CONSTRUCTEYE: AI-DRIVEN AUTONOMOUS MOBILE ROBOTICS FOR 3D CONSTRUCTION PROGRESS MONITORING

Ashiwin Rajendran

NUS-ISS, National University of Singapore, Singapore 119615

ABSTRACT

This project introduces **ConstructEye**, an innovative AI-powered mobile robotic system designed to automate monitoring of construction progress through advanced semantic segmentation and 3D point cloud analysis.

Traditionally, construction progress tracking has been based on manual inspection methods that are labor intensive, time consuming, and often inaccurate. These limitations can lead to inefficiencies, project delays, and cost overruns. **ConstructEye** addresses this critical challenge by integrating mobile robotics with state-of-the-art deep learning techniques for understanding the semantic scene.

The system is equipped with an RGB-D camera and a LiDAR sensor mounted on a mobile robot platform to either manually or autonomously capture detailed 3D scans of construction environments. These scans are processed in real time using deep learning-based semantic segmentation models trained on publicly available indoor scene datasets. The models are able to detect key architectural and interior components such as walls, floors, doors, windows, and furniture.

The segmented scene is then systematically compared with a reference 3D models created by designers to detect discrepancies, track construction progress, and identify missing structural elements. This automated pipeline provides accurate, real-time insights for project managers and significantly reduces the effort required for manual monitoring. The system improves construction oversight, facilitates timely decision making, and improves overall project execution fidelity.

Keywords: Construction Progress Monitoring, Mobile Robotics, Semantic Segmentation, 3D Point Cloud Analysis

1 Introduction

Performing accurate construction progress tracking remains a significant challenge in the architecture, engineering, and construction industry due to the reliance on manual inspections, which are often time consuming, labor intensive, and prone to errors. These limitations can lead to inefficiencies, missed deadlines, and budget overruns. As construction projects become increasingly complex, the need for intelligent automated systems capable of delivering real-time progress evaluation has become critical.

This project presents **ConstructEye**, a mobile robotics-based digital twin system that leverages semantic segmentation and 3D point cloud analysis to automate the monitoring of construction progress. The system autonomously captures site data using a mobile robot equipped with LiDAR and RGB-D cameras, generating high-resolution 3D scans of indoor environments which can be extended to outdoor as well.

The captured scans are processed using deep learning-based semantic segmentation models trained on large-scale indoor datasets. These models have been extensively fine-tuned to optimize accuracy and ensure reliable real-time inference. Structural elements such as walls, floors, windows, doors, and furniture are identified and extracted from the 3D point cloud. These detected elements are then compared against the original 3D design created using CAD or Solidworks to assess construction completeness and highlight deviations.

The highlights of this project are summarized below. To exhibit the development as a prototype, the problem statement has been addressed through Mobile Robotic simulation platforms.

- **Digital Twin for Real-Time Monitoring:** Enables dynamic tracking by continuously capturing and comparing live construction site data with reference design models.
- **AI-Powered Progress Evaluation:** Employs semantic segmentation on 3D point clouds to detect and classify structural components, ensuring a robust assessment of the construction status.
- **Optimized Deep Learning Models:** Semantic segmentation networks are enhanced through parameter tuning and refinement techniques to maximize performance under real-world conditions.
- **Autonomous Mobile Robotics:** A mobile robot equipped with LiDAR and RGB-D sensors navigates the site autonomously, reducing the dependency on manual inspections and minimizing human error.

- **Human-Readable Construction Insights:** Translates technical inference results into clear insights that construction managers can use to evaluate progress and make decisions.

- **Versatile and Scalable pipeline:** Designed for future extensions, including automated anomaly detection, progress percentage estimation, and aerial mapping through drone integration. Techniques such as Iterative Closest Point (ICP) or global registration can also be added for precise alignment and deviation analysis.

2 Literature Review

Several recent studies have explored the integration of computer vision and 3D point cloud analysis for automated construction progress monitoring. The key contributions from the existing literature are outlined below:

A computer vision and point cloud-based monitoring approach for automated construction tasks using full-scale robotized mobile cranes [1]: This study proposes a comprehensive computer vision and point cloud-based monitoring framework for use with full-scale robotized mobile cranes. The system is designed to offer real-time feedback to crane control mechanisms by recognizing construction resources, conducting 3D measurements of site objects, and tracking worker movements and activities. The feedback loop aims to improve task automation and situational awareness during construction operations.

Automation of Construction Progress Monitoring by Integrating 3D Point Cloud Data with an IFC-Based BIM Model [2]: This work introduces a methodology that aligns the built point cloud data with the built building information modeling (BIM) models. Using laser scanning technologies, the system enables automated assessment of the progress of construction.

Robot-Assisted Mobile Scanning for Automated 3D Reconstruction [3]: This research explores an autonomous 3D scanning solution using a legged mobile robot equipped with a solid-state LiDAR sensor. The robot facilitates continuous laser scanning of construction environments, supporting the automated generation of 3D reconstructions.

These studies collectively highlight the advances in construction automation using robotics and point-cloud analysis. However, most focus on outdoor environments, structured setups (e.g., cranes), or rely heavily on BIM integration. In contrast, our proposed solution—**ConstructEye**—uniquely focuses on real-time semantic understanding and object-level comparison using deep learning models, with broader applicability across indoor and outdoor construction scenarios.

