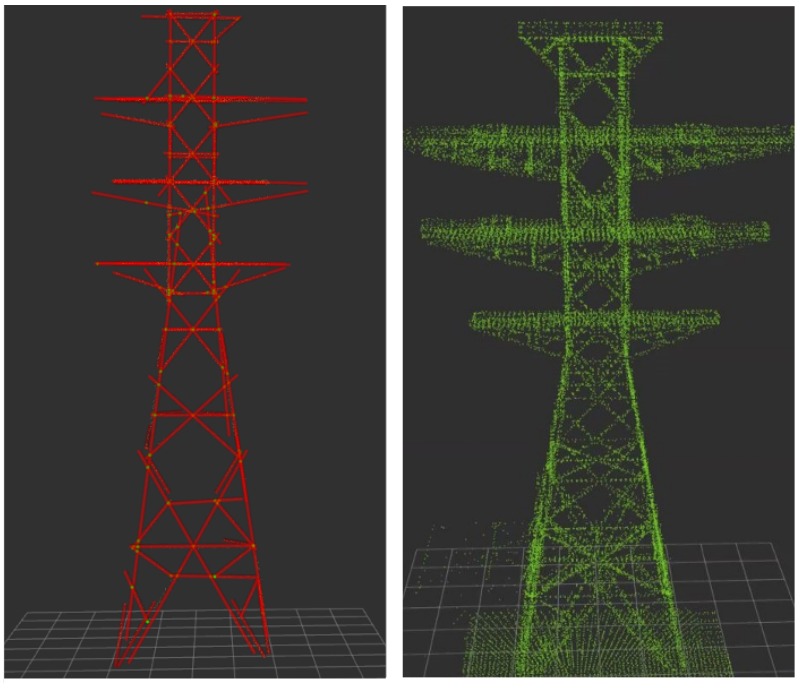
SS 2023-2024, SMT-ROB, DISAL-SP188

Start: 19.02.2024
Finish: 07.06.2024

Geometric Reconstruction of a Steel Structure from onboard Sensor Data

Robin Junod



Professor: Alcherio Martinoli
Assistant: Lucas WÃ¤lti

This page intentionally left blank.

Contents

Chapter 1 Introduction

This project presents an investigation into the geometric reconstruction of steel structures based on data acquired from onboard sensors. The objective is to develop a method for creating accurate 3D models of steel infrastructures using sensor data, which is inherently noisy and complex. This approach is motivated by the need for more efficient and error-resistant methods for structural inspection, moving beyond traditional manual techniques that are often time-consuming and costly.

The methodology involves the collection of spatial data through sensors mounted on drones and robots. This data is then processed to filter out noise and extract meaningful geometric information about the structure. For this purpose, the algorithm is based on the 3D hough transform. The project worked on Robot Operating System (ROS) and Webots for the simulation part.

1.1 Previous Work

The primary goal of this project was to perform segmentation on point clouds of steel structures. Various methods were tested to identify the most effective solution for real-time analysis of 3D measurements.

RANSAC, DBSCAN, and GMM were explored for their suitability in segmenting point clouds. RANSAC is powerful for identifying planes within point clouds but is limited to separating planes and is computationally intensive. DBSCAN effectively reduces noise and clusters large regions without requiring prior knowledge of the number of clusters, but it is time-consuming and ineffective for segmenting individual metal beams. GMM can cluster point clouds based on spatial distribution and works well in some scenarios, but it requires prior knowledge of the number of clusters and struggles with complex arrangements, making it also time-consuming for real-time processing.

On the other hand, the 3D Iterative Hough Transform [1] has demonstrated promising results. This method was robustness to noise and can accurately detect lines within 3D point clouds, even in complex arrangements. It is particularly well-suited for steel infrastructures, which are often composed of linear elements such as beams and columns. It provides accurate line parameters, making it highly effective for detailed segmentation tasks of steel structures. However, it requires careful tuning of parameters like the spread of points around lines (dx) and the minimum vote count ($minvotes$).

The 3D Hough Transform shows significant potential, compared to the other approaches, for real-time analysis of steel structures, providing accurate and reliable segmentation results.

1.1.1 3D Iterative Hough Transform

The second part of this project focused on developing a real-time segmentation pipeline for 3D point clouds of steel infrastructures using the 3D Iterative Hough Transform. This method was chosen for its effectiveness with steel structures.

The implementation was integrated into the ROS system and simulated using Webots. Most coding was done in C++ for optimal performance. The main pipeline is as follows:

Point Cloud Acquisition

- Point clouds were acquired from a Time of Flight (ToF) sensor. Each point cloud was synchronized with the drone's pose data.

Point Cloud Pre-processing

- Filtering thresholds were applied using the Point Cloud Library (PCL) to exclude distant points. A voxel grid filter ensured uniform density and reduced the number of points for computation.

Iterative 3D Hough Transform

- The algorithm identifies lines in 3D point clouds by converting points into a parameter space. Iteratively, the most prominent line is detected, fitted, and removed from consideration [1].

Segments Sorting

- Principal Component Analysis (PCA) was used to evaluate segments, retaining those that met criteria like point count, radius, and density.

Segments Fusion and Intersection

- New segments were integrated with existing ones based on similarity in position and parameters. Fusion involved averaging segments based on PCA coefficients and point counts.

