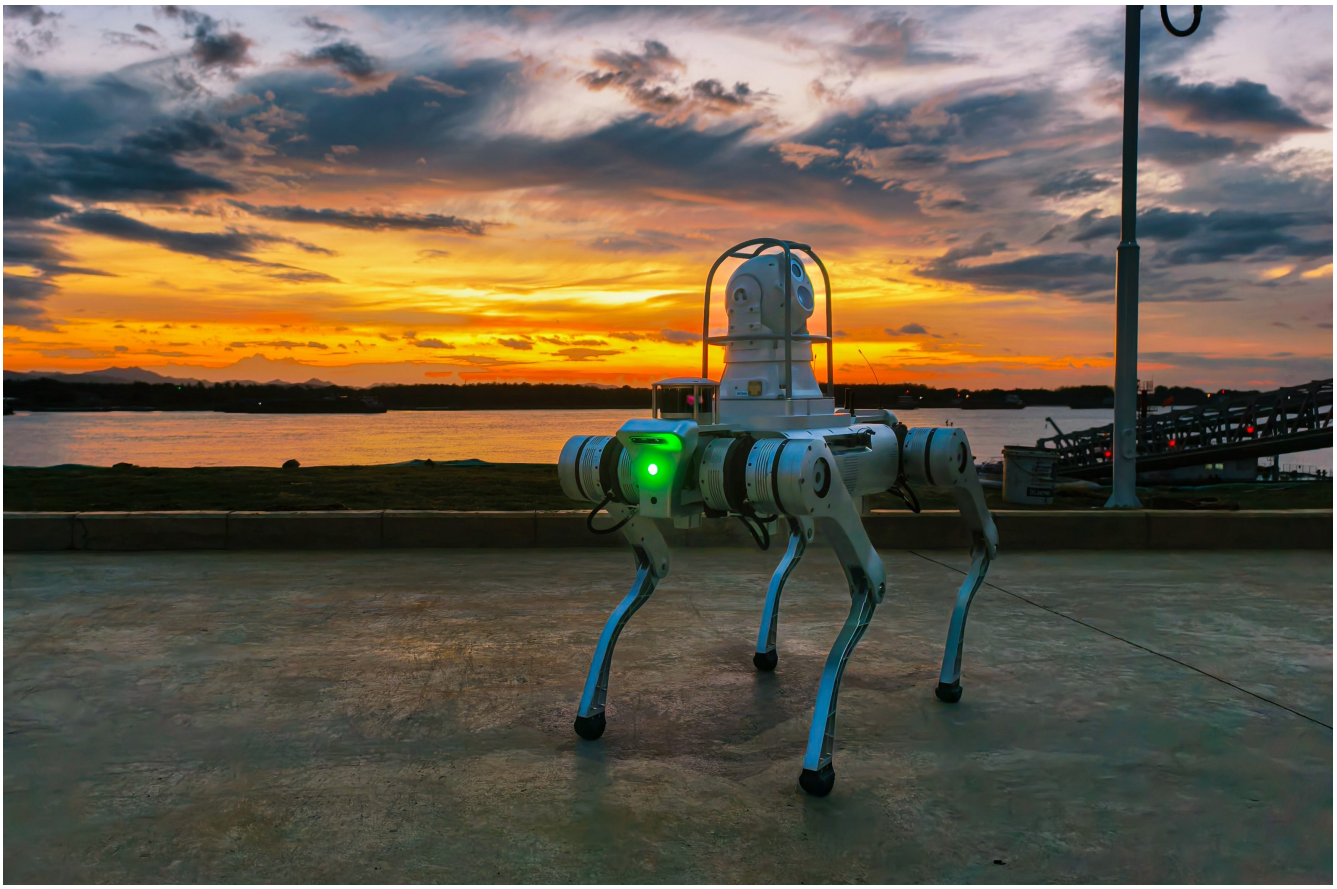
# Engineering Assistant Internship

***Linxai Technologies Co. Ltd.***

## Enhancing Graph-Based SLAM through Inverse Kinematics Integration



**Léon Perruchot-Triboulet**
Engineering Student in Autonomous Robotics
leon.perruchot-triboulet@ensta.fr

ROB 26 – May-August 2025

## Abstract

This report offers a comprehensive overview of my internship as an engineering assistant at Linxai Tech (灵锶智能), a robotics firm based in Shenzhen, China. This four month internship was centered around improvements of online Simultaneous Localization And Mapping (SLAM) algorithms for a quadruped robot platform. Over the course of this internship, I had the opportunity to experiment with various techniques for sensor data fusion to achieve robust localization and mapping in previoulsy unknown environments.
By integrating inertial measurements with LiDAR and joint encoder data over a graph based SLAM framework, I was able to achieve accurate position estimation robust to small sensor noise.
The proposed algorithms had to have as small a computational footprint as possible to be able to run alongside other programs needed for the robot to perform its different tasks.

**Keywords :** *Sensor fusion, SLAM, LiDAR, Inverse Kinematics, factor graph*

## Résumé

Le présent rapport présente une vue d'ensemble du stage d'assistant ingénieur effectué au sein de Linxai Tech (灵锶智能), une entreprise de robotique basée à Shenzhen, en Chine. Ce stage de quatre mois a porté sur l'amélioration d'algorithmes de localisation et cartographie simultanées (SLAM) appliqués à un robot quadrupède. Au cours du stage, j'ai exploré différentes techniques de fusion de données capteurs afin d'obtenir une localisation et une cartographie robustes dans des environnements inconnus du robot. En fusionnant des mesures inertielles avec des données issues du LiDAR et des encodeurs au sein d'un algorithme de SLAM par optimisation de graphe, j'ai pu obtenir une estimation de position précise et robuste au bruit des capteurs. Les algorithmes développés ont été conçus pour avoir une empreinte computationnelle minimale, permettant une exécution en temps réel en parallèle avec d'autres processus nécessaires au bon fonctionnement du robot.

**Mot-clefs :** *Fusion de capteurs, SLAM, LiDAR, Cinématique Inverse, graphe de facteurs*

## Acknowlegments

I would like to express my gratitude to Mr. Xiao Kai (肖恺) for giving me the opportunity to undertake this internship at his company. Thanks to him I have developed knowledge and skills that will benefit me throughout my career. I also would like to thank Liang Ziyang (梁子洋), my supervisor, for his guidance during this internship. Finally a warm thank you to every colleague whose warm welcome and support helped make these four months an enlightening experience.

# Contents

# Introduction

## Engineering assistant internship at ENSTA

ENSTA Bretagne's engineer cursus covers the last year of the Bachelor's degree and both years of the Master's degree. The Engineering assistant internship takes place one year after the specialization in autonomous robotics in between the two years of the Master's degree.

The goal of the Engineering assistant internship in the first year of the Master's program is to allow the student to confront their knowledge with the activities of the internationalized industrial and technological sectors corresponding to the openings in the program and to put this knowledge into perspective for the final phase of professionalization that constitutes the third and final year of the program.

## Linxai Intelligent Co. Ltd.

Linxai Intelligent Co. Ltd. (广东灵锶) is a robotics company founded in 2023 and headquartered in Shenzhen, Guangdong Province, China. Expanding rapidly, it currently operates two offices in the Longhua District: one dedicated to operations, mechanical design, and software development, and another focused on manufacturing, assembly, and testing. The company employs over 50 people and keeps growing.

Linxai specializes in the development of quadruped robots, with particular emphasis on payload capacity and intelligent behaviours. Research and development efforts focus on integrating advanced artificial intelligence algorithms for both motion control and perception. The company's long-term objective is to build robust robots capable of autonomous navigation and complex task execution in both structured and unstructured environments.

Figure 1: Galaxy Twin towers, Linxai headquarters in Shenzhen, China

## Robot localization

One of the fundamental challenges in mobile robotics is the ability of a robot to determine its position and orientation within an environment. This problem, known as localization, is essential for enabling autonomous navigation, obstacle avoidance, and interaction with the surroundings.

The field of Simultaneous Localization and Mapping (SLAM) emerged in the late 1980s and early 1990s as a response to this challenge [1]. The key idea of SLAM is that a robot must build a map of an unknown environment while simultaneously estimating its own trajectory within that map. Early solutions relied heavily on probabilistic methods, with the Extended Kalman Filter (EKF) and Particle Filters becoming standard tools. Over the past two decades, SLAM has advanced significantly with the integration of laser scanners, cameras, and more recently, deep learning techniques [2][3].