3 Proposed Approach

To address the challenge of monitoring the progress of the construction, an integrated ROS-based system pipeline was developed. This pipeline combines multiple components—robot simulation, 3D mapping, semantic segmentation, and object comparison—into a unified framework as illustrated in **Fig. 1**. Key technologies include Robotnik simulation packages, RTAB-Map for SLAM, and a deep learning-based inference module.

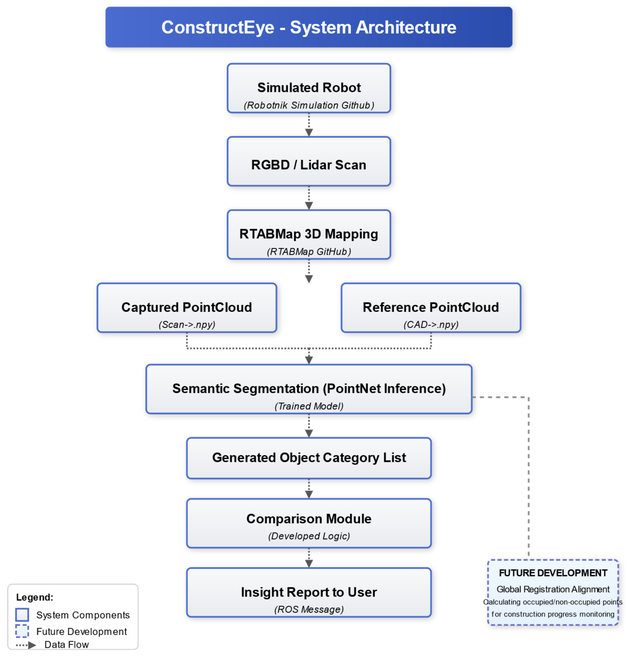


Fig. 1. Proposed Technical Approach

The system comprises the following components:

- **Simulated Mobile Robot:** The Summit-XL robot model, equipped with plugins for sensors which includes RGB-D and LiDAR, is used for data acquisition within a Gazebo simulation environment.

- **SLAM Mapping (RTAB-Map):** A real-time SLAM module using RTAB-Map integrates either RGB-D or LiDAR as input to generate an updated 3D map of the environment which employs the mapping-based 3D reconstruction method.

- **Point Cloud Capture (ROS Node):** A ROS subscriber node extracts the scanned point cloud data from the robot and saves it for further analysis (Inference).

- **Reference Model Loader:** A module designed to load predefined reference 3D point clouds derived from original CAD models representing the intended construction state.

- **Semantic Segmentation Inference:** This module uses trained deep learning models (e.g., PointNet) to classify elements of the point cloud (e.g., wall, chair, table, window) from both captured and reference scans.

- **Object Comparison Module:** After semantic segmentation, this module compares the identified object classes in the scanned point cloud with those in the reference point cloud to assess structural completeness or detect missing elements. In the current implementation, this comparison is performed through a basic message parsing mechanism. However, the module is designed with extensibility in mind - it can be enhanced in future iterations to calculate geometric differences between point clouds and estimate construction completion percentages, as illustrated in **Fig. 1** in the proposed technical approach.

- **ROS Message Output:** The detected and compared results are encapsulated into a custom ROS message for decision making.

- **Dockerized Pipeline:** The entire system is packaged into a Docker container to ensure reproducibility and ease of deployment in any Ubuntu 20.04 environment.

4 Experimental Results

4.1 Dataset

Public datasets have been utilized for training the semantic segmentation model in this project, while self-sourced Gazebo simulation models were employed for inference and testing.

- **Stanford 3D Indoor Scene Dataset (S3DIS):** This dataset comprises six extensive indoor areas encompassing 271 rooms. Each point in the scene's point cloud is annotated with one of 13 semantic categories as illustrated in **Fig. 2**, making it a valuable resource for indoor semantic segmentation tasks.

- **Permission Compliance:** Proper permissions have been secured to utilize these open-source public datasets in accordance with usage guidelines.

4.2 Implementation Details

4.2.1 Reference Model Generation and Feasibility

In this prototype, reference models are derived directly from existing Gazebo models available within the simulation environment. These reference models serve as the baseline for semantic comparison against the real-time point clouds captured during simulated robot navigation.

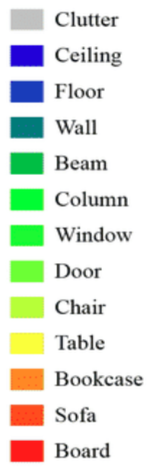


Fig. 2. Semantic object list in S3DIS dataset

Real-World Deployment Consideration: In practical deployments, the reference model would originate from actual building designs created using CAD software such as SolidWorks. To align with the proposed inference pipeline, these design models must be converted into `.npy` (NumPy) point cloud format. While the detailed steps for such conversion are beyond the scope of this report, feasibility has been validated externally.

CAD-to-PointCloud Workflow (Feasibility - Reference and Prediction): Most CAD tools support export to `URDF` using built-in plugins or exporters. This URDF model can then be converted to `SDF` format and further linked to `STL` files representing mesh geometry. These STL files can be transformed into `.npy` format, making them compatible with the semantic segmentation inference engine. As a note, Some SolidWorks models -typically works in millimeter scale, so dimensional adjustments may be required to ensure unit consistency with the Gazebo simulation (This can be done by adjusting the URDF parameters).

