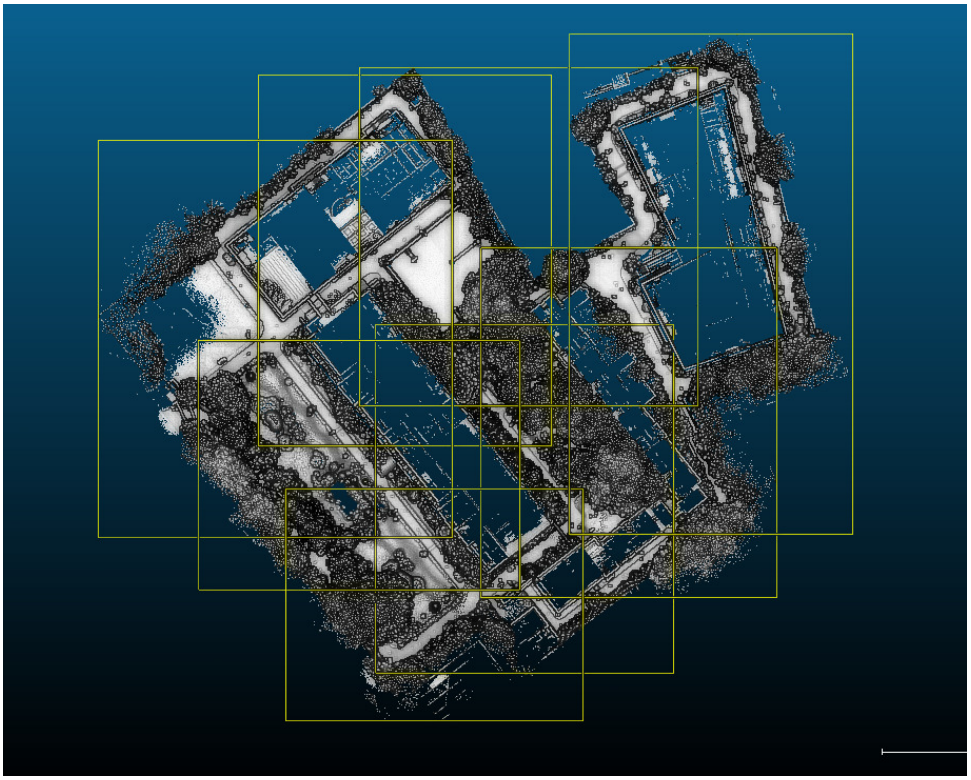


Figure 3: Partitioning of the dataset into tiles

4.4 Initial Segmentation Experiments

KPConv was selected as the baseline segmentation model. The pipeline was first validated on the Paris–Lille dataset to ensure training and inference stability before deployment on campus data.

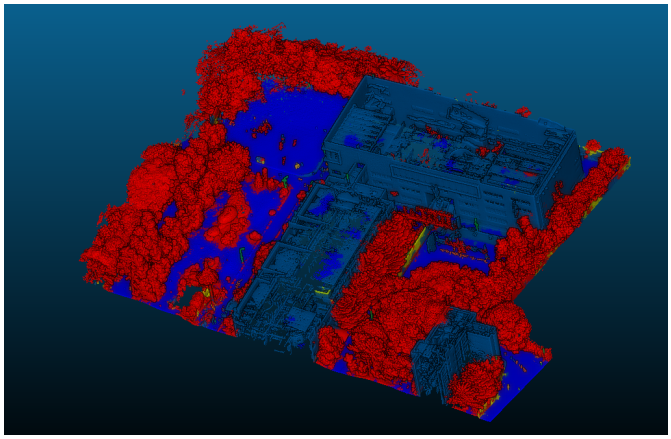
5 Work After Mid-Semester

After mid-semester, inference was attempted directly on seven manually partitioned campus blocks using a pre-trained Paris–Lille KPConv model. Even inference-only runs resulted in system crashes due to excessive file sizes.

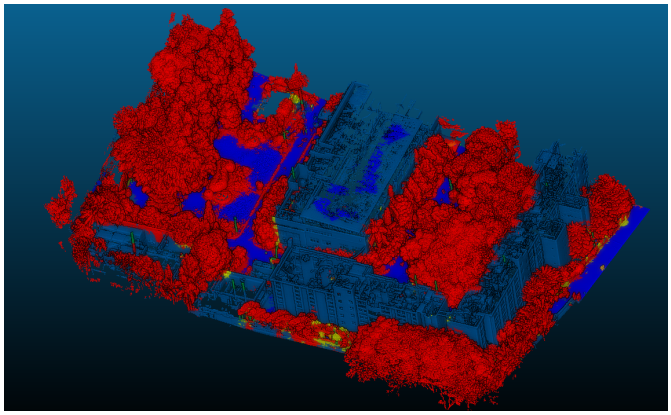
To address this, a dedicated partitioning pipeline was implemented, resulting in **108** partitions of approximately **1 GB or less** each. This enabled stable and fast inference as well as feasible model training.

