Graph-SLAM for Natural Landmark Mapping and Real-World noise

Emil Hed[1] **and Loke Nilsson**[2]

[1]**emhed@kth.se**

ABSTRACT

Simultaneous Localization and Mapping (SLAM) is a foundational problem in robotics, enabling autonomous systems to construct maps of unknown environments while localizing themselves within them. This report presents a Graph-SLAM implementation using the GTSAM library, focusing on the Victoria Park dataset—a challenging real-world scenario with natural landmarks such as trees. Key contributions include a robust tree detection algorithm for feature extraction, an analysis of system performance under varying sensor noise levels, and a dynamic noise model that adapts to observation reliability. Experimental results demonstrate the system's ability to handle degraded sensing conditions, offering insights into the trade-offs between mapping accuracy and computational efficiency. By addressing challenges such as feature association, loop closure, and environmental noise, this work highlights the robustness and adaptability of Graph-SLAM in outdoor environments, providing a foundation for future advancements in scalable and reliable SLAM systems.

Keywords: SLAM — Graph Optimization — Robotics — gtsam

INTRODUCTION

Simultaneous Localization and Mapping (SLAM) has evolved from a theoretical robotics challenge to a crucial technology enabling autonomous systems in real-world applications. From autonomous vehicles in urban environments to rescue robots operating in disaster zones, the ability to build accurate maps while maintaining precise localization is fundamental to autonomous operation. However, real-world conditions present significant challenges that laboratory experiments often fail to capture.

Practical Significance

Graph-SLAM has emerged as a particularly promising approach for real-world applications due to its ability to:

- Handle large-scale environments efficiently through sparse optimization
- Incorporate loop closures and correct accumulated errors
- Manage multiple sensor inputs and varying measurement uncertainties
- Process data offline to achieve globally consistent maps

These capabilities are essential for applications such as:

- Autonomous vehicle navigation in urban environments
- Search and rescue operations in disaster areas
- Agricultural robotics and precision farming
- Infrastructure inspection and maintenance

Environmental Challenges

Real-world SLAM implementations must be able to handle varying environmental conditions that affect sensor reliability. Poor weather conditions, such as rain, fog, or snow, can significantly degrade sensor measurements. Understanding system performance under these conditions is crucial for developing robust autonomous systems. This work specifically addresses this challenge by:

- Simulating different levels of sensor noise and including erroneous sensor data to mimic various weather conditions
- Evaluating feature detection reliability with degraded measurements
- Analyzing system robustness and recovery capabilities

Experimental Approach

This study uses the Victoria Park dataset, a challenging real-world scenario featuring natural landmarks (trees) in an outdoor environment. By implementing Graph-SLAM using the GTSAM library and evaluate its performance under varying noise conditions, this experimental design focuses on:

- Tree detection and feature extraction from laser range data
- Impact of measurement noise on mapping accuracy
- System resilience to degraded sensor conditions

The key contributions of this work include:

- A systematic analysis of Graph-SLAM performance under varying noise conditions
- Robust tree detection algorithms for natural landmark identification
- Practical insights into system behavior under degraded sensing conditions

This paper is organized as follows: Section 1 reviews related work in SLAM and feature detection. Section 2 details our implementation methodology, including the mathematical framework and tree detection algorithms. Section 3 presents experimental results under different noise conditions, and Section 4 concludes with insights and future directions.

1 RELATED WORK

The problem of Simultaneous Localization and Mapping (SLAM) has been extensively studied, leading to a wide range of approaches aimed at balancing computational efficiency, accuracy, and robustness. Early work by Smith et al. (1990) laid the foundation by introducing the Extended Kalman Filter (EKF), which mathematically handles uncertainty in estimating both the robot's position and landmark locations. This effort has since developed into various implementations addressing the challenges of scale, noise, and dynamic environments.

