Adapting SegMap: a LiDAR place recognition framework for standalone use in C++ applications

Thabisa Maweni[1*], *Paul* Amayo[2]

[1]Centre For Robotics and Future Production, CSIR, Pretoria, South Africa
[2]Electrical & Electronic Engineering, University of Cape Town, Cape Town, South Africa

Abstract. Autonomous mobile robots rely on accurate environmental mapping and continuous self-localisation for effective navigation, often achieved through complex algorithms that combine data from multiple sensors. Aru-SegMap is an adaptation of SegMap, a widely used 3D point cloud segment-based map representation, for modern ROS2-based and standalone C++ applications focused on localisation. SegMatch, a 3D point cloud segmentation and matching library integral to SegMap, reliably estimates a robot's position and detects loop closures. This adaptation involved modularising the original library, decoupling it from a deprecated TensorFlow C++ API and ROS1, and integrating visualisation capabilities, enabling greater flexibility and usability for continued robotics research and development. Aru_SegMap was validated using datasets of varied agricultural environments. It is functional, produces consistent segments, and provides reliable localisation.

1 Introduction

Autonomous mobile robots rely on precise environmental mapping and continuous self-localisation for navigation tasks [1]. However, traditional visual-based place recognition methods struggle in structure-poor environments like forests. LiDAR technology offers a solution insensitive to illumination and has long-range capabilities. SegMap, a map representation approach based on 3D point cloud segments, promises to address this challenge [1]. The core functionality of SegMap involves segmenting 3D point clouds into meaningful sections (objects in the vicinity of the robot) and matching them over time with segments previously visited to estimate the robot's position [2]. Despite its effectiveness, SegMap is tightly coupled with ROS 1 and deprecated dependencies such as TensorFlow 1 and is not directly usable outside of ROS environments as it is. To extend its utility, there is a need to adapt SegMap for use in modern ROS 2-based and independent of ROS for non-ROS C++ projects.

Autonomous mobile robots can be useful for various applications and fields or industries, including agricultural field robotics. The most critical capabilities for agricultural field robots are navigation, detection and mapping, with navigation being the first step [3]. This study

* Corresponding author: tmaweni@csir.co.za

aims to develop a navigation system through place recognition to allow the robot to recognise a scene where it has been before. For an agricultural use case, this is a crucial task which can assist in many activities such as crop health monitoring, traversing each row to identify and locate citrus trees (or other specific types) and potentially performing other tasks like spraying or harvesting while avoiding obstacles. The robot must be capable of determining its precise location within the orchard and operating efficiently. Accurate localization ensures that it follows the path planned by the navigation system. Accurate localisation will ensure that the robot follows through the intended path (that a navigation system plans). The project broadly studied how a robot can navigate an unstructured environment while mitigating viewpoint variations.

This paper presents the first of two contributions focused on deploying segment-based localisation for autonomous navigation in agricultural environments robust to viewpoint variations. This paper focuses on adapting the deprecated SegMap framework to obtain a working version that will be used as a reliable localiser outside the ROS ecosystem, with minimal dependencies and maximum modularity. This includes restructuring the SegMap pipeline for integration into a custom navigation architecture, decoupling core components, and implementing standalone tools for segment management, map I/O and visualisation. The adaptation addresses the practical challenges of applying SegMap in real-world robotic systems where computationally efficient, real-time operation and flexibility are critical. The new framework, Aru_SegMap (African Robotics Unit SegMap) is validated through localisation experiments on real agricultural data captured on wheeled mobile robots, demonstrating consistent performance across repeat runs and variable environments. This lays the foundation for the second part of the broad project, which will explore full deployment and integration into a teach-and-repeat navigation system.

2 Methodology

Reliability for the proposed localisation system is defined by the robot's ability to localise itself on a previously mapped area, detect loop closures, and generate a target map to localise on. This section covers the work done to achieve this reliability. Using the backbone of SegMap, our localiser computes point cloud Euclidean segmentation and matching. It performs a geometric verification instead of deep-learned features to estimate the robot's pose, as illustrated in Figure 1. This pose will be used as input for a navigation system.