For real-world data collection, the mobile robot can autonomously scan its surroundings using LiDAR or RGB-D sensors. The captured point cloud data is similarly converted to the `.npy` format for processing and semantic labeling.

Simulation Pipeline: In the current simulation-based prototype:

- The reference point cloud in `.npy` (Numpy) format is derived by scanning the Gazebo-based URDF models.

- The mobile robot is navigated by the user through the same environment to gather predicted point clouds, which are also saved in `.npy` format.

Both files are passed into the PointNet-based semantic segmentation model to infer class labels and perform structural comparison. The details of simulation setup and data acquisition are explained in the following sections.

4.2.2 Robot Simulator Setup

To simulate differenet/various environment, the summit_xl_sim package from Robotnik was utilized. The repository was cloned from:

```
https://github.com/RobotnikAutomation/summit_xl_sim/tree/
                    noetic-devel
```

For compatibility with ROS Noetic, the `ros-devel` branch was explicitly checked out within the `robotnik_msgs` package. The workspace was then built successfully after removing non-essential packages.

Sensor Integration: After careful review, appropriate sensors has been selected. The robot was equipped with an Orbbec RGBD camera and a Velodyne VLP-16 LiDAR. Both sensors were mounted and activated within the URDF model to enable 3D perception of the environment.

Velodyne Simulation Plugin: To accurately emulate LiDAR behavior in Gazebo, the Velodyne simulator plugin was installed.

Validation and Launch: ROS topics were monitored to confirm that all sensors were actively publishing data. Additionally, configuration parameters for the Velodyne VLP-16 were tuned to reflect real-world performance characteristics. A sample Snippet from the Rviz platform is visualized in **Fig. 4**.

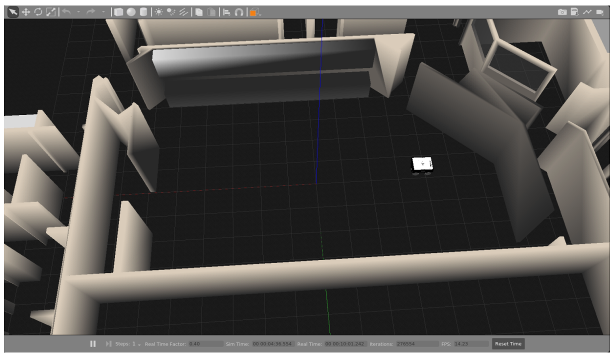


Fig. 3. Gazebo Simulation

4.2.3 Robot Teleoperation Setup

To enable manual control of the mobile robot during simulation, a keyboard teleoperation interface was configured.

Installation: The teleoperation control package has been installed to maneuver the robot across the simulation environment.

Execution: The node was then executed with a custom topic remapping to direct velocity commands to the robot's control interface.

This setup allows the user to control the robot's linear and angular motion using keyboard input, useful for navigating the environment.

4.2.4 RTAB-Map Setup

RTAB-Map was integrated into the simulation environment to enable real-time 3D mapping and localization.

Installation: The following dependencies were installed:

- `perception_pcl` package: Cloned into the workspace and switched to the `melodic-devel` branch.
- `rtabmap_ros` package: Installed by following the official instructions from the RTAB-Map GitHub repository.

```
https://github.com/introlab/rtabmap_ros/tree/master
```

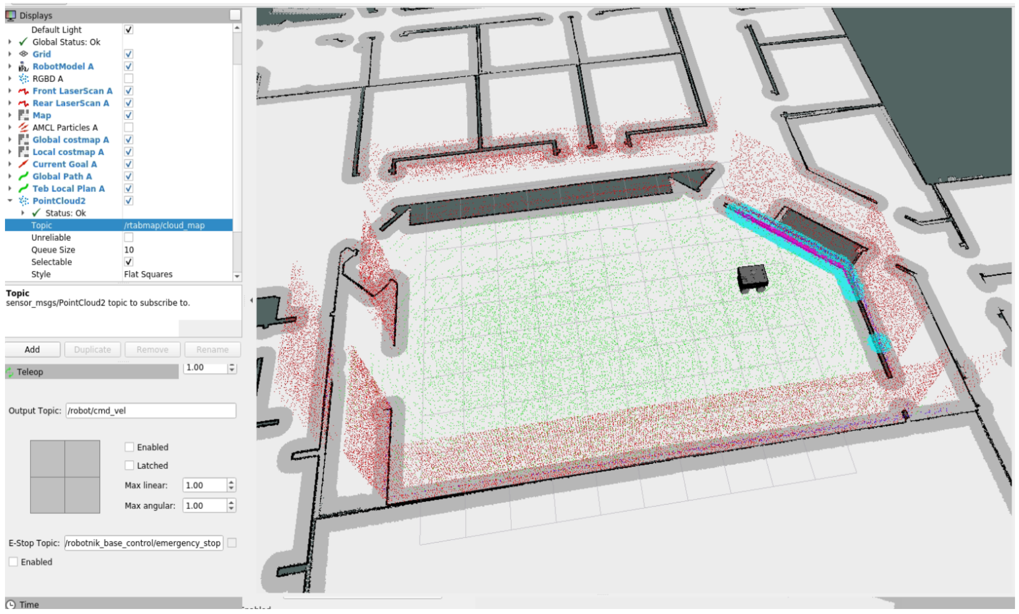
Execution: RTAB-Map is utilized for 3D reconstruction through mapping-based alignment of point clouds. Various parameters and thresholds were carefully fine-tuned through multiple iterations to achieve accurate and consistent 3D environment mapping. It is important to note that the system supports flexible configuration — the mapping process can be initiated using either LiDAR-only input or RGB-D camera input, depending on the sensor setup.