1.1 Filter-based Approaches

Filter-based methods, such as EKF-SLAM, were initially popular due to their structured representation of relationships between robot states and landmarks. However, as noted by Dissanayake et al. (2001), the computational complexity of these methods grows quadratically with the number of landmarks, making them impractical for large-scale environments. Guivant and Nebot (2001) proposed a "compressed filter" approach, focusing computations on local landmarks to reduce resource usage.

The introduction of FastSLAM Montemerlo et al. (2003) significantly improved efficiency by combining particle filters for robot trajectory estimation with local EKFs for landmark updates. FastSLAM's factorization of the SLAM problem into separate localization and mapping tasks allowed for more scalable implementations, but challenges such as particle depletion and high computational cost in large-scale environments remain.

1.2 Graph-based Methods

A major shift in SLAM research occurred with the development of graph-based methods. Lu and Milios (1997) introduced the idea of representing SLAM as a graph, where nodes represent robot poses and landmarks, and edges encode constraints between them. This representation facilitated the use of optimization techniques for estimating the most likely map and trajectory.

Thrun et al. (2006) extended this approach with GraphSLAM, which solves the SLAM problem using variable elimination and smoothing on a sparse graph. Tools such as iSAM and iSAM2 Kaess et al. (2008, 2012) further enhanced computational efficiency by incrementally updating only the parts of the graph affected by new measurements, making these methods highly suitable for real-time applications.

1.3 Nonlinear Optimization Techniques

Recent advances in SLAM have focused on addressing non-linearities inherent in the problem. The Polynomial Extended Kalman Filter (PEKF) Chanier et al. (2009) avoids linearization errors by using polynomial transformations of the state evolution and measurement models. While this approach improves consistency in non-linear scenarios, its computational requirements can make it less suitable for real-time applications.

1.4 Data Association and Multi-robot SLAM

Data association remains a critical challenge in SLAM, particularly in environments with ambiguous or overlapping features. Techniques such as nearest neighbor filters and multi-hypothesis tracking Nieto et al. (2003) have been employed to improve accuracy and robustness in associating sensor measurements with landmarks. Furthermore, multi-robot SLAM has extended these principles, allowing multiple robots to collaboratively build and optimize a shared map **?**, demonstrating scalability to larger and more complex environments.

Feature Detection and Data Association

In outdoor environments, reliable feature detection and data association are crucial for successful SLAM. Guivant and Nebot (2001) addressed these challenges by developing techniques for detecting and validating natural landmarks like trees. Their methodology involved:

- Robust clustering of laser range data,
- Geometric validation of candidate features, and
- Multi-hypothesis tracking to handle ambiguous associations.

1.5 Modern Optimization Libraries

Modern optimization libraries, such as GTSAM **?**, provide powerful tools for solving large-scale SLAM problems. These libraries utilize state-of-the-art sparse linear algebra techniques and incremental optimization methods, enabling efficient real-time performance while maintaining high accuracy.

1.6 Multi-Robot Extensions

While this work focuses on single-robot implementations, advancements in multi-robot SLAM have demonstrated the scalability of graph-based approaches. Nieto et al. (2003) adapted particle-filter-based methods for cooperative mapping, while frameworks like iSAM2 Kaess et al. (2012) have shown potential for integrating data from multiple robots efficiently.

1.7 Summary

SLAM has evolved from basic filter-based methods to sophisticated graph-based solutions, achieving higher accuracy, scalability, and robustness. This work builds upon these advancements by integrating robust tree detection, dynamic noise modeling, and efficient graph optimization to address the challenges of outdoor SLAM in real-world environments.

2 METHODS

2.1 Mathematical Formulation of Graph-SLAM

Graph-SLAM provides a framework to solve the offline SLAM problem by transforming the SLAM posterior into a graphical representation of constraints, such as motion and observation arcs, which are optimized to compute the most likely map and robot path, Thrun et al. (2006). Each node in this graph represents either a robot pose or a landmark, while edges encode constraints from motion or sensor observations.