This section describes the methods followed to adapt the SegMap framework for standalone use and to evaluate its reliability in unstructured environments. The Aru-SegMap system consists of three core components:

- A custom point cloud processing pipeline,
- A modified SegMap localisation backend and
- A lightweight visualiser

The architecture is designed to operate independently of the ROS framework, making it suitable for deployment on constrained robotic platforms.

2.1 Localisation workflow

The localisation process follows SegMap's standard approach, involving several key steps, as seen in Figure 1. Initially, point cloud data is collected from a LiDAR scanner, followed by pre-processing in a ROS environment before Euclidean cluster-based segmentation.

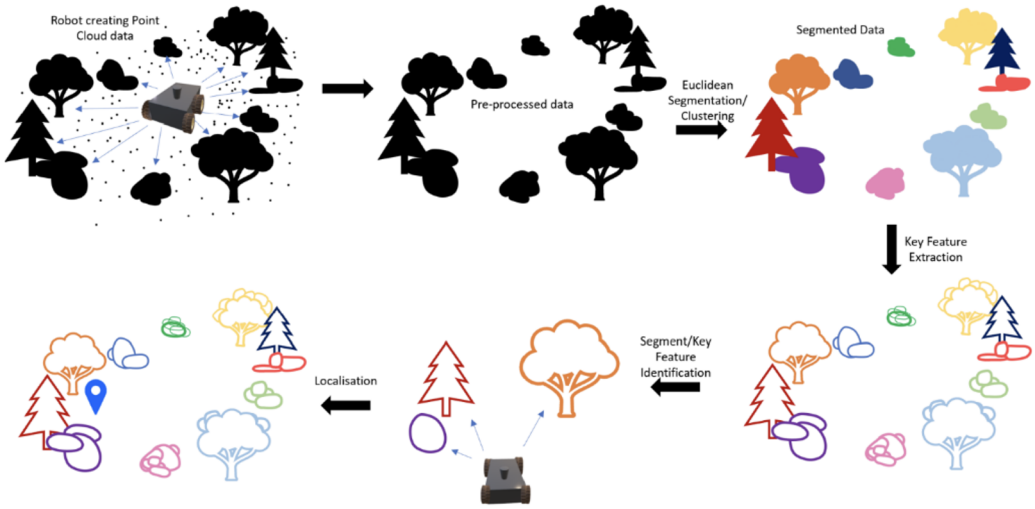

Fig.1. Diagram illustrating the workflow of the segment-matching approach from SegMap

Segmented clusters are then assigned oriented key poses, and relevant features are extracted. During experimentation, the robot performs segmentation and feature extraction on the source point cloud, matching segments to a previously collected target map. The segment matching performs a geometrical verification on the descriptors to estimate the robot's xyz pose. We add a validation loop to evaluate the reliability of our system which will be discussed in the experimental section.

2.2 Decoupling from ROS1

The primary challenge in deploying SegMap was its strong dependence on ROS1. The original framework is deeply integrated with ROS for handling data flow, transformations, and visualisation. This work aimed to eliminate these dependencies by modularising SegMap's core components into reusable functions integrated into a standalone C++ framework, eliminating the need for ROS messages, topics, and services while maintaining the core functionality.

In the original framework, ROS publishers and subscribers handle parameters and point cloud data flow through ROS-based messaging. RVIZ, a Visualisation tool, is used to display point clouds, segment centroids, and localisation results. The following changes were implemented:

- Data handling: ROS subscribers and publishers were replaced with a PCL-based I/O custom interface and direct memory access for reading point cloud data from .ply or .pcd files. Implemented a custom data loader to replace ROS Bag files with serialised point cloud datasets.
- Transformations: ROS TF2 transformations were replaced with Eigen-based transformation matrices.
- Visualisation: A Python-based Open3d visualiser was built using pybind [4] to render, local maps, the segmented target, source clouds and matches.