This configuration enables robust LiDAR-based SLAM, eliminating the dependency on visual odometry or RGB-D data, making it adaptable for both indoor and outdoor mapping scenarios. A sample PointCloud 3D Map generated from the RTAB-mapping is illustrated in **Fig. 5**.

4.2.5 PointNet Network Architecture

The segmentation backbone is based on PointNet [4], which is a deep learning architecture specifically designed to process unordered 3D point clouds directly, avoiding the need for voxelization or image projections. Each point is passed through a shared Multi-Layer Perceptron (MLP), and a symmetric max-pooling function aggregates global features, ensuring permutation invariance.

A spatial transformer module further enhances robustness by learning to align input data to a canonical space. PointNet



Default Light	✓
Global Status: Ok	
Grid	✓
RobotModel A	✓
RGBD A	☐
Front LaserScan A	✓
Rear LaserScan A	✓
Map	✓
AMCL Particles A	☐
Global costmap A	✓
Local costmap A	✓
Current Goal A	✓
Global Path A	✓
Teb Local Plan A	✓
PointCloud2	✓
Status: Ok	
Topic	/rtabmap/cloud_map
Unreliable	☐
Queue Size	10
Selectable	✓
Style	Flat Squares

Topic
sensor_msgs/PointCloud2 topic to subscribe to.

Teleop 1.00

Output Topic: /robot/cmd_vel

☐ Enabled
☐ Latched
Max linear: 1.00
Max angular: 1.00

E-Stop Topic: /robotnik_base_control/emergency_stop
☐ Enabled

Fig. 4. RViz showing the mobile robot navigating within the simulated environment. Green points represent the ground surface, while red points correspond to wall features. The point cloud was generated using RTAB-Map and published as a `PointCloud2` ROS message.

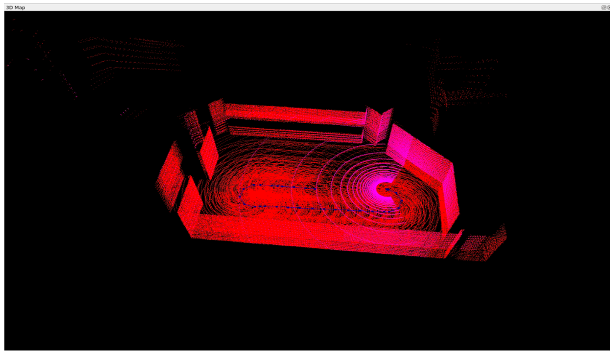


Fig. 5. 3D point cloud registration generated using RTAB-Map for real-time mapping and localization within the simulated environment

supports both classification and segmentation tasks, combining local and global features to make accurate per-point predictions. It achieves competitive performance across benchmarks and offers theoretical guarantees of robustness to noise and missing data.

The PointNet model serves as the core architecture for semantic segmentation in this project. Its primary function is to operate directly on unordered 3D point sets and predict per-point semantic labels. The architecture is composed of three main components: a symmetric function for aggregation, global-local feature fusion, and transformation alignment networks. The PointNet Architecture is illustrated in **Fig. 6**.

Symmetric Function for Unordered Point Sets: One of the core challenges in point cloud processing is handling the unordered nature of point sets. PointNet addresses this by applying a shared Multi-Layer Perceptron (MLP) to each point independently, followed by a symmetric function—specifically, max pooling—to aggregate point-wise features into a global descriptor. This strategy ensures that the network is invariant to permutations of input points.

Global and Local Feature Fusion: After generating the global feature vector through max pooling, this representation is concatenated back with individual point features. This enables the network to capture both local geometric structures and global semantic context. The fused features are passed through additional MLP layers to yield per-point predictions. This architecture allows PointNet to perform well on tasks such as scene segmentation, where both fine-grained details and global scene understanding are critical.

Alignment via T-Net Modules: To ensure robustness against geometric transformations (e.g., rotations and translations), PointNet introduces two alignment networks—T-Nets. The first T-Net learns a 3×3 affine transformation matrix that aligns the input point cloud to a canonical pose. A second T-Net aligns the feature space by learning a higher-dimensional transformation matrix. A regularization loss is added to encourage this matrix to approximate an orthogonal transformation, ensuring minimal information loss:

$$L_{\text{reg}} = \|I - AA^T\|_F^2$$

where A is the predicted feature transformation matrix, and I is the identity matrix.