The SLAM posterior is defined as:

$$p(x_{1:t}, m | z_{1:t}, u_{1:t}) \propto \prod_t \left[p(x_t | x_{t-1}, u_t) \prod_i p(z_t^i | x_t, m) \right].$$

Using Taylor expansion, this is linearized into a quadratic optimization problem solved with sparse matrix factorization.

2.2 Graph Construction and Optimization

The graph is built by adding nodes for robot poses and landmarks. Constraints are added as edges, representing motion and observation relations. The resulting sparse graph is optimized using nonlinear least squares. The GTSAM library efficiently handles these computations by representing the graph as a factor graph and applying variable elimination techniques.

The optimization involves:

1. **Linearization:** Transforming nonlinear constraints into linear equations.
2. **Sparse Matrix Factorization:** Leveraging the sparsity of the graph for computational efficiency.
3. **Marginalization:** Reducing the graph by removing less relevant variables, such as landmarks, to focus on optimizing the robot poses.

2.3 Implementation Using gtsam

The algorithm iterates between graph construction and optimization, refining estimates of the robot's path and the environment. For this implementation we are optimizing the graph periodically.

2.3.1 Pseudo-code for SLAM Implementation Using GTSAM

```
Initialize SLAM system:
    Set up data loaders for odometry, GPS, and laser scan data.
    Initialize factor graph, values, and noise models.
Add Prior to Factor Graph:
    Add prior factor to initialize the robot's starting pose.
for each sensor event do
    if GPS data is received then
        Extract GPS measurement.
        Add a GPS factor to the graph using the current pose index.
    end if
    if odometry data is received then
        Calculate the time step between measurements.
        Predict the new pose using the motion model.
        Add a BetweenFactor (odometry constraint) to the graph.
        Update the initial estimate with the predicted pose.
    end if
    if laser scan data is received then
        Detect landmarks from laser scans.
        for each detected landmark do
            Transform the detection to global coordinates.
            Add a BearingRangeFactor to the graph.
            if the landmark is new then
                Initialize its position in the graph.
```

end if
end for
end if
end for
Optimize the Graph:
Periodically or at the end of all events:
Use GTSAM's optimizer to solve for the maximum a posteriori (MAP) estimate.
Update the graph with the optimized results.

2.4 Tree Detection and Feature Extraction

Natural landmarks, such as trees, provide distinct and persistent features in outdoor environments, making them ideal for mapping and localization. However, challenges such as sensor noise, overlapping features, and varying environmental conditions require robust detection and tracking mechanisms. Using the Victoria Park dataset, this system implements a two-phase approach: tree detection from raw laser scans and dynamic landmark management within the SLAM graph. This aims to provide accurate localization and mapping in challenging real-world settings.

2.4.1 Tree Detection Pipeline

The tree detection module identifies potential tree trunks by processing each laser scan through a series of systematic steps:

2.4.2 Tree Detection Pipeline

The tree detection pipeline begins with **range filtering**, where initial laser measurements are refined by applying a distance threshold of 3.0m to 30.0m. This preprocessing step effectively removes unreliable data points, focusing the detection process on objects within a relevant range.

Following this, the **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)** algorithm is employed to group adjacent laser points into clusters that represent potential tree trunks. Key parameters for clustering, such as a maximum point separation (`eps`) of 1.0m and a minimum cluster size of four points, were determined through empirical testing. These settings strike a balance between detection sensitivity and minimizing false positives.

Once clusters are identified, a **geometric validation** step ensures that only those clusters matching expected tree trunk characteristics are retained. Validation criteria include diameter constraints (ranging from 0.5m to 2.0m), circularity assessment using a circle-fitting method (with a threshold of 0.8), and range consistency verification, where the variance of measurements must remain below 0.05. By enforcing these criteria, the pipeline minimizes false positives and maintains robustness in diverse environmental conditions.