SLAM is crucial because it allows robots to operate in environments where prior maps are unavailable or unreliable. Applications extend from autonomous vehicles and drones to service robots and, as in this internship, quadruped robots. The real time ability to robustly localize and map directly impacts the autonomy, efficiency, and safety of robotic systems and thus makes it a priority.

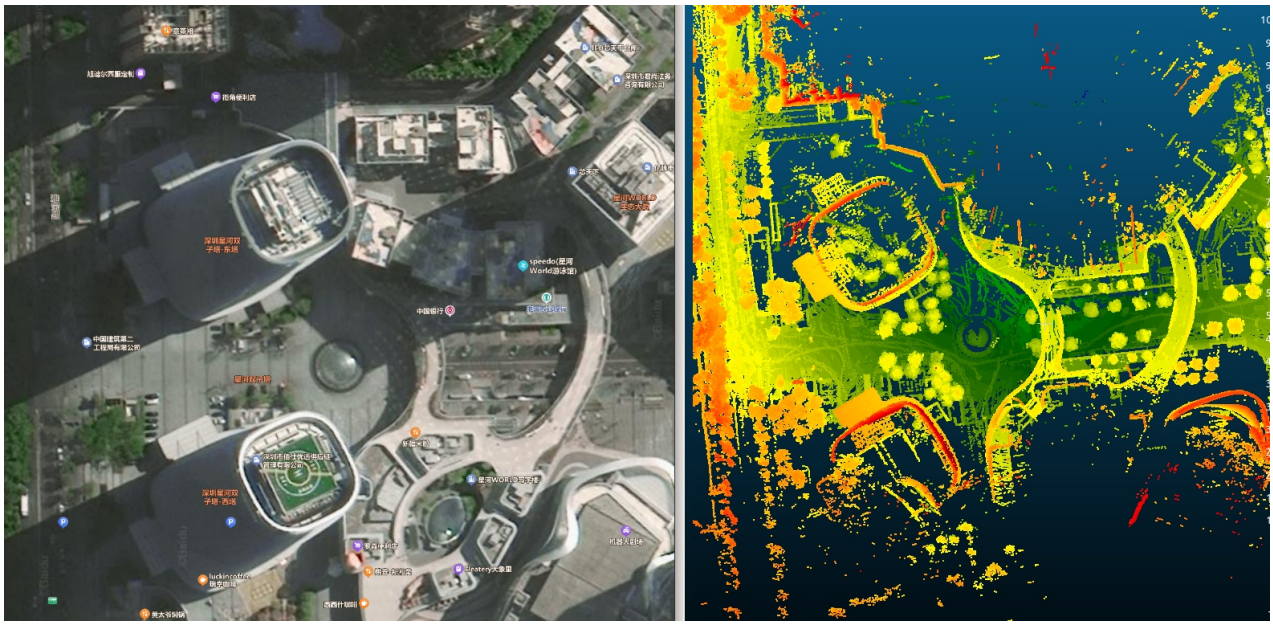


Figure 2: Comparing a satellite image with a mapped environment

# 1 Context

## 1.1 The D50 quadruped robot

The flagship product developed by Linxai is the D50, a quadruped robot weighing 65 kg. It is designed to carry a nominal payload of 50 kg, with a maximum capacity of 100 kg. The platform was conceived with modularity in mind, enabling it to adapt to a wide range of missions in both structured and unstructured environments, including flat and rough terrain.

The D50 is equipped with an extensive sensor suite to support state estimation and navigation. This includes an Inertial Measurement Unit (IMU), a 3D LiDAR, two pairs of RGB and depth cameras mounted at the front and rear of the chassis, and joint encoders for leg odometry. Such configuration provides redundant sources of information, which can be leverage to correct for each sensor's own bias and faults.

While Linxai's primary focus is on legged robotics, the company has recently initiated the development of a wheeled–quadruped hybrid platform, replacing the rubber feet with wheels. This architecture aims to combine the adaptability of legged robots with the efficiency and agility of wheeled locomotion. For further technical details on this topic, readers are referred to the work of Astrid BARAU and Rémi GENOUX-LUBAIN, their complementary reports originate from internships undergone parallel to this one.

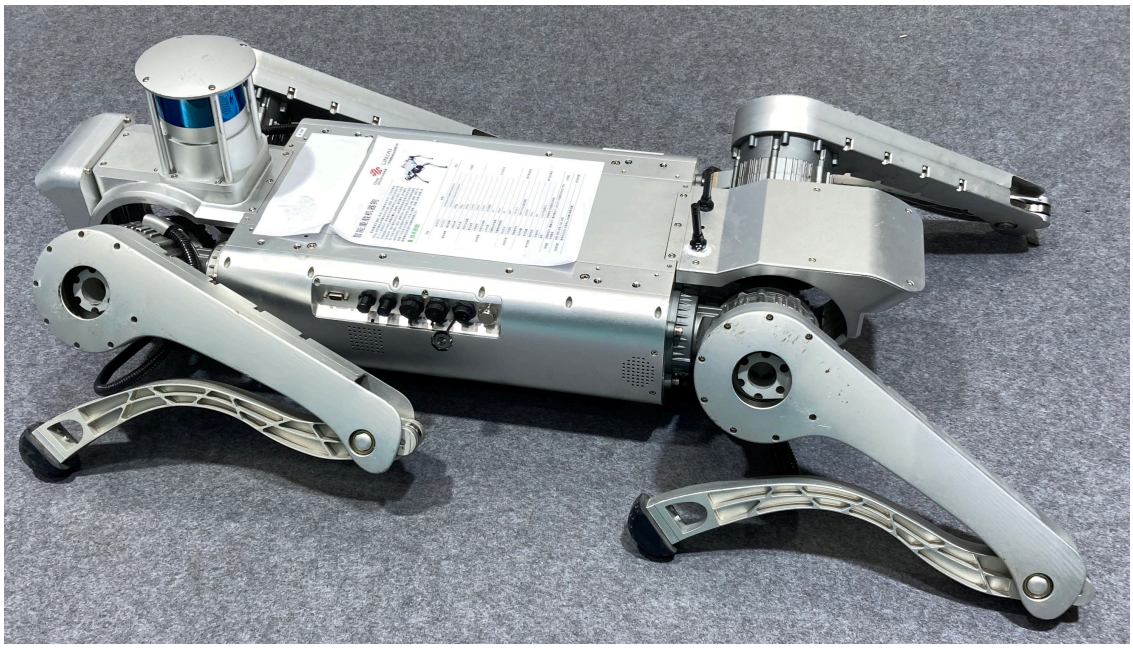


Figure 3: An early version of the D50

## 1.2 Objectives of the internship

Accurate state estimation is a fundamental prerequisite for autonomous robots. Inertial Measurement Units (IMUs) provide reliable short-term motion estimates, but their inherent drift necessitates fusion with additional sensors for long-term localization. For mobile robots designed in part for inspection tasks, environmental mapping is equally important. Modern SLAM (Simultaneous Localization and Mapping) systems typically address this by combining LiDAR and IMU data to produce local maps of the surroundings. However, these

methods degrade in perceptually deficient environments (e.g., tunnels or repetitive corridors) where feature matching becomes ambiguous and loop closures are scarce.

For the D50 platform, which currently lacks GNSS capabilities, minimizing drift is particularly critical. Legged robots offer a unique advantage in this regard: their locomotion inherently encodes motion information through joint angles and foot–ground contact patterns. This proprioceptive odometry can be exploited as a complementary data source, providing robustness in scenarios where exteroceptive sensors underperform.

The primary objective of this internship was to integrate leg odometry into the D50's SLAM pipeline in order to improve both accuracy and drift resistance. Beyond this initial objective, I also had the opportunity to investigate additional strategies for refining the localization workflow.

# 2 Sensor fusion with an Extended Kalman Filter (EKF)

## 2.1 Sensor suite overview

The D50 quadruped robot is equipped with a diverse set of sensors that support both locomotion and perception tasks:

— One Inertial Measurement Unit (IMU)
— One 3D LiDAR
— Twelve joint encoders (3 per limb)
— Two RGB-D cameras (front-mounted in the head and rear-mounted in the tail)
— Two RGB cameras (front and rear)

These sensors provide complementary information that is concurrently exploited by multiple functional modules of the robot. To manage acquisition, processing, and inter-process communication, the D50 relies on the Robot Operating System (ROS). ROS's modular node architecture facilitates rapid prototyping, streamlined integration of new algorithms, and robust communication between modules. In particular, ROS ensures reliable data transmission and prevents unnecessary duplication of sensor processing.