4.2.6 PointNet Semantic Segmentation Pipeline

The PointNet architecture, originally designed for part-level segmentation, is readily adaptable for scene-level semantic segmentation, where each point in a 3D environment is assigned a semantic label representing object categories such as walls, chairs, tables, or clutter. The semantic segmentation network training and inference pipeline was implemented using TensorFlow 1.15 in a custom Conda environment. The process is outlined below:

1. **Environment Setup and Dependencies:**
    - Conda environment was created with Python 3.7 and TensorFlow GPU 1.15.
    - Additional dependencies such as `protobuf=3.20` and `h5py` were installed.
2. **Dataset Preparation and Preprocessing:**
    - The S3DIS dataset was extracted and processed using `collect_indoor3d_data.py`.
    - Room-wise slicing and label generation were performed.
    - A modified version of `gen_indoor3d_h5.py` was used to generate training-ready HDF5 files.
3. **Training Configuration and Optimization:** For model training and evaluation, the Stanford 3D Indoor Scene Dataset (S3DIS) was employed. This dataset includes 3D scans from six distinct areas, covering a total of 271 rooms, with each point labeled under one of 13 semantic categories. The training pipeline was extensively optimized to align with the practical requirements of this project. Key training configurations and optimization strategies are outlined below.
    - Input: XYZ coordinates, optionally with RGB and normalized room coordinates.
    - Loss Function: Categorical cross-entropy.
    - Optimizer: Adam; trained for approximately 42 epochs.
    - Batch size ranged from 2 to 4 based on GPU constraints.
    - Learning rate scheduling with decay and dropout for generalization.
    - Early stopping and checkpointing enabled for efficient convergence.
    - Hyperparameter Tuning based on Training Progress
    - Multi scale pooling to preserve dominant and smooth features.
    - Redundant fully connected layers were optimized for faster convergence
    - Attention mechanisms were integrated to improve feature importance weighting.
    - Model pruning was applied to convert to float32 weights for efficient inference on the embedded platform.
4. **Inference Pipeline Enhancements:** The trained model checkpoints were saved in the `.ckpt` format. Post-training, the inference script was adapted to support inference on a single point cloud input. Significant modifications were made to the original inference pipeline to effectively process point cloud data captured by the simulation robot equipped with a Velodyne LiDAR sensor. Semantic labels were inferred for both the reference model and the live-captured point clouds. These outputs were then parsed to users to derive meaningful insights.
    - Inference optimized for real-time single point cloud input (not batched).
    - Grid-based sampling was replaced with KD-tree partitioning for speed and consistency.
    - Class-based color mapping and object counting were added for better visualization.
    - Robust error handling was integrated for real-time data inputs.
    - The inference script has been integrated with ROS nodes to enable seamless execution upon service calls. When the ROS service is triggered, the system performs inference on the input point cloud, which can originate from either the reference model or live-captured data.

4.2.7 ConstructEye ROS Pipeline

The integration of all subsystems—robot simulation, teleoperation, RTAB SLAM mapping, deep learning model (PointNet) training, and the inference mechanism—forms the core of the ConstructEye ROS-based pipeline. While model development and training constitute a significant part of the project, real-world deployment demands seamless orchestration among these components. This is achieved using the Robot Operating System (ROS) infrastructure.

Inference Execution via ROS Services: The inference script is designed to operate on two types of input:

- **Reference Point Cloud:** Generated from the 3D design model.
- **Captured Point Cloud:** Acquired through real-time scanning using the mobile robot.

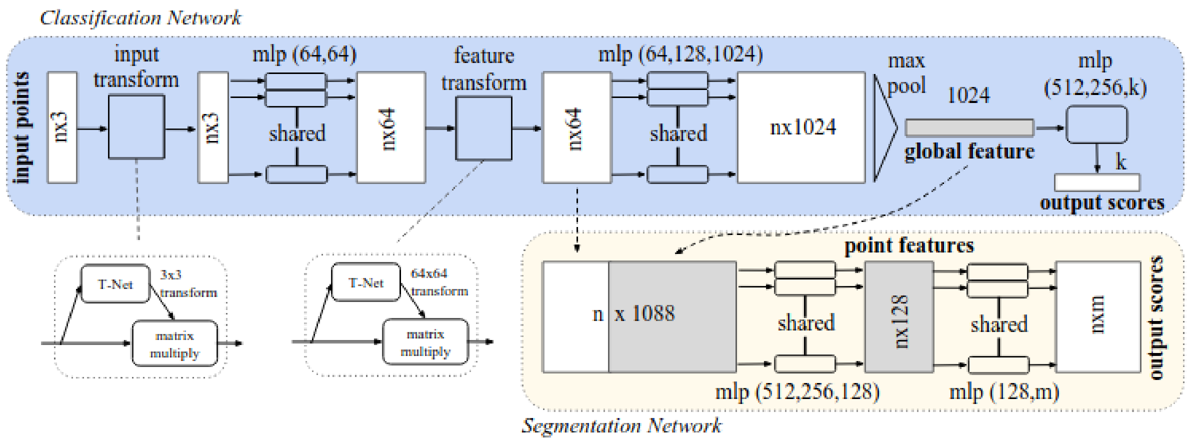


Fig. 6. PointNet Architecture. The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per-point scores. "mlp" stands for multi-layer perceptron, numbers in brackets are layer sizes. Batch normalization is used for all layers with ReLU, and dropout layers are applied in the final MLP for classification.