- Intersections between segments were identified by finding the shortest distance between them, calculating intersection points if they were close enough.

In conclusion, the 3D Iterative Hough Transform for real-time segmentation of steel infrastructure point clouds proved effective, with high accuracy in matching extracted

segments to ground truth data. While sensitive to noise and requiring careful parameter tuning, the project showed significant potential for enhancing autonomous inspection capabilities in steel infrastructure.

1.2 Current problematic

One of the key challenges identified in the implementation of the 3D Iterative Hough Transform for real-time segmentation of steel infrastructure point clouds is its sensitivity to noise, particularly noise in the drone's position data. Accurate segmentation relies heavily on precise point cloud data, which can be significantly affected by any inaccuracies in the drone's positioning. This noise can lead to errors in detecting and mapping structural elements, thus reducing the overall efficiency and reliability of the segmentation process.

My work will focus on addressing this issue, with a particular focus on improving Visual Odometry (VO) using the existing Time of Flight (ToF) sensors on the Starling drone to enhance the accuracy and robustness of the segmentation pipeline.

Chapter 2 Literature review

2.1 Visual-Informed Localization

Position estimation is a critical component in the navigation and control of drones. Traditional methods rely heavily on GPS, but the limitations of GPS in terms of accuracy and availability, especially in indoor or urban environments, have led to the exploration of alternative methods. Visual information, obtained through cameras and LiDAR, has emerged as a promising solution for enhancing the accuracy and robustness of position estimation.

2.1.1 Visual odometry

Visual Odometry (VO) involves the estimation of a drone's trajectory by analyzing sequential images. The core idea is to track features across frames and estimate the motion of the camera between frames. There are multiple ways of performing VO; I will present two of the main approaches: feature-based methods and deep learning methods. Both approaches aim to use visual features to determine the drone's position.

Feature-Based VO

Feature-based VO methods detect and match visual features between consecutive frames to estimate camera motion. ORB-SLAM [2], a state of the art system in this domain, was introduced in 2015 with updates in 2016 (ORB2-SLAM) and 2020 (ORB-SLAM3). It uses ORB (Oriented FAST and Rotated BRIEF [3]) features for real-time monocular SLAM. ORB features quickly identify corners based on intensity differences and are robust against scale and rotation changes. These features are extracted and tracked across frames to determine the drone's position and motion.

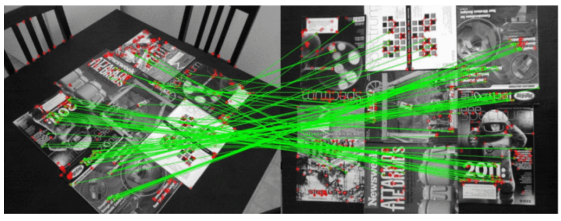


Figure 2.1: Matching result using ORB features [3]

ORB-SLAM Workflow

- **Initialization:** The system initializes by creating an initial map from the first few frames, detecting ORB features, matching them, and estimating the initial camera pose.
- **Tracking:** For each new frame, ORB features are extracted and matched with previous frames or the local map. The camera pose is estimated using methods like the Perspective-n-Point (PnP) algorithm and RANSAC for robustness.
- **Local Mapping:** Keyframes with significant changes are selected periodically. New map points are added via triangulation from multiple keyframes, followed by local bundle adjustment to optimize poses and map points.
- **Loop Closing:** The system checks for loop closures when revisiting mapped areas. Pose graph optimization corrects drift and improves map consistency by adjusting camera poses and map points to align with loop closure constraints.

ORB-SLAM achieves real-time performance and high accuracy in challenging environments through efficient feature detection, robust tracking, and optimization techniques.

Deep Learning-Based VO

The development of deep learning has made VO methods highly efficient and state-of-the-art. Convolutional neural networks (CNNs) excel at extracting robust features from images, surpassing traditional handcrafted methods and improving accuracy and reliability in diverse environments. Deep learning enables end-to-end VO systems, where the entire pipeline, from feature extraction to pose estimation, is learned directly from data.

A key example is "DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks" by Sijie Wang and Yizhou Wang [4], which combines CNNs for spatial feature extraction and recurrent neural networks (RNNs) for modeling temporal dependencies. This approach captures both spatial and temporal aspects of visual odometry, enhancing performance over traditional methods. Despite requiring labeled datasets for training, deep learning-based VO represents the cutting edge in visual odometry technology.

2.1.2 LiDAR-Based odometry

LiDAR-based odometry involves the use of LiDAR sensors to measure distances to surrounding objects, creating a point cloud that represents the 3D environment. The primary objective is to estimate the change in position and orientation of the LiDAR sensor over time, which can be achieved through various methods, including scan matching, feature-based approaches, and machine learning techniques. We are going to focus more on the scan matching and feature-based approaches.

Scan matching : ICP

Scan matching techniques align consecutive point clouds to estimate motion. In this field, the Iterative Closest Point (ICP) [5] [6] is a widely used method. This approach minimize the difference between two point clouds by iteratively refining the alignment.

The key concept of ICP can be summarized in two steps:

1. compute correspondences points between the two point clouds.
2. compute the transformation which minimizes distance between the corresponding points.