2.3 Modularization and integration

The original SegMap framework was modularised and refactored to enable a more maintainable and extensible system. Originally, it used multiple loosely connected libraries, namely *libpointmatcher* for scan registration, *laser_slam* for mapping, and *segmatch* for segment-based localisation. While effective, these components relied on outdated dependencies such as the C++ TensorFlow API, which is now deprecated and difficult to build or integrate. This work streamlines and creates a self-contained framework that eliminates these bottlenecks. Our implementation retains the core mapping and segmentation concepts and combines them with *laser_slam* for lidar-based odometry but replaces the TensorFlow-based descriptors with purely geometric features, reducing complexity and improving compatibility.

We extended the *SegMapper* class to accept pre-recorded scan and pose data from disk (in *.ply* and *.txt* formats), removing the dependency on ROS runtime infrastructure. This allows the system to be easily tested, debugged, and scaled to larger datasets or different environments. Additionally, the modular design allowed us to integrate external components such as *MapClosures* [5] for loop closure validation and evaluation, without disrupting the core pipeline. This modular and flexible architecture forms the foundation for the experiments and evaluations presented in the following sections.

2.4 Visualisation with Pybind11 and Open3D

Leveraging Pybind11, SegMap's input and output C++ data structures are exposed to Python, allowing for seamless integration of segmentation results and visualisation within a Python-based workflow Open3D. This visualisation pipeline provides real-time feedback, which is crucial for debugging and understanding the system performance in various scenarios. Functionality to colour-code source and target segments to aid visualisation was added to ensure consistent colour assignment per match, see Figure 2.

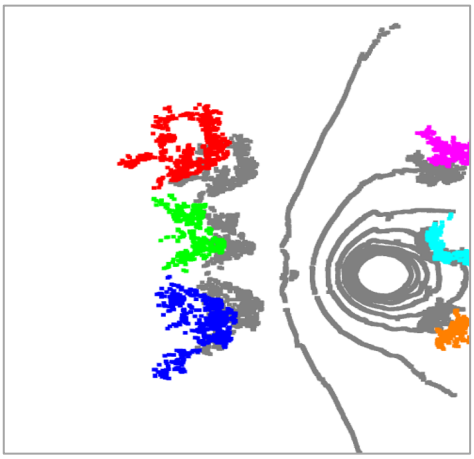

Fig. 2. Colour-coded segment match pairs in Open3D overlayed over target pointcloud in grey.

3 Experimental setup

This section describes the experimental setup used to evaluate the adapted SegMap framework, followed by the workflow applied across all localisation experiments.

3.1 Experiment setup and workflow

Localisation experiments follow the general workflow outlined in Figure 3. across all experiments. In this workflow, a physical mobile robot equipped with a LiDaR scanner generates point cloud data and records it as ROS bags while traversing the environment. These recorded bag files are processed in a custom Python-based script to extract individual *.ply* format files along with their corresponding transformation data in .txt format to create a dataset. This data is first pre-processed and used as input for the framework. The robot collects scans while navigating the environmment. Individual scans are extracted and passed through a preprocessing module to downsample using voxel filters, and ground points are removed for segmentation, feature extraction, and localisation.