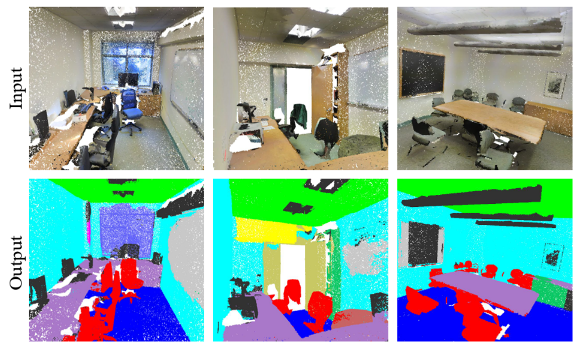


Fig. 7. Example from the S3DIS dataset: The top row shows the input point cloud, while the bottom row displays the corresponding semantic segmentation results (on points).

To make the inference modular and user-controllable, the core script is encapsulated as a ROS node. Users can trigger inference simply by calling a dedicated ROS service. Upon invocation, the system processes the point cloud and returns the detected objects.

Object Detection and Messaging: The results of inference—i.e., the list of detected semantic classes such as walls, doors, chairs, and tables—are published as a custom ROS message. This inference provides both quantitative data and real-time visual feedback to users through ROS. The respective object list which has been trained can be seen in (**Fig. 2**). And the respective ROS message and visualization can be seen in (**Fig. 8**) and (**Fig. 9**).

Deployment Strategy: To ensure robustness and reproducibility:

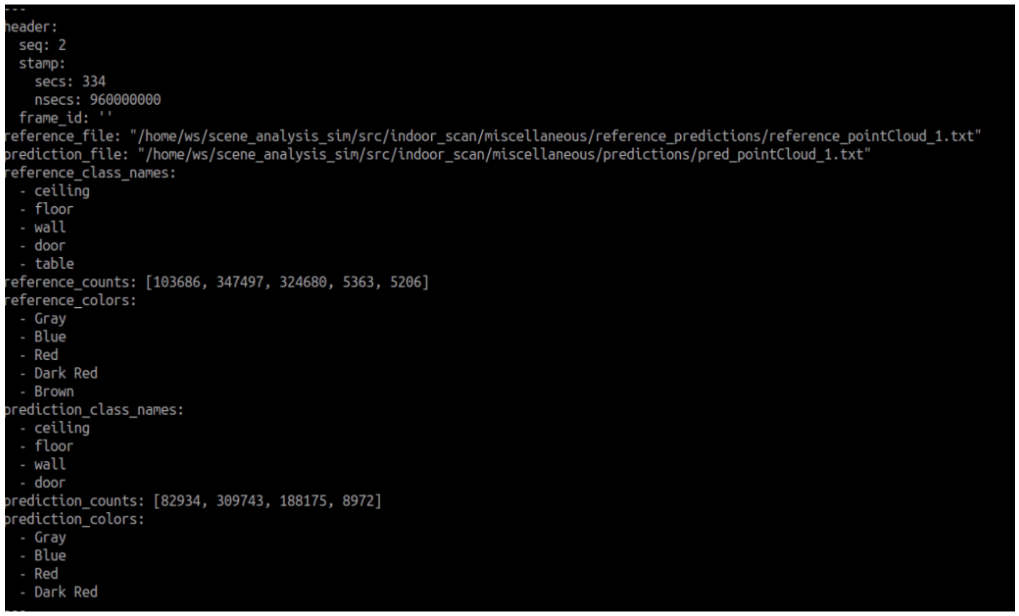
- The entire pipeline—simulation, SLAM, inference, and ROS nodes—is managed using `tmux` sessions.
- All components are packaged into a single Docker image based on Ubuntu 20.04 with ROS Noetic.
- Upon container startup, all necessary modules are automatically launched, providing a turnkey solution for end-to-end system execution.

This deployment strategy ensures that project managers or system evaluators can quickly initiate the platform and perform inference without additional manual steps.

4.3 Performance Metrics

The training and validation were conducted concurrently, with the model being trained on the entire S3DIS dataset and validated specifically on Area 6 to assess generalization. The corresponding outputs were saved as ".obj" files with logs. The following metrics were used during the evaluation process:

- **Mean Loss:** Average loss computed across all training batches.
- **Accuracy:** Overall classification accuracy on the training dataset.
- **Eval Mean Loss:** Loss calculated on the validation subset (Area 6).



```
---
eader:
  seq: 2
  stamp:
    secs: 334
    nsecs: 960000000
  frame_id: ''
eference_file: "/home/ws/scene_analysis_sim/src/indoor_scan/miscellaneous/reference_predictions/reference_pointCloud_1.txt"
rediction_file: "/home/ws/scene_analysis_sim/src/indoor_scan/miscellaneous/predictions/pred_pointCloud_1.txt"
eference_class_names:
  - ceiling
  - floor
  - wall
  - door
  - table
eference_counts: [103686, 347497, 324680, 5363, 5206]
eference_colors:
  - Gray
  - Blue
  - Red
  - Dark Red
  - Brown
rediction_class_names:
  - ceiling
  - floor
  - wall
  - door
rediction_counts: [82934, 309743, 188175, 8972]
rediction_colors:
  - Gray
  - Blue
  - Red
  - Dark Red
```

Fig. 8. ROS message of (**Fig. 9**) depicting the objects in the reference model predictions and scanned model predictions along with its colors and visualization. In future, the counts can be normalized to estimate the percentage of occupancy .