5.1 Pre-trained Model Inference

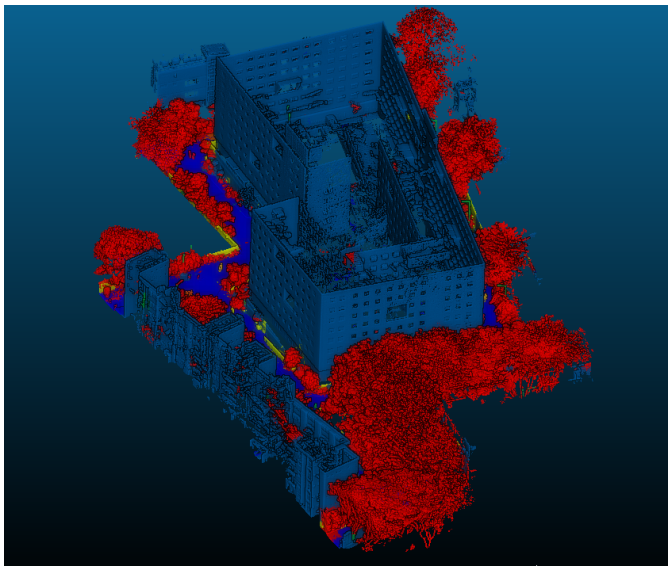
Inference using the pre-trained Paris–Lille model on the first ten partitions yielded surprisingly strong qualitative results. These partitions were merged and manually refined to reduce visible errors.



(a) Campus Block A



(b) Campus Block B



(c) Campus Block C

Figure 4: Semantic segmentation results on campus blocks using the pre-trained Paris–Lille KPConv model

5.2 Custom Training and Evaluation

The refined data was split into three subsets and used to train a custom KPConv model. However, inference using this model on unseen partitions produced worse results than the pre-trained baseline. Increasing training epochs and tuning parameters failed to yield consistent improvements.

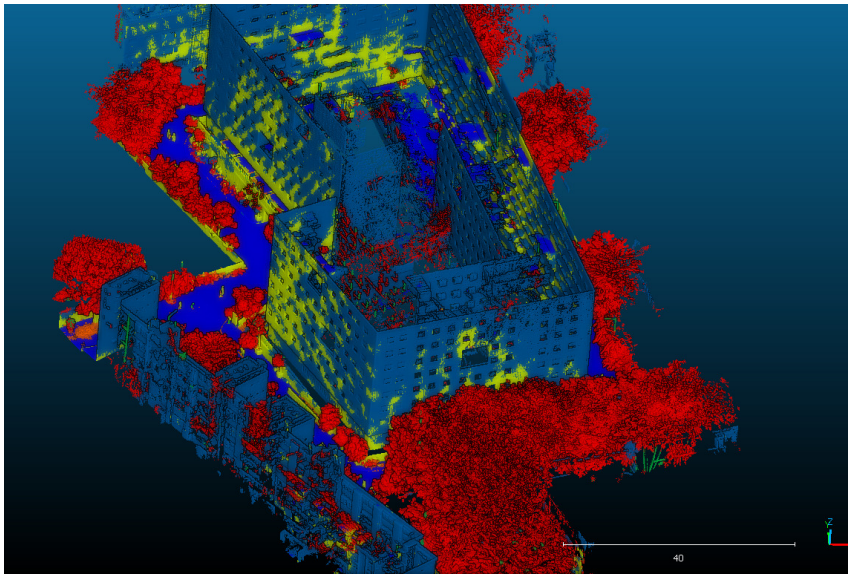


Figure 5: Semantic segmentation results on campus blocks using the custom-trained KPConv model, illustrating degraded performance relative to the pre-trained Paris–Lille baseline

5.3 Attempted Dataset Expansion

To address suspected data scarcity, inference was performed on the next ten partitions using the pre-trained model, followed by attempted manual refinement. These partitions exhibited significantly higher error rates, making refinement far more time-consuming. Only approximately **15%** of the dataset could be refined, compared to the planned 40%.