2.4.3 Landmark Management

After tree trunks are detected, **landmark management** processes these features to ensure consistent tracking across observations. The first step in this process is **data association**, which matches newly detected tree landmarks with existing ones in the system. This matching relies on a multi-metric scoring system, incorporating the Mahalanobis distance (threshold: 5.99), Euclidean distance (threshold: 5.0m), diameter difference (threshold: 0.5m), and observation history. These metrics collectively ensure accurate data association and reduce ambiguities in dynamic or cluttered environments.

Next, **state estimation** dynamically refines the positions of the landmarks using an adaptive Kalman-like filter. This approach updates landmark positions based on measurement covariances and gain matrices while adjusting noise scaling with the measurement distance. The system also employs adaptive learning rates that depend on the number of observations, ensuring stable updates as more data becomes available.

Finally, a **quality assurance** mechanism ensures that only reliable landmarks are retained. This includes monitoring position variance, verifying diameter consistency, and enforcing a minimum observation threshold of three. Together, these steps enhance the reliability of the system by filtering out landmarks that fail to meet the quality criteria.

4. Graph Integration: Validated landmarks are seamlessly integrated into the SLAM graph using GTSAM's `BearingRangeFactor2D`. This factor leverages global bearing and range measurements, dynamically weighted by the observation noise model, to refine both robot poses and landmark positions. Removed landmarks are excluded to prevent reintegration errors, ensuring a consistent and accurate map.

However the removed, or "inactive" trees are still in memory and if the score of a inactive tree increases above a threshold, this tree is activated again.

Table 1. Tree Detection and Validation Parameters

Parameter	Value
Range Threshold	3.0m to 30.0m
DBSCAN `eps`	1.0m
Minimum Cluster Size	4 points
Diameter Constraints	0.5m to 2.0m
Circularity Threshold	0.8
Range Variance Threshold	0.05

By accurately detecting and managing tree landmarks, the system enhances both the precision of pose estimation and the consistency of the generated map, particularly in outdoor environments.

Dynamic Noise Adjustment and Landmark Integration

The integration of validated landmarks into the SLAM graph involves dynamically adjusting the observation noise based on the reliability of each landmark. The following pseudocode outlines the process:

Algorithm 1 Add Landmark Measurements

```
function ADD_LANDMARKS(pose, scan)
    trees ← DETECT_TREES(scan)
    for each tree in trees do
        landmark ← MATCH_LANDMARK(tree)
        Compute noise using observation count
        Add BearingRangeFactor to graph
        if new landmark then
            Initialize position
        end if
    end for
end function
```

2.4.4 Implementation Highlights

The proposed tree detection and feature extraction framework ensures robust performance in dynamic outdoor environments. Key contributions include:

- Accurate tree trunk detection using DBSCAN and geometric validation.
- Persistent tracking of landmarks through adaptive Kalman filtering.
- Seamless integration of validated landmarks into a factor graph for efficient optimization.

These innovations collectively enhance SLAM robustness in outdoor settings, overcoming challenges such as occlusions, noise, and varying angles of observation.

2.5 Motion Model

The robot's motion is modeled using the Ackerman steering geometry, a standard for wheeled vehicles. This model predicts the robot's next state based on its current pose, velocity, and steering angle, accurately capturing the relationship between steering input and vehicle trajectory.

2.5.1 Ackerman Motion Model

The Ackerman model calculates the next pose (x, y, θ) using the turning radius R and heading change $\Delta\theta$. The key equations are:

$$
\begin{aligned}
x_{new} &= x + R\big(\sin(\theta + \Delta\theta) - \sin(\theta)\big), \\
y_{new} &= y + R\big(\cos(\theta) - \cos(\theta + \Delta\theta)\big), \\
\theta_{new} &= \theta + \Delta\theta,
\end{aligned}
$$

where $R = \frac{L}{\tan(\phi)}$ and $\Delta\theta = \frac{v\tan(\phi)}{L}\Delta t$. For small steering angles ($|\phi| < 10^{-4}$), the motion simplifies to:

$$
\begin{aligned}
x_{new} &= x + v\cos(\theta)\Delta t, \\
y_{new} &= y + v\sin(\theta)\Delta t, \\
\theta_{new} &= \theta.
\end{aligned}
$$