- **Eval Accuracy:** Classification accuracy on the validation subset.
- **Eval Average Class Accuracy:** Mean per-class accuracy across the 13 semantic categories.

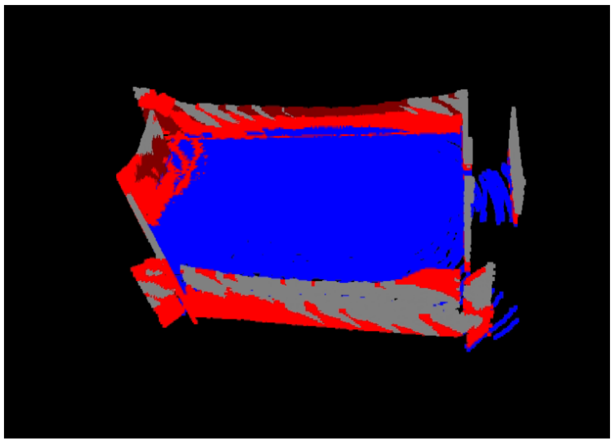


Fig. 9. Segmented point cloud visualization showing various object categories, with each class represented in a distinct color. Refer (**Fig. 8**) for its respective ROS message output.

4.4 Experimental Results

The proposed segmentation model demonstrated high performance, achieving accurate and consistent results in real-time inference. Qualitative outcomes were validated through visualizations of point cloud scans captured by the simulated mobile robot. Quantitative evaluation was conducted using standard performance metrics, which are discussed below.

Training and validation progress is illustrated in the subsequent figures: the learning rate schedule (**Fig. 10**), loss curve (**Fig. 11**), and accuracy curve (**Fig. 12**). The noticeable drift observed between epochs 20k to 30k is attributed to a modification in the training configuration followed by a restart of the training process. The final evaluation metrics are summarized as follows:

- **Mean Training Loss:** 0.225680
- **Training Accuracy:** 0.920728
- **Validation (Area 6) Mean Loss:** 0.630568
- **Validation Accuracy:** 0.822827
- **Validation Average Class Accuracy:** 0.628137

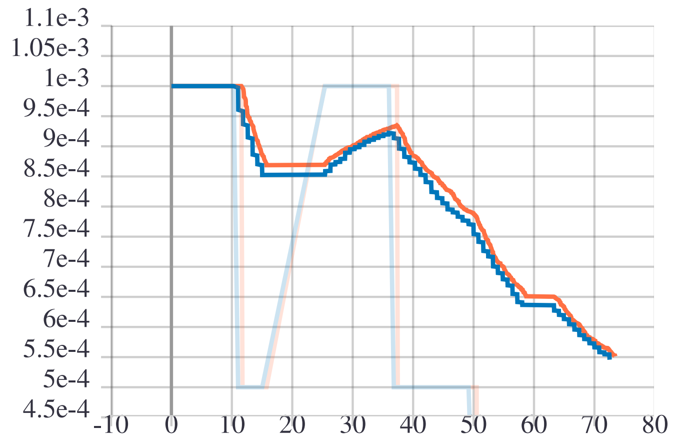


Fig. 10. Learning Rate plot: The orange curve represents training LR over epochs, while the blue curve denotes test LR over epochs.

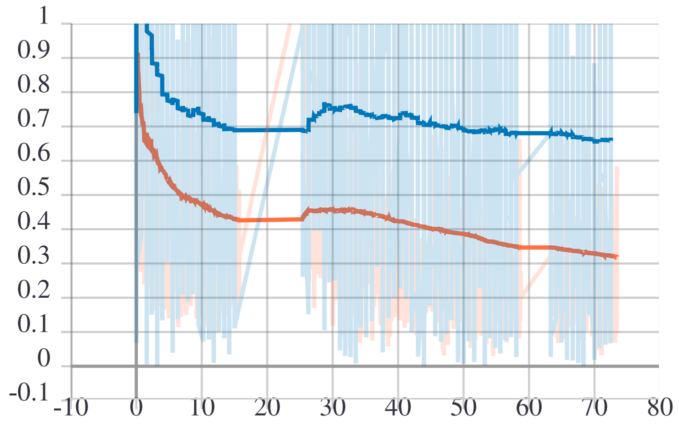


Fig. 11. Loss plot: The orange curve represents training loss over epochs, while the blue curve denotes test loss over epochs.

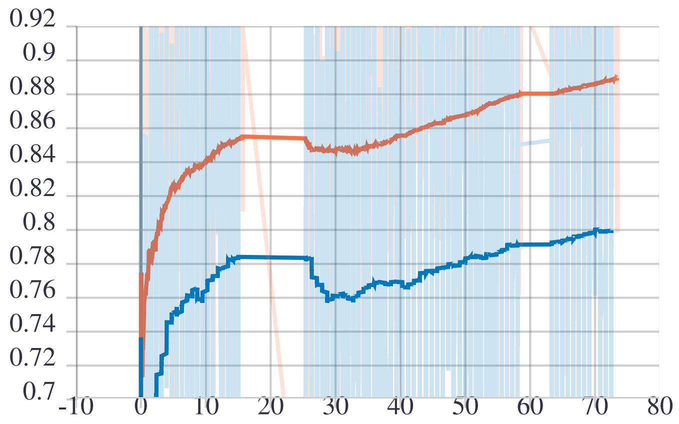


Fig. 12. Accuracy plot: The orange curve represents training accuracy over epochs, while the blue curve denotes test accuracy over epochs.