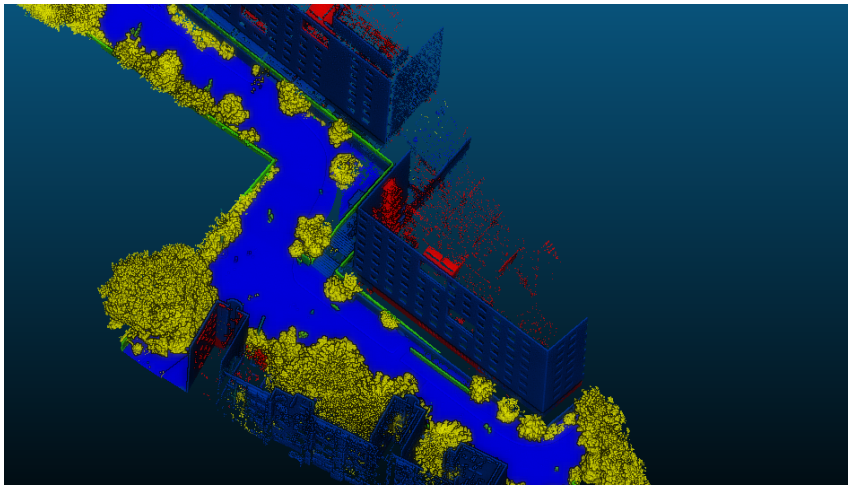


Figure 6: Semantic segmentation result after manual refinement on approximately 10% of the campus point cloud.

6 Final Project Directory Structure

This section documents the complete directory structure of the final project as organized on the campus workstation. The structure reflects the logical separation of environment setup, preprocessing, segmentation, experimentation, logs, and backups, and is designed to ensure reproducibility, traceability, and ease of extension.

6.1 Directory Organization and Purpose

- **env.yaml** defines the Conda environment used across all experiments, ensuring reproducibility of library versions, CUDA compatibility, and Open3D-ML dependencies.
- **Preprocessed Campus** contains the fully registered and cleaned campus point cloud in LAS/LAZ format, serving as the canonical raw input to the segmentation pipeline.
- **Preprocess** includes notebooks that implement cumulative and divisive preprocessing strategies explored during early pipeline design.
- **Segmentation** contains all components related to partitioning, inference, training, refinement, and benchmarking:
  - **Partition Data** stores scripts for grouping, renaming, and summarizing partitioned point cloud files.
  - **Manual Refinement** contains partially refined ground-truth subsets used for custom training experiments.
  - **Open3D-ML** stores the KPConv framework, configuration files, and datasets used for training and inference.
  - **logs** records training logs and model checkpoints.
  - **.ply files** represent intermediate and final campus blocks used for inference, visualization, and refinement.
  - **KPConv_*.ipynb notebooks** implement training and inference workflows for both pre-trained and custom models.
- **Backup Directory (Drive F:)** contains archived inference results and grouped campus partitions, ensuring data redundancy and experiment traceability.

This directory structure reflects the final, stable organization of the project and provides a clear mapping between pipeline stages, code artifacts, and generated data products.

Final File Structure

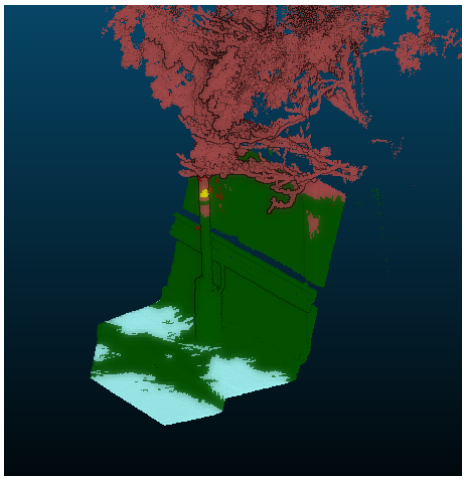
```
.
├── E:
│   └── PE
│       └── Mathew - 238 GB
│           ├── env.yaml
│           ├── Preprocessed Campus
│           │   └── FullCampus.laz
│           └── Main
│               ├── Preprocess
│               │   ├── cumulative_pipeline.ipynb
│               │   └── divisive_pipeline.ipynb
│               └── Segmentation
│                   ├── Partition Data
│                   │   ├── file_grouper.ipynb
│                   │   ├── remove_fields.ipynb
│                   │   ├── rename_file.ipynb
│                   │   └── Partition Summary
│                   ├── Manual Refinement
│                   │   ├── simplify.ipynb
│                   │   ├── A1-A10.ply
│                   │   ├── A11-A20_ongoing13.ply
│                   │   └── A11-A20_ongoing14.ply
│                   ├── kernels
│                   ├── logs
│                   │   ├── ...
│                   │   └── KPFCNN_ParisLille3D_torch
│                   │       ├── ...
│                   │       └── checkpoint
│                   ├── Open3D-ML
│                   │   ├── ...
│                   │   └── ml3d
│                   │       ├── ...
│                   │       ├── configs
│                   │       │   ├── ...
│                   │       │   ├── kpconv_parislille3d.yaml
│                   │       │   └── kpconv_parislille3d_mod.yaml
│                   │       └── datasets
│                   │           ├── ...
│                   │           └── Paris
│                   │               ├── ...
│                   │               ├── excluded_from_train
│                   │               ├── Old_test
│                   │               ├── test_10_classes
│                   │               └── training_10_classes
│                   │                   ├── ...
│                   │                   ├── remove_original_cloud_index.sh
│                   │                   └── reclassify_barriers.sh
│                   ├── FullCampus.ply
│                   ├── Campus_BlockA.ply
│                   ├── Campus_BlockBa.ply
│                   ├── Campus_BlockBb.ply
│                   ├── Campus_BlockBc.ply
│                   ├── Campus_BlockCa.ply
│                   ├── Campus_BlockCb.ply
│                   ├── Campus_BlockCc.ply
│                   ├── IIITB_Base - Target_Readable.ply
│                   ├── IIITB_Base.ply
│                   ├── IIITB_Base_RGB.ply
│                   ├── IIITB_Base_train.ply
│                   ├── IIITB_Base_train2.ply
│                   ├── ckpt_IIITB.pth
│                   ├── kpconv_parislille3d_202011241550utc.pth
│                   ├── outlier_remover.py
│                   ├── downsampler.py
│                   ├── preprocess_pipeline.py
│                   ├── outlier_remover.py
│                   ├── background.py
│                   ├── resource_monitor.log
│                   ├── processing_log.csv
│                   ├── clean_iiitb_base.ipynb
│                   ├── reclassify_iiitb_base.ipynb
│                   ├── preprocess_pipeline.ipynb
│                   ├── partitioner.ipynb
│                   ├── KPConv_train.ipynb
│                   ├── KPConv_train_after_clean.ipynb
│                   ├── KPConv_single_inference.ipynb
│                   └── KPConv_multiple_inference.ipynb
└── F:
    └── PE
        └── Mathew Backup - 193 GB
            ├── Campus Inference
            │   └── All inferences
            └── Grps (A to Bb)
```