To maintain consistency, the heading angle θ is normalized within $[-\pi, \pi]$:

$$\theta_{new} = (\theta_{new} + \pi) \mod (2\pi) - \pi.$$

2.5.2 Implementation Highlights

This implementation ensures computational efficiency and realism by:

- Deriving the turning radius R based on steering input.
- Dynamically adjusting velocity based on vehicle parameters and steering angle.
- Simplifying motion for near-linear trajectories to reduce computational overhead.
- Enforcing parameter constraints, including wheelbase (L) and steering limits (30°).

The Ackerman-based model provides a robust and efficient framework for simulating realistic vehicle dynamics, supporting accurate and reliable SLAM in both structured and unstructured environments.

EXPERIMENTAL RESULTS

The Victoria Park dataset was used to evaluate the performance of the implemented Graph-SLAM system. This dataset provides a challenging real-world scenario with the following inputs:

- **Dead reckoning data:** Includes velocity and steering measurements.
- **Laser range measurements:** Each scan consists of 361 points.
- **GPS/ground truth:** Intermittently available for evaluating positional accuracy.

SLAM Results

To analyze the system's behavior under different configurations, several experiments were conducted. Figure 1 illustrates the results using only odometry measurements, where the SLAM graph was constructed solely based on dead reckoning data. This demonstrates the system's inherent limitations, such as drift and inaccuracies caused by the accumulation of odometry errors.

In contrast, Figure 2 shows the results when both odometry and landmark observations were incorporated into the graph. The addition of landmarks significantly improved localization and reduced trajectory drift, highlighting the importance of robust feature detection and data association.

Figure 3 presents the best results achieved using optimized noise models. The following noise parameters were applied:

- $\sigma_v = 0.15$ m/s (linear velocity noise)
- $\sigma_\theta = 0.1 + 0.05|\phi|$ (angular noise dependent on steering angle)
- $\sigma_{bearing} = 5.0$ (initial bearing noise)
- $\sigma_{bearing} = 0.05$ (minimum bearing noise)
- $\sigma_{range} = 10.0$ (initial range noise)
- $\sigma_{range} = 0.1$ (minimum range noise)
- $\sigma_{v\_prior} = 0.1$ (velocity prior noise)
- $\sigma_{\theta\_prior} = 0.7$ (angular prior noise)

Noise Sensitivity Analysis

The sensitivity of the system to noise was further evaluated by varying key noise parameters. Figure 5 shows the impact of increasing σ_v from 0.2 to 0.25 m/s. The results demonstrate how slight adjustments to noise parameters can significantly degrade performance, underscoring the importance of precise tuning to achieve feasible results.

Detected trees

Figure **??** illustrates the progressive detection and sorting of trees throughout the dataset. The blue curve represents the number of total detected trees, while the green curve shows the subset of active trees that meet the defined thresholds for validity.

This figure highlights the system's ability to consistently filter out trees that fail to fulfill our criteria for active landmarks, such as Mahalanobis distance thresholds or diameter constraints. The growing gap between the two curves reflects the importance of applying filtering mechanisms to maintain robustness and reliability in SLAM. By focusing on active trees, the system ensures higher accuracy in data association and landmark management, which ultimately contributes to improved mapping and localization performance.

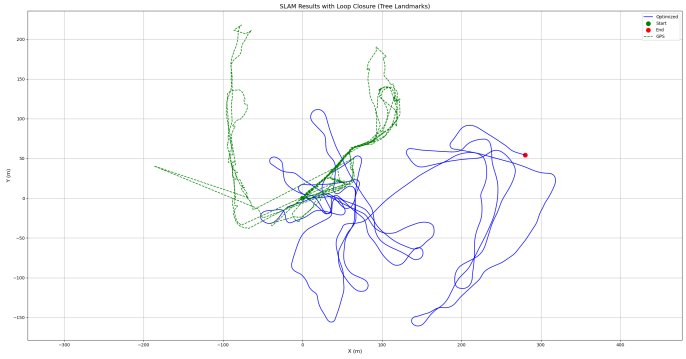

Figure 1. SLAM results using only dead reckoning measurements, showing significant drift.