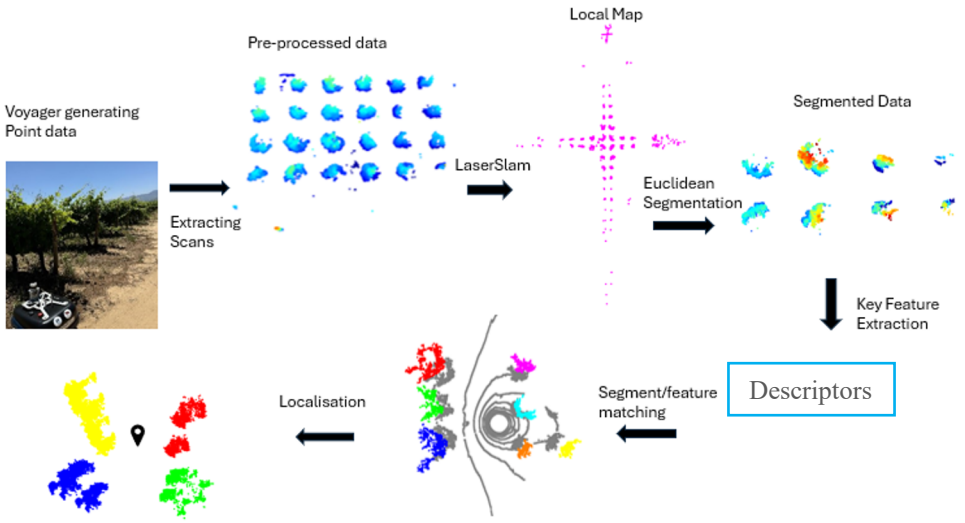

Fig. 3. Experimental workflow used across all datasets.

The validation dataset was collected using Voyager [6], a CSIR-developed research platform equipped with an Ouster LiDAR (OS-32) and the ClearPath Husky platform. The experiments were conducted at multiple locations with varying levels of structure and complexity. Data was collected over multiple sessions to evaluate consistency in segmentation and matching across different passes. Creating this dataset was crucial for validating the system, as it ensured testing on data that was entirely separate from the development set. This separation helped assess the system's ability to generalise to new, unseen environments.

Experiments were also conducted on the "CitrusFarm" Dataset [7], a well-used multimodal citrus tree farm dataset collected on a wheeled robot operating in agricultural areas. This data was used for baseline comparison because it has been used to test the capabilities of other place recognition algorithms and localisers and for mapping.

Fig. 4. Voyager (left) with Ouster LiDAR utilised for experimental field data collection in Paarl, Western Cape, alongside ClearPath Husky robot (right) with a Velodyne LiDAR.

3.2 Validation tests/experiments

To validate the segmentation and matching capabilities, the framework was fed the point cloud scans for the source and target clouds. The extracted features and their convergence were examined through the command line, with Figure 5. illustrating an example of this type of experiment.

The far-left image is a segmented target cloud of a stone farm dataset, representing stone fruit trees in rows. The middle image results from segmentation in the robot's selected segmentation radius. The last image illustrates a pair of target and source cloud segment matches. The visualiser assigns matched segments the same colour.

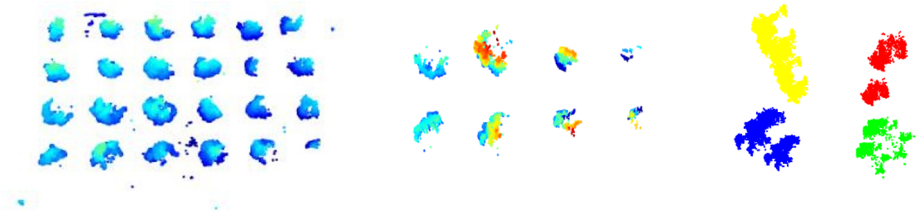

Fig. 5. An example of Aru-SegMap visualisation. Data splitting strategy for evaluation.

During early development and testing of Aru-SegMap, processing complete map trajectories (e.g., the full target clouds for sequence 2) required over 30 minutes per run, making iteration and debugging inefficient. To mitigate this, we introduced a data-splitting strategy focused on reducing processing time while preserving meaningful spatial relationships for evaluating loop closures.

We segmented the robot’s trajectory from the southeast portion of sequence 2 [7] into smaller, overlapping chunks. These splits were deliberately designed to include overlap between consecutive segments, enabling the evaluation of loop closure detection despite the reduced data volume. This approach significantly accelerated the development cycle, allowing for quicker visualisation, debugging, and analysis.

Later, the same splits were used during evaluation to assess how reliably Aru-SegMap could detect revisits within overlapping portions, effectively simulating realistic loop closure scenarios under resource-constrained conditions.

Fig. 6. Trajectory split used for efficient development and loop closure evaluation.

4 Results

Localisation and loop closure detection were both tested successfully. All experiments yield consistent results, and as the robot drives, we can reconstruct the 3D map/scene. For large-scale experiments, we used the CitrusFarm dataset, a multimodal agricultural dataset which covers over 7km [7]. For the small scale, we collected our own dataset from a local stone farm and a citrus farm. Processing and execution time outside ROS was a challenge. For the validation in the initial stages, the target map used had to be cropped significantly. Modifications to the pipeline were introduced to reduce the processing time for the standalone implementation.