To further optimize performance, the D50 implements a dedicated sensor pipeline designed to minimize redundant computations and synchronize data across modules. However at the time of the redaction of this report some processes still compute redundant information. For instance the motion control stack uses a Reinforcment Learning policy to generate joint commands, from propriocetive data (IMU and joint encoders) and do not rely on other sensors. During this process, the neural network may internaly infer information about the robot's environment that isn't being outputed and needs to be computed elsewhere. This would be the case of a terrain roughness estimation, that could be leveraged by the localization stack to adapt its parameters.

Figure 4 illustrates how raw sensor streams are processed and fused to generate high-level information used by the localization and navigation stacks.

## 2.2 Odometry sources on the D50

To perform localization, the D50 robot leverages multiple sources of odometry that provide insights into its position using both intrinsic and extrinsic information. At the time of this internship, the most accurate position estimate is obtained from a tightly coupled LiDAR–IMU odometry source, which will be described in detail in section 3.1. In addition, an inverse kinematics (IK) based odometry is computed to satisfy the

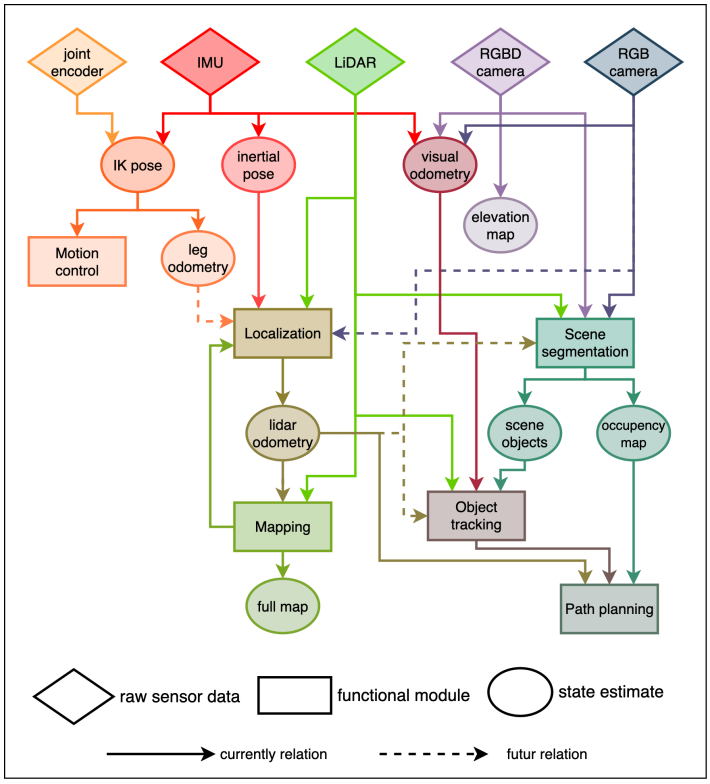



Figure 4: Simplified relationship between sensors and the generated data streams

requirements of the robot's classical motion control algorithms. Each odometry source has distinct advantages and limitations, which can be exploited to achieve a more robust overall estimation.

LiDAR odometry provides reliable 2D position estimates in environments that are relatively static and free of excessive noise or perceptual degeneracy (e.g., tunnels or feature-poor areas). While effective locally, LiDAR odometry may accumulate errors in elevation over time. Although this has minimal impact on short-range navigation, it can lead to drift in global maps over long missions. However, its integration with loop closures or complementary data sources can significantly improve precision and enable consistent environmental mapping.

Leg odometry, derived from joint encoders and the robot's kinematics, provides a reliable measure of elevation and complements the 2D LiDAR measurements. However, it is subject to constant drift due to encoder inaccuracies and foot–ground slippage. This report will not cover the extent of the leg odometry pipeline although some insights over potential improvements will be proposed.

By fusing LiDAR–IMU odometry with leg-based odometry, it is possible to obtain more reliable 3D position estimates. Each source counterbalances the weaknesses of the other, combining the environmental awareness of LiDAR with the kinematic reliability of leg odometry.

### 2.3 Time synchronization challenges

To fuse LiDAR and leg odometry, the *robot_localization* ROS package [1] was initially employed. This package provides nonlinear state estimation via sensor fusion for an arbitrary number of sensors. Its usage is relatively straightforward, requiring only a properly configured parameter file to produce state estimates.

1. The package's wiki: http://wiki.ros.org/robot_localization and associated paper [4]

Although the package could run on data from the D50, the results were unsatisfactory due to message stuttering, which caused the EKF node to produce inconsistent estimates. Both leg odometry and LiDAR odometry are being published at approximately 500Hz, which exposed three issues:

- **Computational load:** The EKF struggles to maintain the desired 500Hz output rate due to the high computational demand of other processes concurrently running on the robot.
- **Measurement queue overflow:** Upon lags, incoming measurements accumulate in the filter's queue, eventually overflowing and reducing the reliability of predictions. Moreover at such frequency the queue quickly fills up.
- **Time synchronization errors:** High input frequency causes messages to interleave. While the package can process messages with old timestamps, doing so incurs additional processing overhead and may lead to inconsistencies when messages arrive out of order.

To mitigate these issues several solutions exist and were either found experimentaly by twiking the configuration file or by looking at robotic stack exhanges forum on the internet. Of these solutions the following have the most potential:

- Rebuilding the package with optimized compilation flags to improve EKF computational efficiency.
- Limiting the output rate of the filter to a more reasonable frequency.
- Downsampling input measurements to reduce queue overflow.

However, of these solutions only the second one was implemented. Indeed since the state estimates would not have been fed back into the odometry pipeline, they would have continued to drift, preventing the odometry from converging to a more accurate estimation. Consequently, it was decided to perform the IK-odometry fusion directly within the LiDAR odometry pipeline, ensuring that both odometry sources could leverage improved estimates in real time.

# 3 Graph based SLAM

## 3.1 Overview of LIO-SAM

The LiDAR odometry pipeline on the D50 is based on a fork of LIO-SAM [5]. LIO-SAM is a framework for tightly coupled LiDAR–inertial odometry via smoothing and mapping, enabling real-time trajectory estimation and accurate map-building. The system is built around two factor graphs, which allow the integration of multiple relative and absolute measurements, including loop closures and GNSS, as factors in the optimization. The overall LIO-SAM framework, is illustrated in Figure 5.
LIO-SAM operates two parallel factor graphs to facilitate real-time sensor fusion:

**IMU Preintegration Graph:** This graph processes inertial measurements to provide an initial guess for the LiDAR odometry optimization. The associated ROS node preprocesses LiDAR point clouds. It runs at approximately 500Hz, offering continuous, up-to-date state estimates from the latest lidar odometry calculations. The state estimates produced by this factor graph are being fed into the other factor graph as the red factors in Figure 5

**Map Optimization Graph:** Operating at a lower frequency (around 20Hz), this graph generates LiDAR odometry by performing point cloud scan matching on keyframes within a sliding window marginalizing older scans. This approach balances computational efficiency with high-accuracy mapping. This graph is also responsible for loop-closure matching as well as GNSS processing (when equipped on a robot). Figure 5 represents the overall Map Optimization factor graph.

Factor graph construction and optimization is handled using GTSAM (Georgia Tech Smoothing and Mapping library) [6] and iSAM2's Levenberg-Marquardt optimizer. GTSAM provides built-in factor types for standard sensor models while also allowing the creation of custom factors to represent specific sensor constraints. iSAM2 performs incremental smoothing, updating the factor graph efficiently as new measurements arrive, which is suitable for real-time applications.
A detailed explanation of factor graphs is beyond the scope of this report. For more information about the underlying theory, properties, and applications of factor graphs please refer to this book[7].