Since ICP computes the transformation necessary to align two point clouds, it can effectively deduce changes in sensor position. This capability makes it particularly valuable for odometry, as it allows for the tracking of sensor movement over time.

ICP Algorithm:
Given a reference point cloud P and a second point cloud Q:

1. Compute corresponding pairs: For each point $q_j \in Q$, find the nearest point $p_i \in P$. This forms pairs of corresponding points (p_i, q_j).
2. Update Transformation: Use Singular Value Decomposition (SVD) to find the optimal rotation R and translation T that minimize the error. This involves the following steps:
   (a) Compute the centroids of the corresponding points:

$$\bar{p} = \frac{1}{N}\sum_{i=1}^{N} p_i, \quad \bar{q} = \frac{1}{N}\sum_{j=1}^{N} q_j \tag{2.1}$$

   (b) Center the points by subtracting the centroids:

$$P' = P - \bar{p}, \quad Q' = Q - \bar{q} \tag{2.2}$$

   (c) Compute the covariance matrix H:

$$H = \sum_{j=1}^{N} (P'_i)(Q'_j)^T \tag{2.3}$$

   (d) Perform SVD on H:

$$H = U\Sigma V^T \tag{2.4}$$

   (e) Compute the rotation R and translation T:

$$R = VU^T, \quad T = \bar{p} - R\bar{q} \tag{2.5}$$

3. Transform the Data Point Set: Update the data point set Q:

$$Q \leftarrow RQ + T \tag{2.6}$$

4. Compute the Error Objective Function: The error objective function error is the sum of the Euclidean distances between all corresponding point pairs after the transformations. If this error is below a certain threshold, the algorithm stops.

5. Iterate: Repeat steps 1-4 until convergence, i.e., until the change in error is below a predefined threshold.

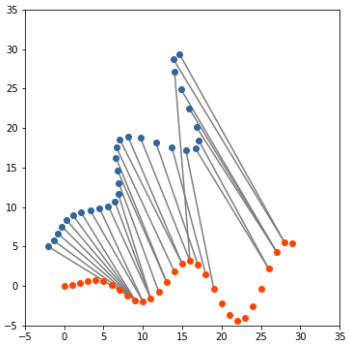


Figure 2.2: Extracting pairs between point clouds (1) [7]

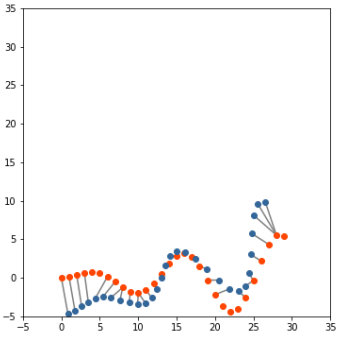


Figure 2.3: Applying the translation (2.b) [7]

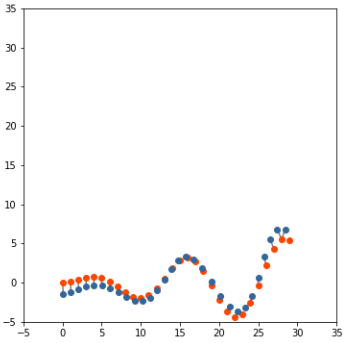


Figure 2.4: Applying the rotation (2.e) [7]

Figure 2.5: Steps of the ICP to repeat until convergence.

ICP limitations and GICP

ICP is powerful for aligning point clouds but has limitations, such as sensitivity to the initial alignment and local minima. Variants of ICP, such as Generalized Iterative Closest Point (GICP) [8], has been developed to improve performance and robustness.

Unlike ICP, which only minimizes the point-to-point distance, GICP minimizes a combination of point-to-point and plane-to-plane distances. Each point in the point cloud is associated with a covariance matrix that captures the local surface geometry. The algorithm then minimizes a combined error metric that incorporates these covariance matrices, addressing both point-to-point and plane-to-plane distances. The computation is more complex than the ICP algorithm, and the transformation is not found directly through SVD decomposition but rather through the minimization of a loss function that considers the geometric and statistical properties of the points.

This approach allows for more accurate and robust alignments even when the initial guess is not perfect. These advancements make GICP a preferred choice in scenarios where ICP's performance is insufficient.

Feature-Based Approaches : LOAM

Feature-based approaches extract distinct features from LiDAR scans, such as edges, planes, or corners, and track these features over time. These methods reduce computational complexity and improve robustness in dynamic environments.

One of the most influential and state-of-the-art papers in LiDAR-based odometry is "LOAM: Lidar Odometry and Mapping in Real-time" by Ji Zhang and Sanjiv Singh, presented at the Robotics: Science and Systems Conference in 2014. This paper introduces a highly efficient and accurate method for simultaneous localization and mapping (SLAM) using LiDAR data.

The odometry process in LOAM involves extracting edge and planar feature points from lidar scans and matching these features between consecutive scans. Unlike traditional ICP, which matches all points, LOAM focuses on significant features and employs robust selection criteria to enhance accuracy. This approach reduces the influence of less informative points and improves robustness.

The optimization process is also different; LOAM uses the Levenberg-Marquardt method for nonlinear optimization of the pose transform, whereas ICP often relies on simple least-squares optimization.

All of these features make LOAM a very robust and fast algorithm for lidar odometry.