4.1 Performance of Aru-SegMap

The performance of the standalone SegMatch framework was tested after modularising the system by testing the key SegMatch functionalities independently. A LiDAR scanning robot follows a structured lawnmower trajectory from the citrus farm during the session (Figure 7a). Although the environment presents challenges such as limited individual features due to the repetitive geometry of an orchard, as shown in Figure 7b, Aru-SegMap could extract consistent segments and maintain accurate localisation even in a repetitive environment. The estimated trajectory is shown in blue by a set of accumulated poses at every point where segmentation was performed. The system demonstrates:

- **Good place recognition** across similar-looking- rows. (no false positive loop closures)
- **Segment consistency**: the extracted segments appear to be representative of the environment's features. The consistent segmentation indicates robust data association in terrains with low variation.
- **The continuous trajectory** shows reliable pose estimation without accumulating drift and global consistency.

The blue line represents the estimated trajectory, and the coloured clusters are the extracted 3D segments. Despite the highly repetitive structure of the orchard, Aru-SegMap maintains stable segmentation and a consistent trajectory without introducing false loop closures.

a) Aerial view of Citrus Farm Sequence (seq 02) , showing trajectory from start (S) to end (E). Taken from the official repository.

b) Segmented clouds generated through Aru-SegMap with estimated robot trajectory

Fig. 7. Segmented 3D map and estimated trajectory from the Aru-SegMap system applied to a citrus orchard dataset (seq 02).

Figure 8. shows that the estimated trajectory follows the expected path (S →E) in the field. SegMap increments the extracted segments as we read scanned pointclouds. The following shows incremental segments visualised every 200th scan.

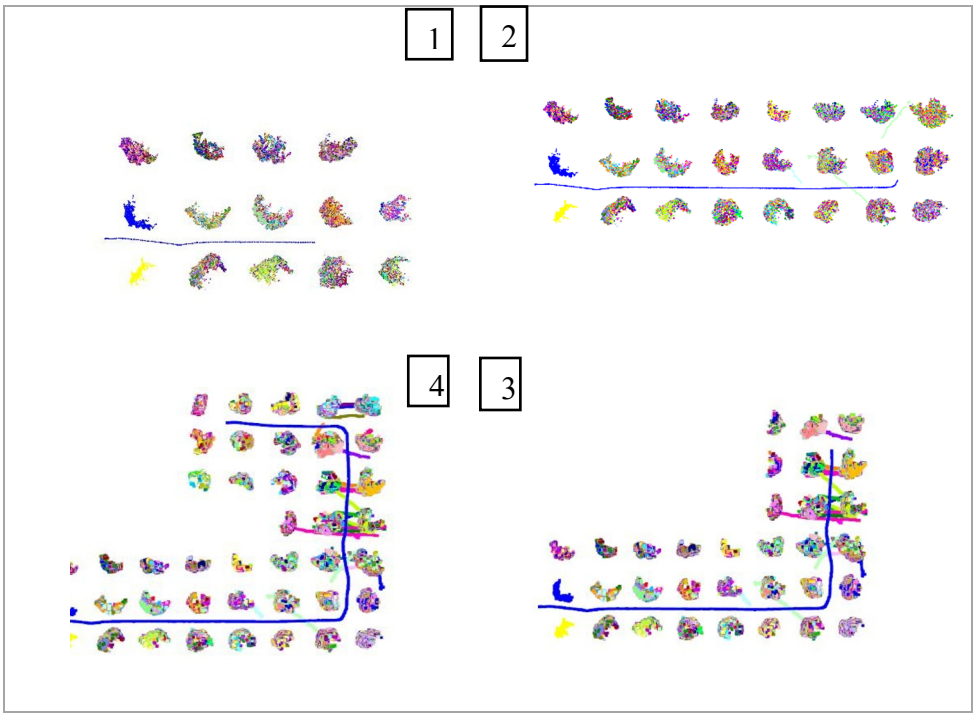

Fig. 8. Incremental segments making up the final map.