Figure 7: Final directory structure of the campus-scale point cloud segmentation project

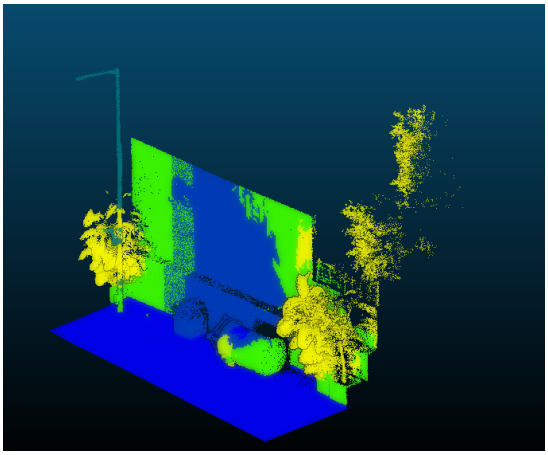
7 Observations and Recommendations

7.1 Observed Failure Modes

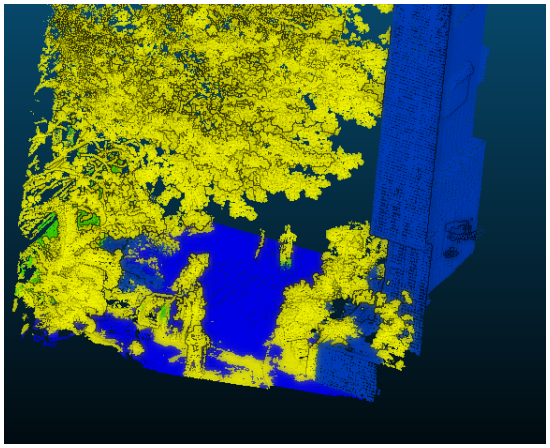
- Frequent confusion between ground, building, and barrier classes.
- Shadowed ground regions and other features under trees misclassified as vegetation.
- Patches on the ground behind pedestrians (resembling shadows) misclassified into pedestrian class.
- Complex structures (e.g., machinery with nearby people) poorly segmented.
- Poles classified correctly only near their tops; lower sections mislabeled inot vegetation due to structural similarity with tree trunks.



(a) Building(green)–ground(light blue) misclassification along structural boundaries



(b) Barrier(green)–building(dark blue) misclassification due to geometric similarity



(c) Segmentation errors in complex scenes involving multiple object types

Figure 8: Representative segmentation failure modes observed during inference on campus point cloud data

7.2 Recommendations

Manual refinement was found to be inefficient and non-scalable. Future efforts should focus on:

- Improving training strategies and data diversity.
- Addressing class imbalance explicitly.
- Exploring OOD training and region growing strategies for training model.
- Exploring domain adaptation rather than brute-force annotation.

Prepared by: Mathew Joseph
Date: December 24, 2025