Chapter 3 Noise Resilience

In real-life applications, achieving precise and reliable absolute positioning of drones is essential for various tasks such as navigation, mapping, and surveillance. However, the inherent limitations and imperfections in positioning systems can introduce challenges that need to be addressed for optimal drone performance.

Even when GPS signals are available, they may not always provide the level of precision required for certain applications. This lack of precision can be particularly problematic our segmentation applications.

To mitigate the limitations of GPS-based positioning, drones often incorporate additional sensors, such as Inertial Measurement Units (IMUs), to track their motion and orientation. IMUs provide continuous measurements of acceleration and angular velocity. However, while IMUs excel at tracking changes in orientation and speed, they are prone to a phenomenon known as drift. Drift refers to the gradual accumulation of error in IMU measurements over time, resulting from sensor noise and integration errors.

In this chapter, we explore strategies for improving the noise resilience of drone positioning systems, focusing specifically on the Starling drone equipped with Time of Flight (ToF) sensors. We will use the point cloud data provided by the ToF sensors to enhance the drone's positioning accuracy. This involves integrating visual odometry and ToF data to mitigate the effects of position noise and achieve robust absolute positioning. We aim to improve the accuracy of the drone's real-time 3D point cloud segmentation capabilities.

3.1 Experimental setup

Simulation We are using Webots to simulate the drone environment. The steel structure in our project is an electric pole designed for long-distance electrical networks. Below is a view of the environment we are working with:

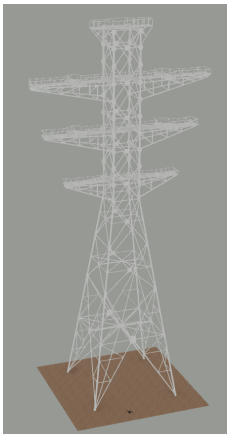


Figure 3.1: Electric pole inside Webots simulation

Drone The drone used in this simulation is based on the Starling drone. This drone is equipped with a single camera that captures images at 10 frames per second (fps) and a Time-of-Flight (ToF) sensor operating at 5 fps. For tracking, the drone is equipped with an Inertial Measurement Unit (IMU) that provides data on orientation and acceleration. Additionally, the drone's position is provided by the Webots supervisor, ensuring accurate navigation within the simulation.

Noise To simulate the drift in the drone's positioning data, we use random walk noise. Random walk noise is a way to model the cumulative nature of drift over time, where small random movements accumulate to create a larger deviation from the true position. Here below is the pseudo code of the random noise used.

Algorithm 1 Random Walk Noise

procedure RANDOMWALKNOISE(*position*, *noise_level*)
    Initialize a cumulative noise vector *cnv* with zeros
    Generate 3D noise vector from Normal distribution with std :*noise_level*
    Update *cnv* by adding *nv* ($cnv \leftarrow cnv + nv$)
    Add *cnv* to *real_position* to get *noisy_position*
    return *noisy_position*
end procedure

One important aspect to consider is that the noise level represents the standard deviation of the normal distribution used to generate the random vector. We used a noise level of 0.002. Since noise is added at each observation, the observation frequency significantly impacts the total deviation from the ground truth. The noise level chosen for the simulation is higher than what would be expected in a real-life scenario. For instance, after 3 minutes of flight, the drift would be approximately 1 meter.

The state estimate of the drone can also be prone to yaw drift. To address this issue and compare the performance of different methods, an additional random walk noise component in yaw has been introduced.

3.2 Odometry with Segmentation

One direct and conservative approach to reducing position noise in the previous implementation was to use the segmentation results from the 3D Hough Transform. This method adjusts the drone's position based on the difference between the segments detected by the drone at time t and the complete structure's segments at the same time. It is important to note that the drone receives point cloud data from the ToF sensor in its own frame. By aligning this point cloud data with the frame of the complete structure, we can identify segmentation errors between parallel segments and use this information to adjust the drone's position, thereby minimizing drift. This method is somewhat of a feature-based odometry approach, where certain features are extracted to aid odometry. In our case, we extract segments as features, using the 3D Hough transform.

3.2.1 Method

As a reminder, the primary objective is to minimize the noise resulting from drift. In this Webots simulation, we model the noise as a random walk affecting the position of the drone. As we are simulating it with webots we also know the real position of the drone which will help us compare the efficiany of our algorithm. The same random seed has been used throughout all of the experiments to keep consistency for comparison.

Let's now dive into the noise reduction algorithm based on the segmentation.

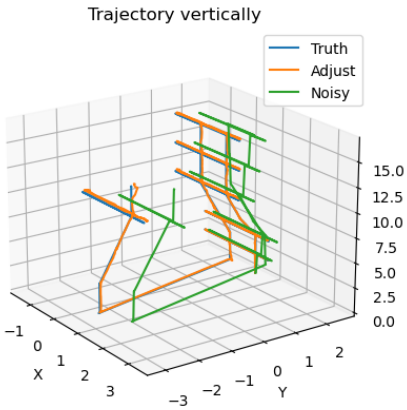
- First, all segments captured by the drone are projected onto the frame of the entire structure.
- Parallel segments are identified, and if the distance between these segments falls within a predefined threshold, they are considered to be identical. The distance between these segments is attributed to positioning errors due to drift.
- The shifting error is computed for all detected segments.
- An average of these distances is then calculated to adjust the drone's position.
- This process is iterative and will converge for errors that are within acceptable limits.