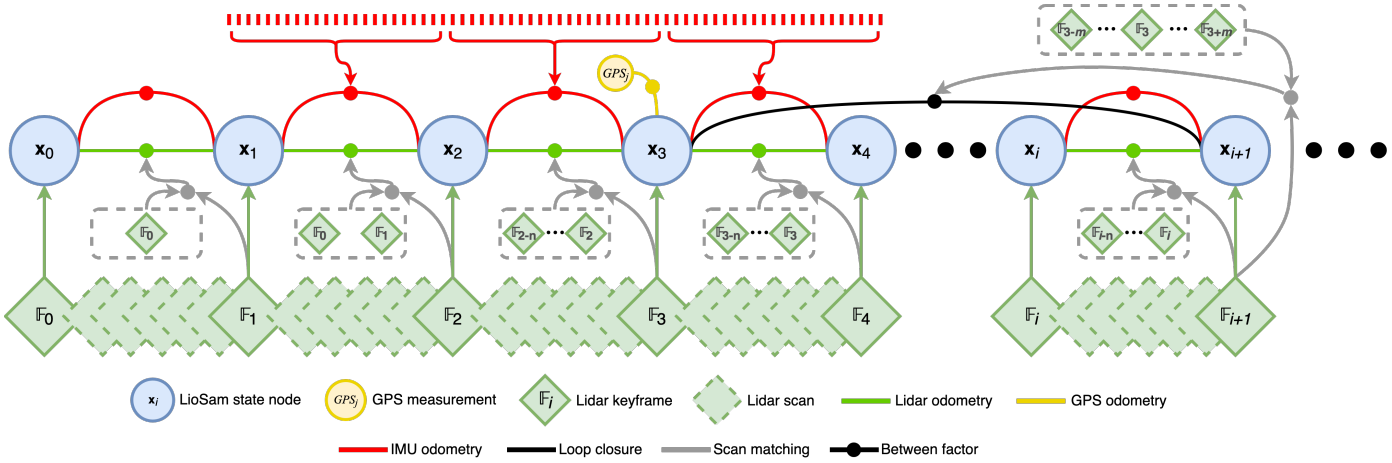



Figure 5: The system structure of LIO-SAM from the original paper [5]

## 3.2 Fusing leg odometry

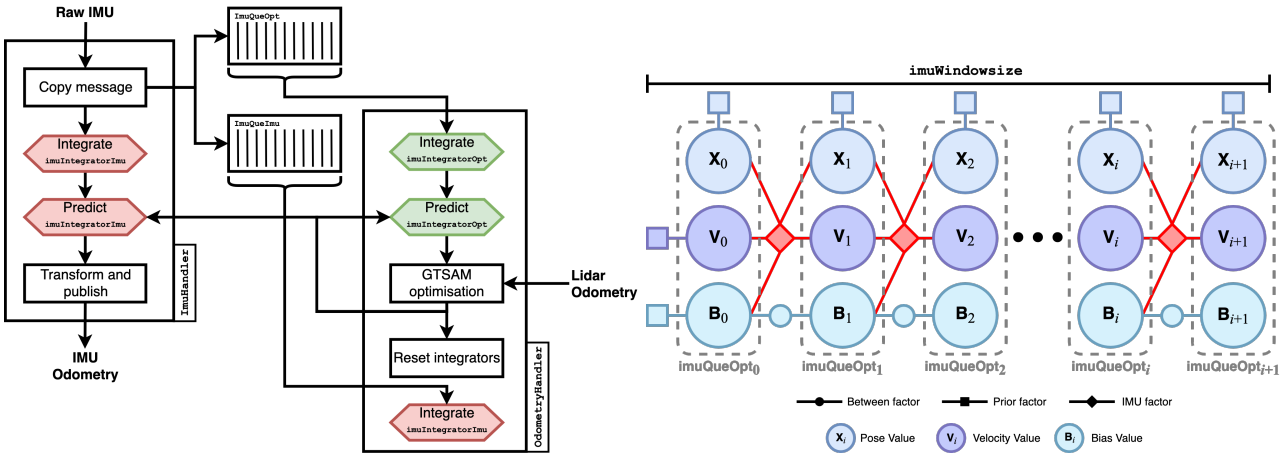
Since LIO-SAM uses two parallel factor graphs, each processing different sensors as input factors, there is a strategic design choice to be considered as to where IK-odometry should be integrated into the pipeline.

### 3.2.1 High-frequency fusion

Since both the IMU and IK odometry pipeline operate at approximately the same frequency, it initially appears wise to fuse leg odometry into the IMU Preintegration factor graph. Moreover this factor graph is responsible for real-time state estimation on the D50, this approach would then allow for leg odometry to contribute live updates on the best state estimate.

The IMU preintegration algorithmic scheme is illustrated in Figure 6a. The IMU preintegration algorithmic scheme processes high-frequency IMU data to estimate the robot's state. The process begins with raw IMU measurements that get constantly integrated and publihsed on a ROS topic as illustrated by the *IMUHandler* thread on the lefthand side of the figure. These raw measurements are also added into two queues until they fill up. Once the queues are full, the *odometryHandler* thread (on the righthand side of the figure) integrates all stored measurements into a new integrator which then feeds its prediction to LIO-SAM's high frequency factor graph as a prior estimate. This factor graph also takes in the latest LiDAR odometry and is then optimized to produce the best state estimate. Once this optimized state is found both integrators are reset and the live integrator (from the *IMUHandler* thread) is updated with the new state and reintegrates the queued measurements as to start from a better initial guess.

As for the factor graph, illustrated in Figure 6b it consists of three factor types and three value types. The value types are the position, velocity and IMU bias at each timestep from the queue. In between two successive




(a) IMU preintegration algorithmic scheme (b) IMU preintegration factor graph

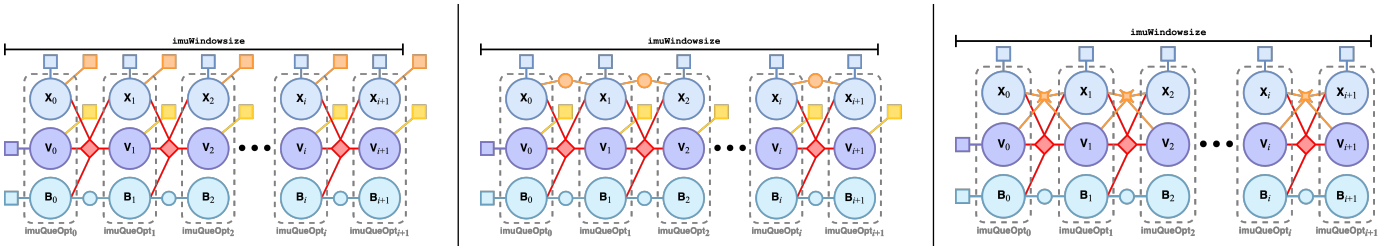
Figure 6: High frequency preintegration scheme and factor graph

measurements are linked with a special IMU factor that takes into account the IMU noie and bias. Additionally successive bias are linked with a between factor provided by the integrator. Finally for each timestep, the position takes in the prediction from the integrator as a prior estimate.

This complex scheme allows for a very accurate high-frequency state estimate. To fuse leg odometry into this scheme multiple approaches were considered. Firstly, leg odoemtry could be fused into a new factor gaph just before publishing the live odometry in the *IMUHandler* thread. However this would have had required to instantiate a new factor graph and optimizer and would not have allowed for the integrators from benefiting of the improved accuracy. Thus it was decided to fuse leg odometry into the existing factor graph of the *odometryHandler* thread.

For leg odometry integration, three factor types were considered:
— **Between Factor:** Linking two nodes with successive pose odometry.
— **Prior Factor:** Assumes that the node should hold a given value to a given covariance.
— **Custom Factor:** Creating a special constraint linking successive position and velocity estimates based on kinematics constraints.

Combinations of these factors were applied to generate the factor graphs from Figure 7




(a) Prior factors (b) Prior and between factors (c) Custom factor

Figure 7: High frequency factor graph modifications to include leg odometry

Of these three proposed factor graphs, only the first two were implemented and tested on log mission data (rosbag). However even after tuning the factors parameters the improvment were not substantial enough to justify further high-frequency experimentation.

This approach also presented two main challenges:

— Synchronization issues: High-frequency messages from IMU and leg odometry may arrive at slightly different timestamps. This was addressed by maintaining a fixed-size queue of recent leg odometry values. The IMU measurement is given priority, and leg odometry is interpolated between two measurements bracketing the current IMU timestamp, ensuring a coherent factor graph.
— Computational cost: Adding leg odometry to the high-frequency graph increases the factor graph size, raising computational demands. As additional systems are implemented on the D50, preserving computational efficiency for state estimation becomes critical.