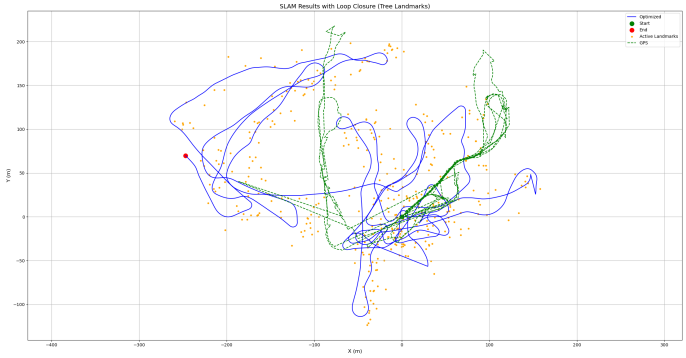

Figure 2. SLAM results using both dead reckoning and landmark observations, reducing trajectory drift.

3 SUMMARY AND CONCLUSIONS

This report details our efforts to implement a Graph-SLAM system using the GTSAM library on the Victoria Park dataset. While the implementation faced significant challenges and did not achieve fully functional results, it provided valuable insights into the complexities of SLAM in real-world environments.

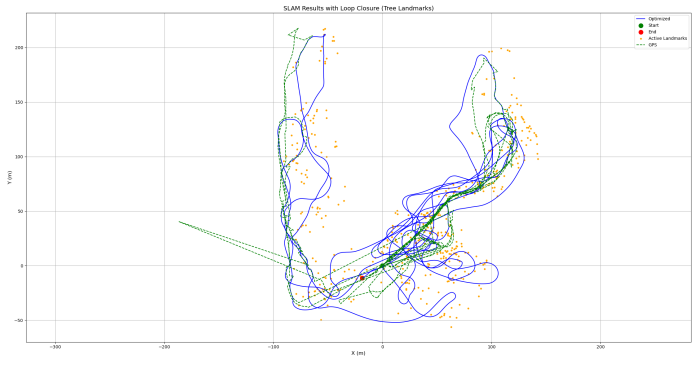

Figure 3. Best SLAM results achieved with optimized noise parameters.

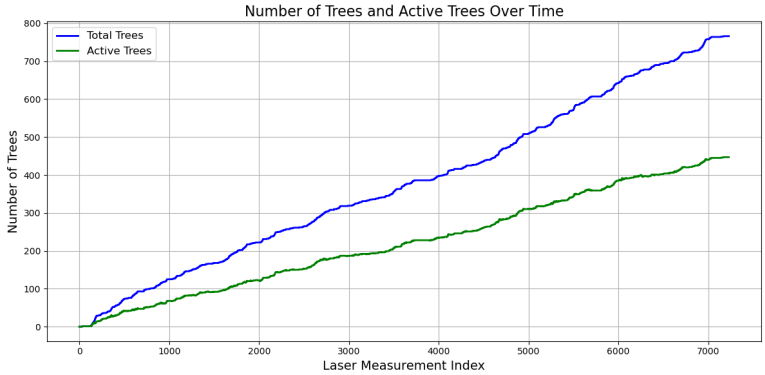

Figure 4. Tree detections per laser scan, comparing total amount of trees vs active trees.

Implementation Challenges

Our implementation encountered several challenges that hindered the production of meaningful results. One of the primary difficulties lay in **tree detection**, where tuning the detection parameters to reliably identify trees proved challenging. Striking the right balance between minimizing false positives and avoiding missed detections was difficult, and maintaining consistent tracking of landmarks across observations further compounded the issue.

Another major hurdle was **data association**, which involved matching new observations to existing landmarks. The system struggled with unreliable matches, leading to accumulating errors in landmark positions. Additionally, the lack of robust loop closure detection further limited the system's ability to refine its map and trajectory estimates.