3.2.2 Results

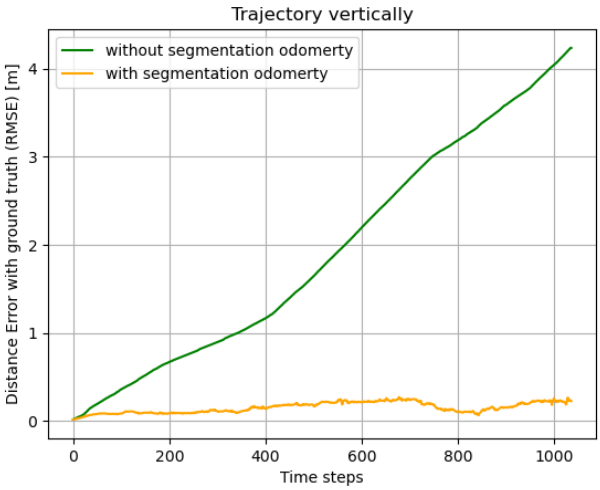
These results compare the performance of the segmentation odometry method. The labels 'Truth', 'Adjust', and 'Noisy' correspond respectively to the real position of the

drone, the position corrected by the segmentation odometry, and the initial noisy position. The other plots show the distance error relative to the real position of the drone, both with and without the odometry adjustment.