### 3.2.2 Low-frequency fusion

An alternative strategy for integrating leg odometry is to operate at a lower frequency, alongside other factors such as LiDAR scan matching, loop closures, and, in future implementations, GNSS. Within this framework, leg odometry can either be directly fused onto the LiDAR odometry values or added into a separate chain of values.

When fused directly, leg odometry can be applied as either a Between factor, representing relative motion between successive poses, or as a Prior factor, constraining individual pose estimates. Measurements are interpolated to match the appropriate timestamps, using the same scheme employed in high-frequency fusion.

In the parallel lane approach, leg odometry is represented explicitly, with Between or Prior factors applied to its nodes. A zero-valued Between factor links the leg odometry lane to the LiDAR lane, allowing the two to inform one another while maintaining distinct representations. Initial tests of these configurations demonstrated superior qualitative performance compared with the high-frequency fusion approach.

These different approaches lead to four different factor graph configarations illustrated in Figure 8.

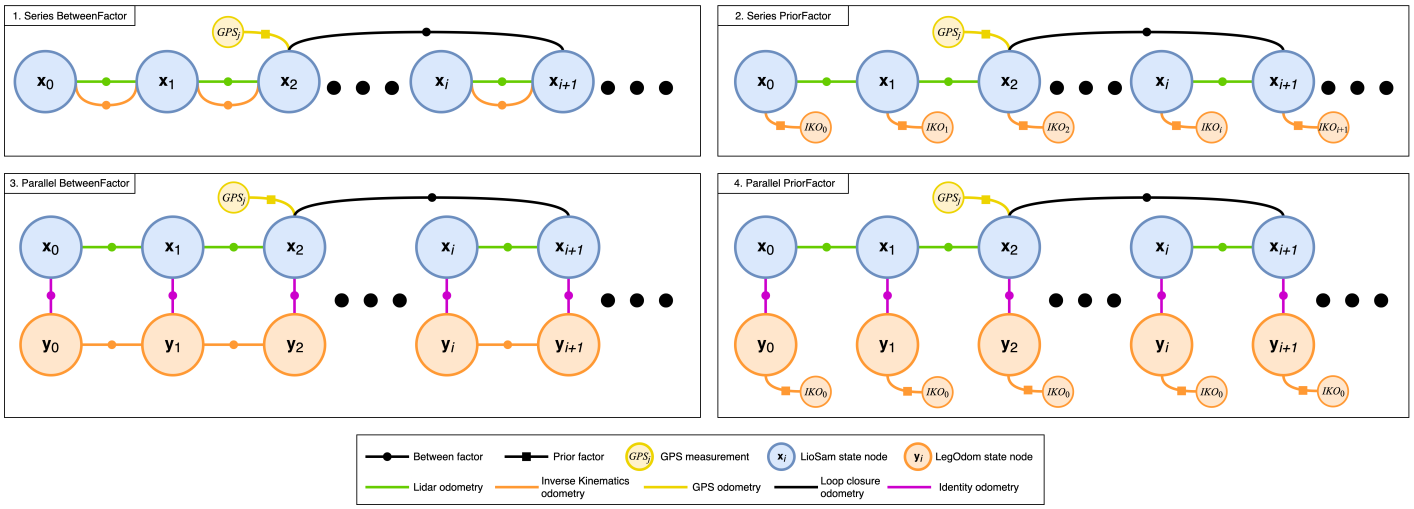



Figure 8: Possible factor graph configurations for low-frequency leg odometry fusion

Ultimately, a hybrid solution was adopted. A Between factor between two LiDAR nodes constrains relative motion using leg odometry. Simultaneously, a Prior factor is applied to the parallel leg odometry lane, linked to the LiDAR lane via an identity factor. This combination allows for the system to leverage leg odometry both for relative motion (which IK provides a good estimation of) and absolute state estimation (mainly for elevation w.r.t the ground), improving elevation estimation and enhancing overall state accuracy.

Preliminary qualitative results indicate a visible reduction in global positional drift over long trajectories,

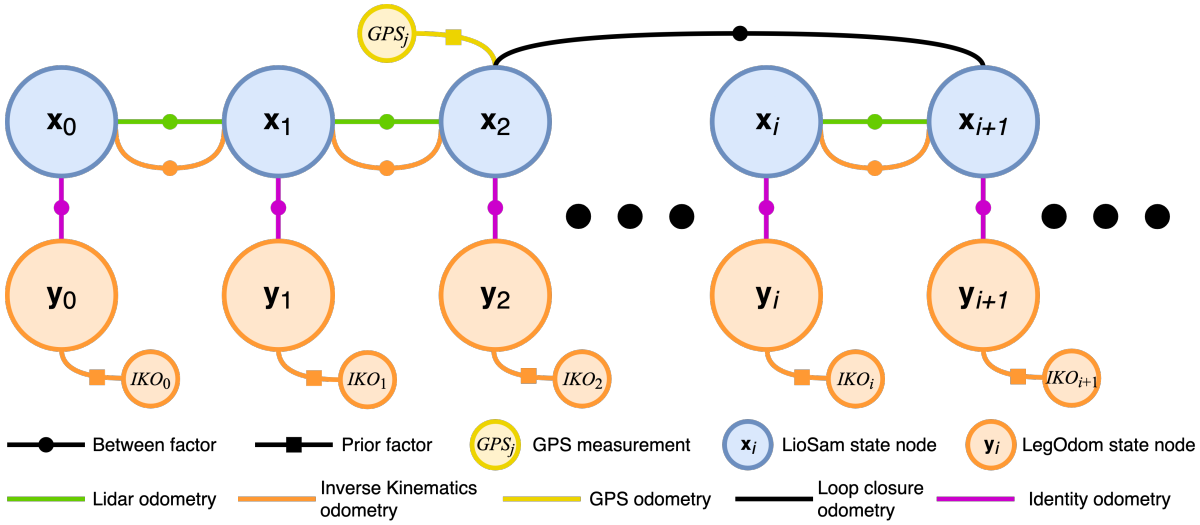



Figure 9: hybrid factor graph configuration for leg odometry fusion

improved elevation accuracy, and more consistent state estimates in environments with sparse LiDAR features or repetitive structures. Quantitatively, while the drift in the XY plane remained around 2 meters by the end of a 30 minutes mission, the elevation drift was reduced to only 20 centimeters. Furthermore, although the real environment varied just by a few meters in elevation along the robot's trajectory, the same dataset when processed without leg odometry exhibited artificial terrain variations of up to 30 meters. This effect is almost completely eliminated when using the hybrid factor graph approach. Figure 10 illustrates the effect of a map being distorted by drift in the absence of leg odometry. Please note that this is not the actual dataset on which the problem was observed, but it serves to illustrate the issue.

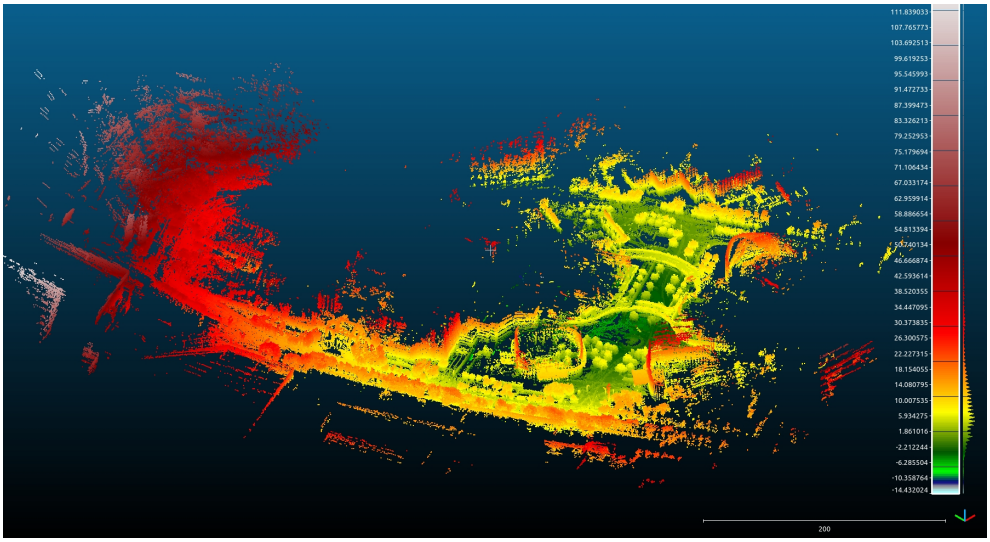


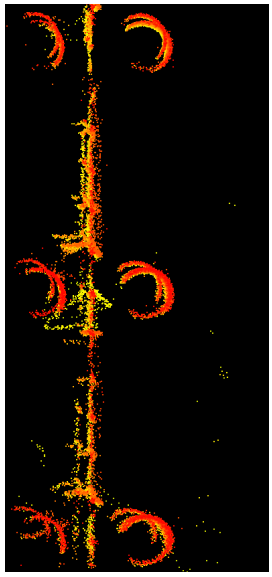
Figure 10: Map wrapping due to elevation drift without leg odometry (color represents height)