4.2 Visualisation pipeline and real-time feedback

A major improvement introduced in this adaptation was the integration of an interactive visualiser using Pybind11 and Open3D shown in Figure 9. The ability to view segmentation and matching online significantly enhances the usability of the framework.

The visualiser built has the following capabilities.

- **Incremental segmented Point Clouds**: Individual segments in the point cloud are color-coded for better differentiation.
- **Local Map**
- **Source and Target Segment Matches:** Corresponding matches are displayed using the same unique colour, providing clear visual feedback on detected matches.
- **Robot Trajectory and Point of Segmentation:**
- **Real-time rendering**

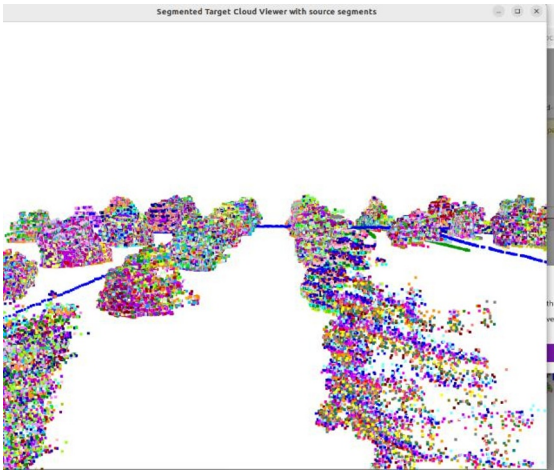

Fig. 8. Interactive Aru-SegMap viewer showing segmented point clouds and robot trajectory in the CitrusFarm dataset.

5 Conclusion

SegMap has achieved success in place recognition problems for urban areas and showed exciting potential to solve localisation problems in urban and agricultural environments. Even though the segment-based (vs semantic) approach shows capabilities to localise in a structure-poor environment lacking distinct objects, SegMap was built for ROS1 and is connected to other deprecated libraries, limiting its use and reducing the ability to extend or maintain. This work adapted SegMap into a standalone C++ implementation Aru-SegMap, removing ROS dependencies completely while preserving the core functionality as tested. Aru_SegMap is functional, produces consistent segments and provides reliable localisation. Future work includes adding ROS2 nodes for easy integration onto the robot architecture to deploy the work onto the physical platform. The output will be used as the localiser to a navigation system.

References

1. G., Tinchev, S., Nobili, M., Fallon, M. *Seeing the Wood for the Trees: Reliable Localisation in Urban and Natural Environments*, in Proceedings of 2018 IEEE/RSJ International Conference for Robots and Systems (IROS) (2018)
2. R., Dube, D., Cramariuc, H., Dugas, M., Sommer, J., Dymczyk, R., Siegwart, C., Cadena, C. SegMap **39**, **IJRR** (2020)
3. H., Gan, W., Lee, *Development of a Navigation System for a Smart Farm,* 6th IFAC Conference on Bio-Robotics BIOROBOTICS (2018)
4. J., Wenzel, J., Rhinelander, D., Moldovan*, pybind11 — Seamless operability between C++11 and Python, https://github.com/pybind/pybind11 (2016)*
5. S., Gupta, T., Guadagnino, B., Mersch, I., Vizzo, C., Stachniss*, Effectively Detecting Loop Closures using Point Cloud Density Maps*, in Proceedings of IEEE International Conference on Robotics and Automation (ICRA) (2024)
6. K., Purdon, J., Dickens, W., de Ronde, K., Ramruthan, G. Grafford, *Voyager, a ground-based mobile robotic platform for research development*, in Proceedings of 2023 RAPDASA-RobMech-PRASA-AMI Conference (2023)
7. H., Teng, Y., Wang, X., Song, K., Xiaoao, *Multimodal Dataset for Localization, Mapping and Crop Monitoring in Citrus Tree Farms*, International Symposium on Visual Computing (2023)