Position Error Improvement

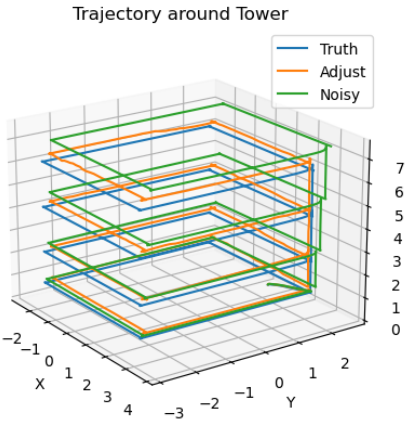


(a) Trajectory

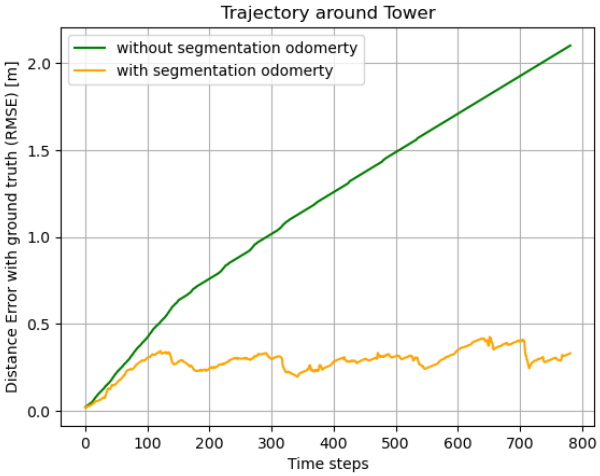


(b) Position error with/without segmentation odometry

Figure 3.2: Experiment with a vertical trajectory

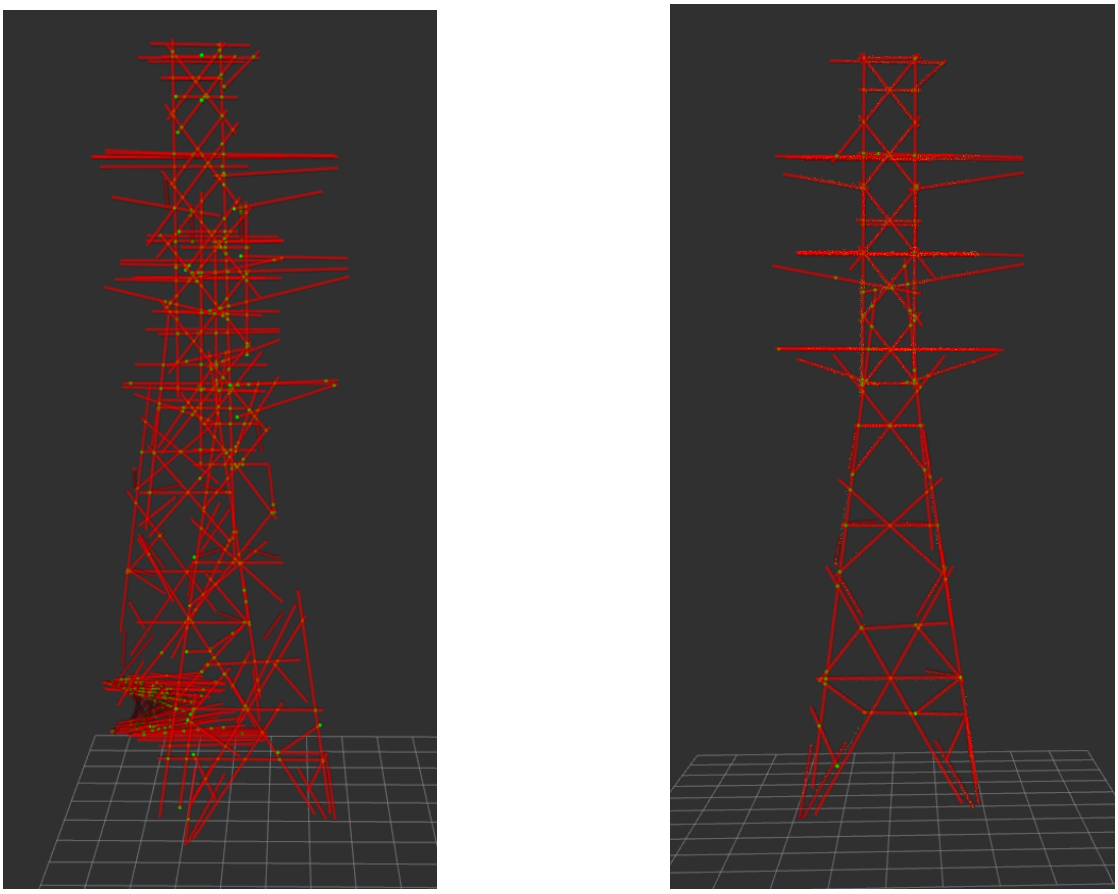


(a) Trajectory



(b) Position error with/without segmentation odometry

Figure 3.3: Experiment with a trajectory going around the tower



(a) Without segmentation odometry

(b) With segmentation odometry

Figure 3.4: Comparison of the segmentation with/without odometry correction

3.2.3 Discussion

The results shows that this algorithm based on segmentation effectively reduces noise and allows for good reconstruction of the structure, as shown in Figure 3.2.

The vertical trajectory yields better results compared to the trajectory around the structure. This difference is due to the drone temporarily losing sight of the structure when flying around it. To manage this issue, a potential solution would be to select a flight path that maintains a consistent view of the structure or to implement a smart path planning algorithm, which could be an area for future development. One of the strongest points of this algorithm is its simplicity, which allows it to readjust the drone's position at high frequencies.

3.3 Odometry with point clouds

The ICP algorithm determine the transformation between consecutive point clouds. This transformation serves two primary purposes. First, it allows us to iteratively build a more comprehensive point cloud by adding new points after each iteration, increasing the resolution and completeness of the map. Second, we use this transformation to minimize drone drift and reduce noise in the positioning of the drone. Reducing noise is essential when dealing with point clouds, as it directly affects the accuracy of the generated point cloud.

3.3.1 Method

The drone we are using is equipped with a ToF (time of flight) sensor, which provides us with a point cloud of the nearby structure. For odometry with point clouds, we used the Iterative Closest Point (ICP) algorithm (see sec 2.1.2). This algorithm determines the transformation needed to align two point clouds and is highly effective when the point clouds have similar structures and good initialization. However, ICP can suffer from inaccuracy and is sensitive to noise. To address these issues, a more robust algorithm called Generalized ICP (GICP) [8] was tested. Additionally, a variant that only applies translation, the 3 DoF ICP, was also tested. In total, three different versions of ICP were tested to compare their performance.

- The standard ICP
- The 3 DoF (degree of freedom) ICP
- The Generalized ICP

The odometry works as follows. Each times a new point cloud (PC1) is available from the ToF sensor, the algorithm compute the transformation between the new point cloud and the global point cloud (PC0). Here below is the main loop of the method.

1. Get a new point cloud in the frame of the drone (PC1)
2. Preprocessing of PC1 (removing NaN, pass-through filtering)
3. Align PC1 with global point cloud (PC0) using one of the ICP method
4. Adding PC1 to PC0, $PC0 \leftarrow PC0 + PC1$
5. Filtering of PC0 (Voxelgrid downsampling)

Noises We tested two different types of noise. The first type induces a constant drift in a random direction. The second type uses a random walk. It is important to note that the random walk noise level was higher than the constant drift to ensure that we could observe a noticeable difference.