### 3.3 Exploiting loop closures

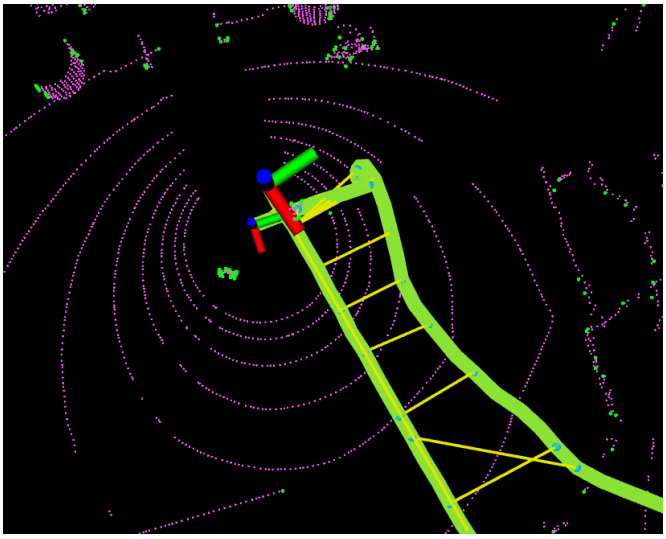
During its operations, the D50 may revisit locations it has previously traversed. In such situations, loop closure can be used to improve state estimation by retroactively correcting accumulated drift. Implementing loop closure involves three key steps: identifying the loop, estimating the relative transformation between the two poses, and incorporating this constraint into the factor graph.

For loop detection, the D50 uses Scan Context[8], a global descriptor derived from 3D LiDAR scans. Scan Context constructs a 2D bird's-eye polar image representing the highest point in each spatial bin and compares these images using a distance metric to determine similarity between scans. Once a candidate match is identified, Scan Context provides an initial estimate of the rotational and translational differences between the two scans. This estimate is then refined using the scan matching algorithms in LIO-SAM.

Once the optimal alignment is determined, a new factor is added to the Map Optimization factor graph, linking the two corresponding pose nodes. Owing to the factor graph structure, subsequent graph optimizations can retroactively adjust the robot's trajectory, thereby improving the overall accuracy of the estimated path. Figure 11 illustrates the effect of loop closure on the D50's trajectory over a 30 minute mission. As shown in Figure 11a, without loop closure, misalignments accumulates over time and are particularly noticable on the columns of the starting building. In contrast Figure 11b demonstrates the improved alignment achieved through loop closure, with scan matching and loop closure factors highlighted in yellow. The remaining misalignment is mainly attributed to other sources of drift introduced by the current pipeline, which could be mitigated by the improvements discussed later.



(a) Without loop closure the columns don't align.



(b) Scan matching and loop closure factors visualized in yellow.

Figure 11: Comparing the effect of loop closure on a 30-minute mission.

To reduce computational load, loop closure is only triggered under plausible conditions. Specifically, it is applied when the robot's estimated position is within a predefined radius of a previously visited location and when that location was reached a sufficient time in the past. This strategy ensures that loop closure is both effective and computationally efficient.

### 3.4 Enhancing scan matching

A core component of LIO-SAM's visual odometry is its scan matching capability. Using the inertial state estimate, the algorithm generates an initial guess of the robot's motion between two keyframes. This estimate serves as a starting point for feature matching between successive point clouds. The latest scan is compared to previous scans that have already been aligned in the map frame, allowing the robot to more accurately estimate its movement. As described earlier, this odometry can then be fused with other sensor data in the factor graph to improve the global state estimate.

However, successful scan matching requires distinctive visual features in the environment. In LIO-SAM, each scan is preprocessed to extract planar and edge features from each LiDAR ring (respectively purple and green dots in Figure 11b). These features are then matched to those of previous scans using iterative closest point (ICP) methods organized in a KD-tree structure.

Scan matching can fail under several conditions. In degenerate environments, the algorithm may either find too few features or encounter features that are too similar, making it difficult to distinguish scans from physically different locations. Noisy environments, such as areas with foliage or irregular surfaces, present another challenge because noisy surface normals degrade the matching process. Finally, dynamic objects moving within the scene can distort perceived objects. On the D50' s operational environments, the latter two issues noisy surfaces and moving objects are the primary concerns.


(a) Fences oclude the scene behind them and create noisy features

(b) Foliage and scooters are inconsistent perceived across successive scans

Figure 12: Perpceptually noisy environments encountered on the D50 while performing scan matching

To address noisy features, LIO-SAM's matching parameters were tuned during the internship. This paramter optimization improved stability and reduced the likelihood of alignment failures. An alternative strategy, though not implemented during the internship, would be to discard LiDAR scans acquired in feature poor environments and rely temporarily on IMU and leg odometry for state estimation.

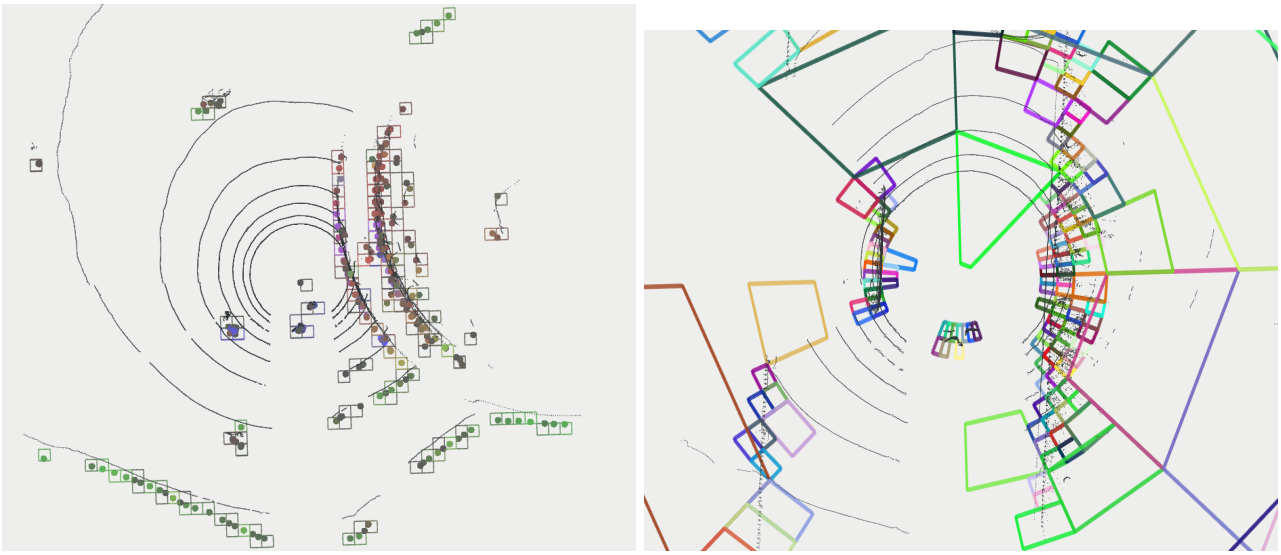
To handle moving objects in the scene, a more sophisticated approach is required. Dynamic objects can introduce significant errors in scan matching, as they may be incorrectly matched to static structures or distort the perceived geometry of the scene. Several methods have been proposed to mitigate this issue, which can be broadly categorized into three groups: point cloud-based methods, image processing methods, and neural network approaches.