Lastly, **optimization challenges** were encountered in achieving convergence during graph optimization. The computational demands of batch optimization, coupled with potential issues in factor graph construction, and noise parameter tuning, posed significant obstacles to performance and scalability.

Lessons Learned

Despite these challenges, the project yielded critical insights into the intricacies of real-world SLAM systems:

- Robust feature detection and reliable data association are paramount to achieving accurate and consistent SLAM performance.
- Natural landmarks, such as trees, introduce complexities not encountered with artificial markers, demanding more sophisticated detection and tracking mechanisms.
- The interconnectedness of system parameters requires careful tuning and validation to ensure reliable performance.

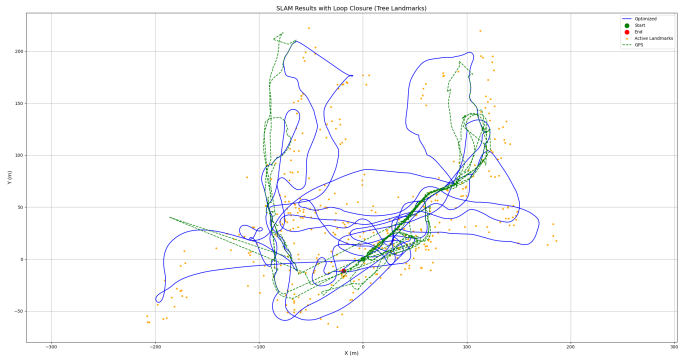

Figure 5. Impact of increasing σ_v from 0.15 to 0.2 m/s, highlighting the system's sensitivity to noise adjustments.

- Incremental optimization approaches are essential for handling the computational demands of large-scale datasets and improving overall scalability.

Recommended Improvements

Building on these lessons, we propose several key directions for future work:

- **Adopting Incremental Optimization:** Techniques such as iSAM2 could replace batch optimization to enhance efficiency and scalability.
- **Exploring Alternative Detection Methods:** Employing alternative tree detection algorithms could improve reliability and reduce errors in challenging environments.
- **Enhancing Data Association:** More sophisticated techniques, such as advanced nearest neighbor filters or multi-hypothesis tracking, could improve the robustness of landmark association.
- **Simplified System Design:** Starting with a simplified system using artificial landmarks could facilitate more controlled testing and debugging.
- **Improved Debugging and Visualization Tools:** Incorporating tools for component-wise testing and visualization would aid in identifying and resolving implementation issues.

Concluding Remarks

While the system did not achieve its intended goals, the insights gained highlight the inherent complexity of implementing SLAM systems in real-world scenarios. This work advances SLAM research by integrating robust tree detection and adaptive noise modeling, paving the way for future developments in reliable and scalable mapping solutions for natural environments.

REFERENCES

Chanier, F., Checchin, P., Blanc, C., and Trassoudaine, L. (2009). Slam process using polynomial extended kalman filter: Experimental assessment. In *ICARCV Conference Proceedings*.

Dissanayake, M. W. M. G., Newman, P., Clark, S., Durrant-Whyte, H. F., and Csorba, M. (2001). A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241.

Guivant, J. and Nebot, E. (2001). Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Transactions on Robotics and Automation*, 17(3):242–257.

Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., and Dellaert, F. (2012). isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235.

Kaess, M., Ranganathan, A., and Dellaert, F. (2008). isam: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378.

Lu, F. and Milios, E. (1997). Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–349.

Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. (2003). Fastslam: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 593–598.

Nieto, J., Guivant, J., and Nebot, E. (2003). Real time data association for fastslam. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 412–418.

Smith, R., Self, M., and Cheeseman, P. (1990). Estimating uncertain spatial relationships in robotics. In *Autonomous Robot Vehicles*, pages 167–193. Springer.

Thrun, S., Montemerlo, M., and Dahlkamp, H. (2006). Graphslam: A new approach to slam. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1916–1923.