3.3.2 Results

Random walk noise

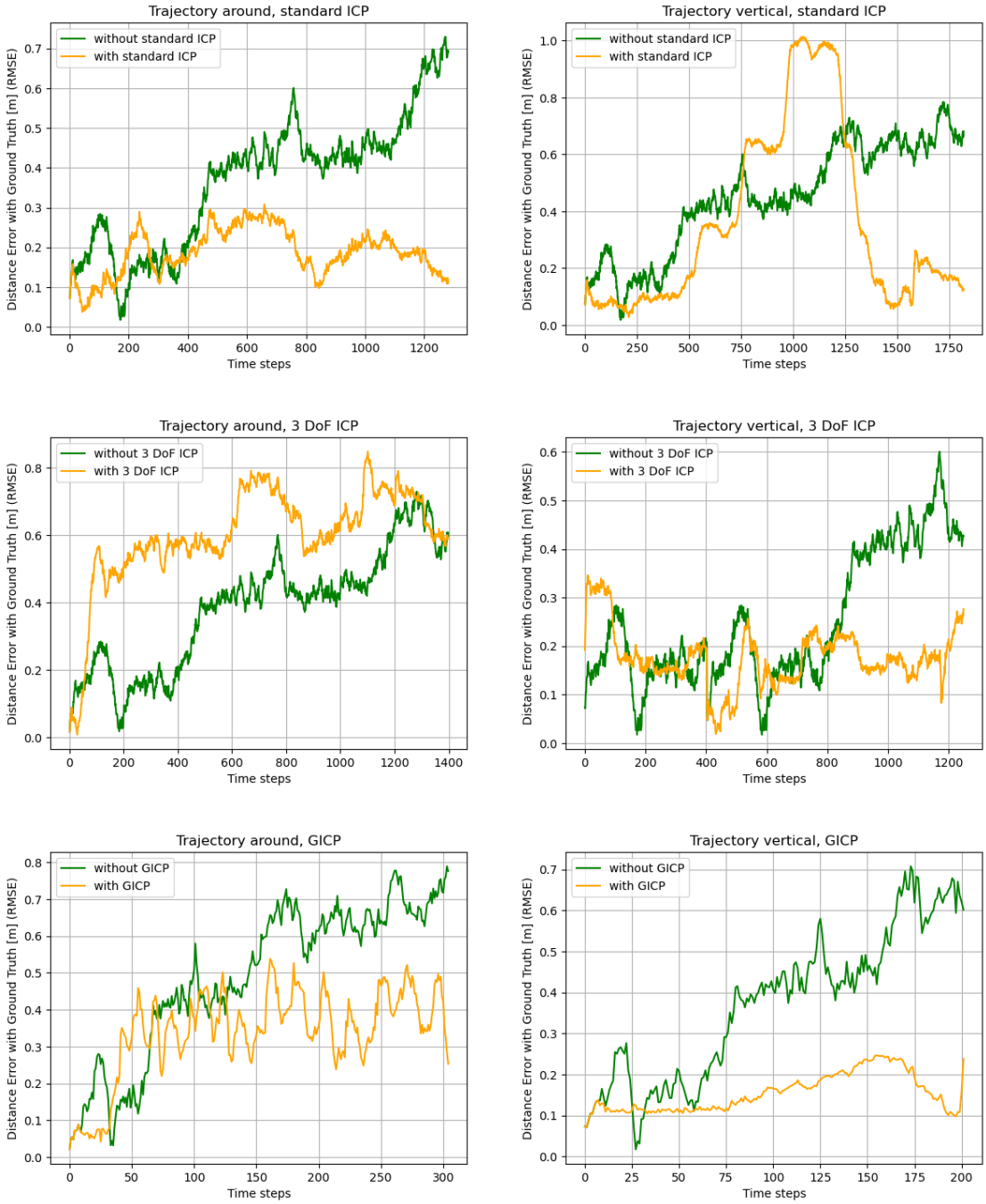


Figure 3.5: Comparison of ICP techniques under random walk (noise level = 0.002)