### 3.4.1 Point cloud based methods

A first attempt involved a straightforward point cloud segmentation approach, where the raw LiDAR data were segmented directly. To make this process more efficient, the point cloud was discretized into a voxel grid, reducing the computational cost of subsequent clustering and classification steps. Two voxelization structures were evaluated: a conventional cubic octree and a cylindrical octree. While the cubic octree is simpler and faster to implement, the cylindrical structure better leverages the natural sparsity of LiDAR data (its cell area increasing with range) maintaining roughly uniform point density across distances.

Once constructed, the octree was linked to a KD-tree to enable efficient nearest-neighbor searches between cells and to cluster points in a meaningful way. Node subdivision within the octree was controlled by several rules, each tested to assess the resulting number of cells. The first rule enforced full subdivision up to a fixed resolution along each axis (X and Y for the cubic octree, and range for the cylindrical one). This method achieved up to a tenfold reduction in cloud size, depending on the environment. A second rule used the standard deviation of point positions within each cell along the octree axis to guide subdivision—allowing the algorithm to separate points belonging to distinct objects that might otherwise fall within the same voxel.

Point cloud based methods were explored primarily to assess the feasibility of the approach. While these methods showed promise for producing accurate segmentation, the complete pipelines were not fully implemented, as alternative techniques were also under investigation. Furthermore, the proposed segmentation operated on a static scan basis and did not incorporate temporal information, preventing effective tracking or identification of moving objects. Despite these limitations, these experiments provided valuable insights into the constraints of classical point cloud processing methods and motivated the exploration of point cloud projection techniques and neural network based approaches.



(a) Cubic octree voxelisation, (leaf division performed based on cell size)

(b) Cylindrical octree voxelisation (leaf division based on the number of points)

Figure 13: Birdeye view of cubic and cylindrical octree voxelisation of a LiDAR scan

### 3.4.2 Image processing methods

Many point cloud segmentation methods simplify the data representation by projecting the 3D point cloud onto a 2D plane, enabling the use of established image processing techniques. Although this projection inevitably

reduces spatial information, it offers significant advantages: the resulting representations are easier to process, and they allow the reuse of numerous well-developed methods from computer vision.

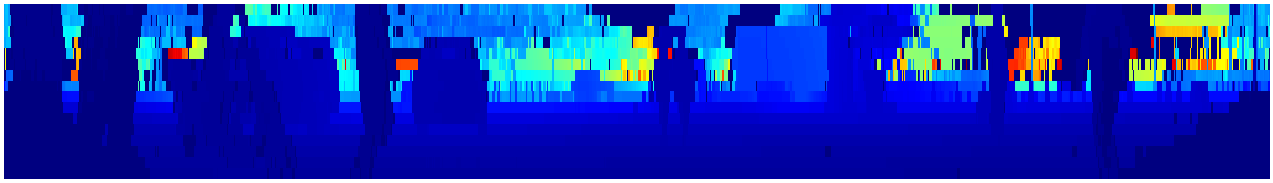
**Cylindrical range image projection :**

This projection maps each point in the LiDAR cloud onto a pixel in a two-dimensional image according to a cylindrical projection model. In cases where multiple 3D points project onto the same pixel, several strategies can be adopted: selecting the point closest to the sensor, choosing the one with the highest intensity value, or computing an average point. The latter approach, however, may introduce artificial structures that do not correspond to any real object in the scene.

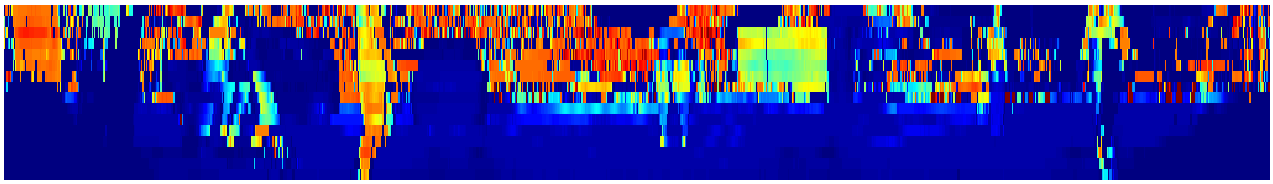
Once the range image is generated, various motion extraction techniques can be applied. A simple approach tested during the internship involved computing the absolute difference between two successive preprocessed images, followed by segmentation and region growing based on the identified pixels. The preprocessing stage included thresholding to exclude points that were either too close or too far from the robot since distant points are typically static, and near points often correspond to self detections and applying morphological closing operations to reduce image noise.

However, this method did not yield satisfactory results due to the high level of residual noise in the projected images. Classical image processing techniques alone were insufficient for the desired application, which was to mask out dynamic points in LiDAR scans to prevent their inclusion during scan matching. Additionally, the LiDAR sensor used on the D50 produced point clouds with 16 rings and 1024 points per ring, resulting in a range image with a resolution of 16×1024 pixels. While this provides adequate horizontal information, the limited vertical resolution makes object segmentation difficult and resulted in segmenting the image based on horizontal lines only. The envisioned strategy would have been to then match different scan lines to grow the observed objects. Figure 14 illustrates the images generated by the cylindrical projection, note that these figures have been streched vertically for better visual representation. One can identify pedestrians,trucks a scooter and a car.

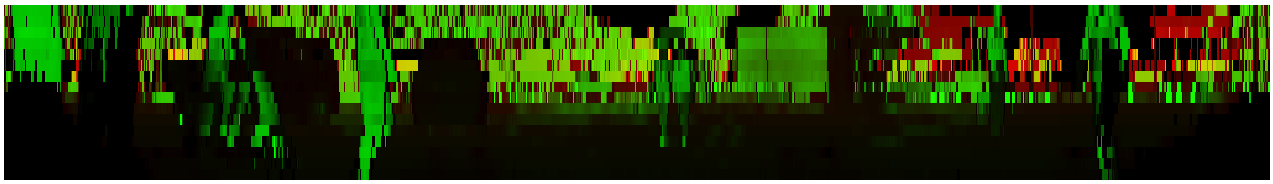
A possible improvement could have been to increase the vertical resolution through interpolation, although this approach was not implemented, as subsequent experiments focused on an alternative strategy.



(a) Cylindrical projection of range information (colormaped)



(b) Cylindrical projection of intensity information (colormaped)



(c) Mixed image of range (red channel), intensity (green channel), blue channel left empty

Figure 14: Cylindrical projection of a point cloud, streched vertically (original image 16×1024 pixels)