4.5 Discussions and Limitations

Despite the promising performance of the ConstructEye system, a few limitations were observed during experimentation and testing.

Segmentation Accuracy and Noise Sensitivity: The semantic segmentation model occasionally yielded incorrect class predictions, primarily due to noise and sparsity in the captured point cloud data. A contributing factor to this discrepancy was the domain shift introduced by training the model on real-world data while evaluating it on simulation-generated scans. The key sources of inaccuracy include:

- Sensor noise causing fragmented or incorrectly labeled object boundaries.
- Occlusions and partial visibility leading to ambiguity in geometric structure recognition.

To mitigate these issues, pre-processing methods like *Statistical Outlier Removal* can be employed to filter noise and sharpen object boundaries. Additionally, increasing the simulated Velodyne LiDAR resolution from 16 to 32 lines improves the density of captured point clouds, further enhancing segmentation accuracy and model robustness during both training and inference.

Data Quality and Occlusions: Sparse point clouds and occlusions reduce the model's ability to identify small or partially visible objects. Improvements in RTAB-Map SLAM tuning, data augmentation, and additional training on more diverse datasets could address these challenges.

CAD-to-PointCloud Conversion Scope: While real-world applications require reference models derived from CAD designs, this project focuses on prototyping and simulation. Therefore, the CAD-to-`.npy` conversion process is not implemented in full. Nevertheless, existing workflows using tools like SolidWorks (via URDF and STL exports) make this conversion feasible, as discussed earlier in the report.

Exploring this pipeline in detail could introduce new integration and scaling challenges, which are acknowledged but remain outside the primary scope of this project.

5 Conclusions and Future Work

The proposed ConstructEye system successfully automates construction progress monitoring by leveraging robust semantic segmentation techniques and a mobile robotics platform. The current implementation, based on simulated environments and public datasets, demonstrates the technical feasibility and practical applicability of the approach for real-time analysis of building progress.

Future Enhancements and Extensions: Several opportunities exist to further improve and extend the system:

- **Aerial Mapping with Drones:** While the mobile robot enables efficient indoor scanning, integrating the system with drones would expand its reach to complex and large-scale outdoor structures, enabling full-site coverage and better accessibility.
- **Enhanced Dataset Generation:** Although publicly available datasets were used in this prototype, future work could involve curating custom datasets through traditional point cloud processing techniques (e.g., RANSAC-based segmentation) to generate labeled data for specific project environments, improving model robustness and generalization.
- **Construction Progress Estimation:** Future pipeline upgrades could incorporate point cloud alignment techniques such as global registration or Iterative Closest Point (ICP) to quantitatively assess discrepancies between reference and captured point clouds. Additionally, object counts from ROS messages can be normalized to derive occupancy percentages or progress metrics.
- **Safety Monitoring and AI Integration:** Real-time video analysis could be integrated for monitoring worker activity and enforcing safety regulations. Moreover, integrating Large Language Models (LLMs) would enable the system to convert raw ROS outputs into human-readable, descriptive reports for decision-makers.

Impact and Feasibility: The extensibility of this solution provides significant value across multiple dimensions:

- **Automation and Efficiency:** Reduces reliance on manual inspection and accelerates decision-making.
- **Data-Driven Insights:** Offers quantifiable progress metrics and visualizations for better project oversight.
- **Scalability and Flexibility:** Can be adapted for both small indoor setups and large-scale outdoor sites.
- **Integration with AI:** Future AI capabilities can offer predictive analytics and anomaly detection.
- **Cost and Resource Optimization:** Depending on budget constraints, the system can be deployed with minimal hardware (e.g., RGB-D camera on a mobile base) or enhanced with advanced sensors and compute units.

This project not only establishes a strong foundation for intelligent construction monitoring but also opens avenues for commercial viability by targeting project managers, construction firms, and industrial automation sectors seeking real-time, accurate, and scalable monitoring solutions.

6 Execution

For detailed instructions on running the final deliverables, please refer to the accompanying `README` file.

7 Author Contributions

All components and methodologies were individually developed by Ashiwin Rajendran.

8 References

[1] Xiao Pan, Tony T. Y. Yang, Ruiwu Liu, Yifei Xiao, and Fan Xie, "A computer vision and point cloud-based monitoring approach for automated construction tasks using full-scale robotized mobile cranes," *Journal of Intelligent Construction*, vol. 3, no. 2, pp. 9180086, 2025.

[2] Paulius Kavaliauskas, Jaime B. Fernandez, Kevin McGuinness, and Andrius Jurelionis, "Automation of construction progress monitoring by integrating 3d point cloud data with an ifc-based bim model," *Buildings*, vol. 12, no. 10, 2022.

[3] Difeng Hu, Vincent J.L. Gan, and Chao Yin, "Robot-assisted mobile scanning for automated 3d reconstruction and point cloud semantic segmentation of building interiors," *Automation in Construction*, vol. 152, pp. 104949, 2023.

[4] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," 2017.