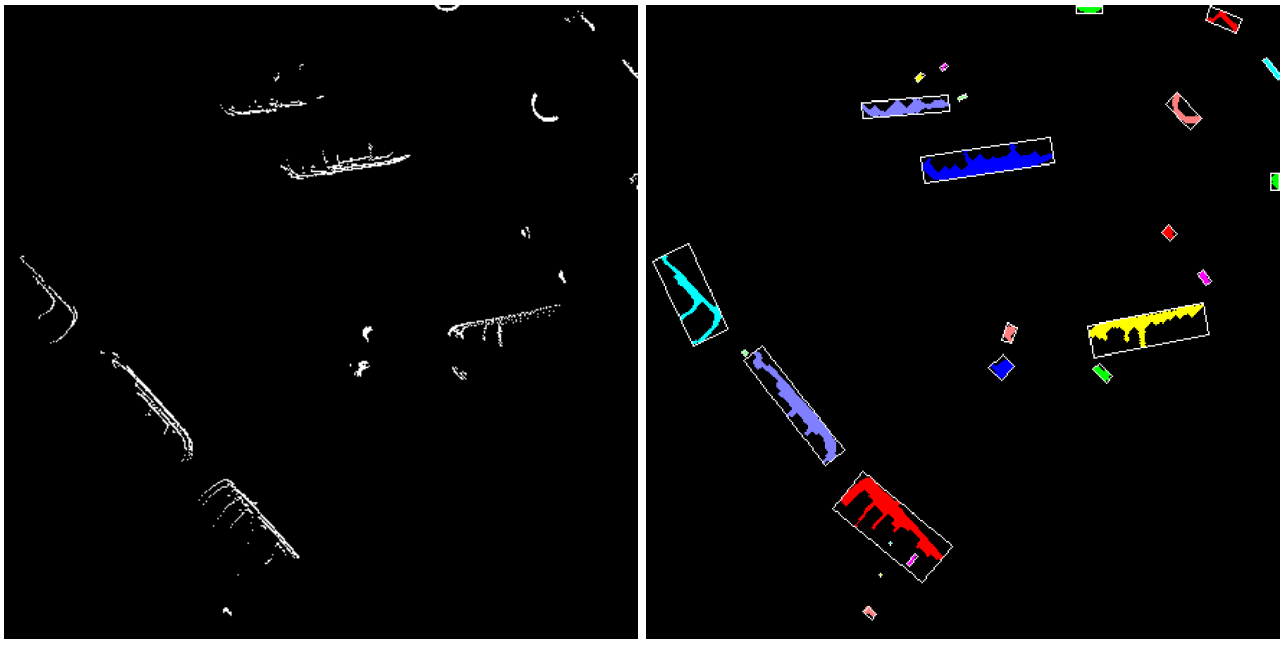
**Bird's eye view projection :**

This projection generates a top-down representation of the scene by projecting LiDAR points onto a horizontal plane. The resulting image can be refined through elevation-based thresholding to retain only points within a specific height range, thereby removing ground and high-altitude noise (e.g. ground ringing and tree foliage). Figure 15a illustrated the projection.

To produce this image, three different approaches were evaluated. The first involved sampling the point cloud into a uniform grid of fixed resolution within a defined maximum range along both the X and Y axes. All points within each cell were aggregated, and the corresponding pixel was assigned the median intensity of those points. This representation has the advantage of being visually interpretable and relatively sparse, which facilitates segmentation.

The processing pipeline applied to the generated image included morphological closing, clustering, removal of small blobs through thresholding, and computation of oriented bounding boxes for each detected cluster. These bounding boxes were then filtered according to aspect ratio, sparsity, and area to remove artifacts caused by scan rings (which can occur when the ground is uneven) as well as nonrelevant structures such as trees or parked bicycles. A box-growing step was subsequently applied to merge adjacent bounding boxes, followed by an additional clustering stage to combine objects that remained disconnected due to their presence on different scan rings. Figure 15b illustrates the segmented scene with objects being well clustered.

While this segmentation method provided an effective proof of concept, it presented several limitations. Due to inconsistent occlusions between consecutive scans, the bounding boxes did not always align perfectly with the underlying objects, complicating motion tracking when relying solely on centroid positions.



(a) Birdeye view of the raw point cloud (cars and pedestrians are visible)

(b) Segmented clusters after image processing (bounding boxes are drawn)

Figure 15: Birdeye view projection and segmentation of a LiDAR scan

Although the image processing approach initially appeared promising and relatively easy to implement, it ultimately proved less effective than expected. This outcome was primarily due to the high number of different methods tested without sufficient scientific grounding in the underlying algorithms. Moreover, the developed pipelines operated solely on individual scans and did not incorporate the temporal relationships between succes-

sive frames, an aspect that could have improved segmentation robustness by leveraging information accumulated over time.

### 3.4.3 Neural network approaches

Toward the end of the internship, the focus shifted to dynamic object segmentation using neural network approaches. Two relevant methods were identified for evaluation: MotionSeg3D[9] and PointNet2[10], specifically the full PyTorch implementation available here [2].

The first method relies on a cylindrical projection of the point cloud, performing segmentation using a convolutional neural network on the current range image and residual images of previous scans projected into the current frame. PointNet2, in contrast, operates directly on the raw point cloud, processing the input through a hierarchical feature learning architecture to produce a segmented point cloud with point-wise annotations. Since PointNet was originally developed for 3D object classification and indoor segmentation, adaptations would be required to apply it to outdoor environments and to train it on a dataset representative of the robot's operational conditions.

MotionSeg3D, a pretrained network, was also identified as suitable for this task, as it has been trained on a public dataset compatible with the intended application. The Semantic KITTI dataset[11] was also considered as a training dataset; it provides annotated LiDAR sequences from a European urban environment collected from a vehicle perspective. Although the data differ from the D50' s operational environment, it was anticipated that the networks could be adapted or serve as a proof of concept. Fine-tuning in simulation or manual labeling of ROS bags could further improve performance if direct deployment on the robot proved ineffective.

As this task was undertaken toward the end of the internship, time constraints limited the scope of the work. Efforts were primarily focused on preparing a training bridge for PointNet2 on the Semantic KITTI dataset, and as a result, the performance of both PointNet2 and MotionSeg3D could not be evaluated within the internship period.

2. Pytorch implementation of PointNet2: https://github.com/erikwijmans/Pointnet2_PyTorch

# Conclusion

The primary objective of this internship was to improve the state estimation and localization capabilities of the D50 quadruped robot by integrating leg odometry into its existing SLAM pipeline. Over the course of the internship, the work focused on analyzing the current LiDAR-IMU based pipeline, evaluating strategies for high and low frequency leg odometry fusion, implementing a hybrid low frequency fusion scheme, and addressing challenges related to scan matching and loop closures.

Key achievements include the successful design and implementation of a hybrid low frequency fusion approach, leveraging leg odometry both as a relative motion constraint and as a global prior to enhance elevation estimation. Preliminary tests demonstrated reduced global drift and improved robustness noisy environements. The internship also explored improvements in scan matching and loop closure strategies, contributing to more consistent global state estimation across extended trajectories.

Several directions for future work have been identified. These include rewriting the leg odometry pipeline to mitigate residual drift, integrating a terrain state estimator to enhance adaptation to ground conditions, and incorporating GNSS to provide a complementary global reference. Additional opportunities exist in neural network based dynamic segmentation modules.

This internship provided extensive hands on experience in robotic software development, sensor fusion, and SLAM algorithms, highlighting the practical challenges of real time state estimation for legged robots. It also underscored the importance of balancing computational efficiency with estimation accuracy in complex, multi sensor systems. Beyond technical contributions, the experience fostered a deeper understanding of industrial methodologies, iterative development, and system level integration in robotics.

Overall, this internship significantly strengthened both technical competence and understanding of autonomous robotic systems, laying a strong foundation for future work in perception, navigation, and autonomy.

## References


[1] Hamid Taheri and Zhao Chun Xia. « SLAM; definition and evolution ». In: *Engineering Applications of Artificial Intelligence* 97 (2021), p. 104032. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2020.104032. URL: https://www.sciencedirect.com/science/article/pii/S0952197620303092.

[2] Saad Mokssit et al. « Deep Learning Techniques for Visual SLAM: A Survey ». In: *IEEE Access* 11 (2023), pp. 20026–20050. DOI: 10.1109/ACCESS.2023.3249661.

[3] Wenda Wang et al. « Recent Advances in SLAM for Degraded Environments: A Review ». In: *IEEE Sensors Journal* 25.15 (2025), pp. 27898–27921. DOI: 10.1109/JSEN.2025.3584218.

[4] T. Moore and D. Stouch. « A Generalized Extended Kalman Filter Implementation for the Robot Operating System ». In: *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, July 2014. DOI: 10.1007/978-3-319-08338-4_25.

[5] Tixiao Shan et al. « LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping ». In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 5135–5142. DOI: 10.1109/IROS45743.2020.9341176.

[6] Frank Dellaert and GTSAM Contributors. *borglab/gtsam*. Version 4.2a8. May 2022. DOI: 10.5281/zenodo.5794541. URL: https://github.com/borglab/gtsam.

[7] Frank Dellaert and Michael Kaess. *Factor Graphs for Robot Perception*. Foundations and Trends in Robotics, Vol. 6, 2017. URL: http://www.cs.cmu.edu/~kaess/pub/Dellaert17fnt.pdf.

[8] Giseop Kim, Sunwook Choi, and Ayoung Kim. *Scan Context++: Structural Place Recognition Robust to Rotation and Lateral Variations in Urban Environments*. 2021. arXiv: 2109.13494 [cs.RO]. URL: https://arxiv.org/abs/2109.13494.

[9] Jiadai Sun et al. « Efficient Spatial-Temporal Information Fusion for LiDAR-Based 3D Moving Object Segmentation ». In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022.

[10] Charles R Qi et al. « PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space ». In: *arXiv preprint arXiv:1706.02413* (2017).

[11] J. Behley et al. « SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences ». In: *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*. 2019.

## List of Figures