Constant drift

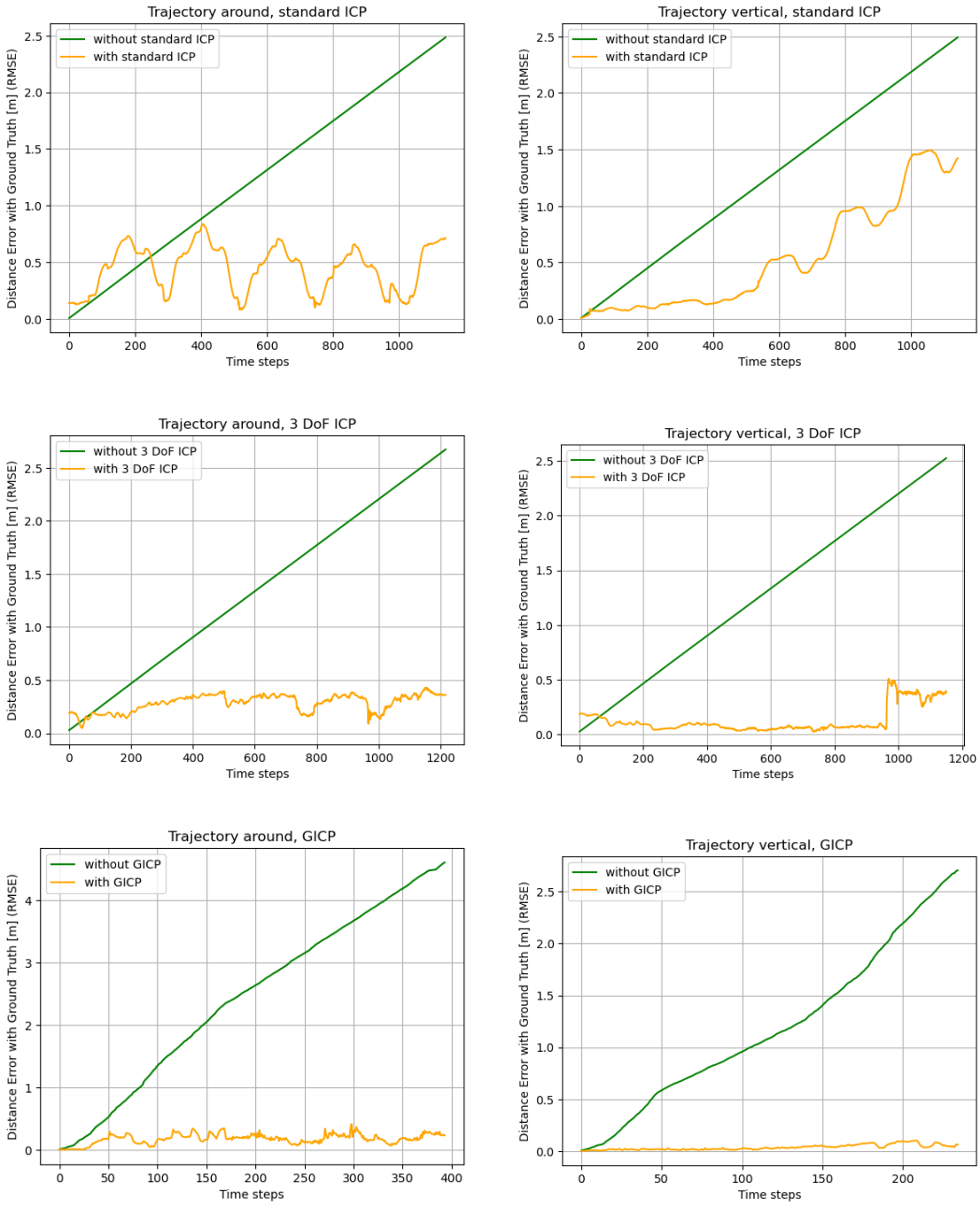


Figure 3.6: Comparison of ICP techniques under constant drift (noise level = 0.0001)

Point clouds comparison

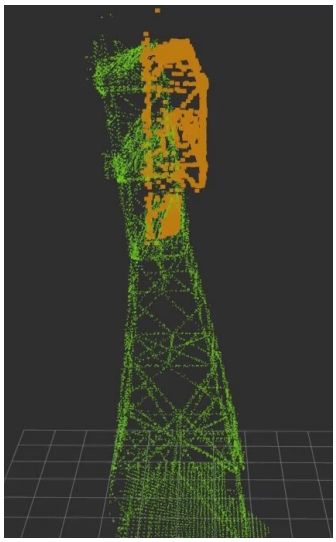


Figure 3.7: Standard ICP. Orange = real position of the point cloud. Green = point cloud estimated

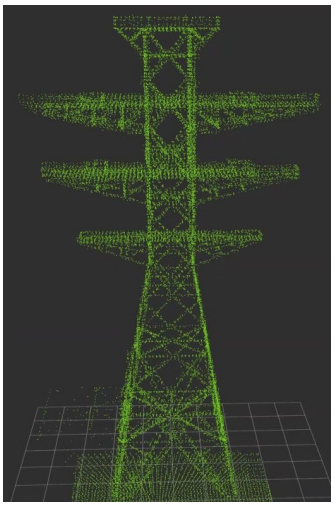


Figure 3.8: 3 DoF ICP. Solved the structure deformation issue of the standard ICP.

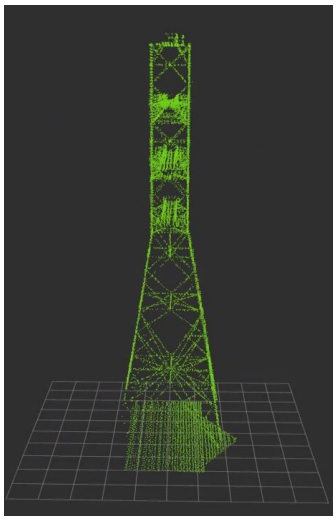


Figure 3.9: GICP, solves the issues encountered with standard ICP

Convergence issue

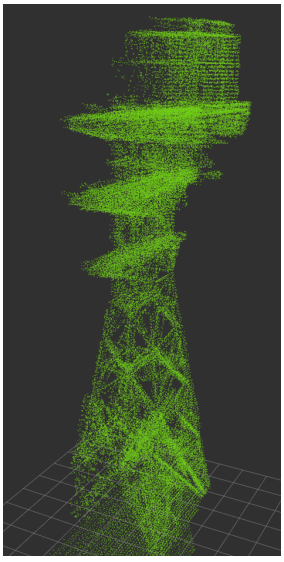


Figure 3.10: Example of incorrect convergence with 3 DoF ICP

3.3.3 Discussion

The standard ICP often fails to accurately determine the drone's position. This issue arises because standard ICP searches for a transformation with 6 degrees of freedom, including orientation correction. As the point cloud size increases during exploration, even a slight initial error in orientation correction leads to poor convergence and an inability to retrieve the correct position.

This occurs because standard ICP performs transformations in 6 DoF, while the noise affects only 3 DoF (translation only). Consequently, ICP mistakenly corrects non-existent orientation errors. As the drone explores and captures only small portions of the environment at each step, these incorrect 'orientation corrections' accumulate over time, resulting in a significant overall error. This is illustrated at the figure 3.7 where the overall electric pole is bent due to these orientation degree of freedom.

To address the issue of orientation correction, we tested a 3 DoF ICP approach. After some tuning, the 3 DoF ICP performed better than the standard ICP, but the algorithm still experienced incorrect convergence (fig. 3.10). This occurs when ICP converges to an incorrect local minima.

The final step was to test a more robust algorithm, the Generalized ICP (GICP). As shown in the results, GICP performed excellently in reducing noise in the drone's position, particularly in scenarios with constant drifting. Additionally, GICP completely resolved the structure deformation issues encountered with standard ICP. The main drawback of GICP is that it is nearly three times more time-consuming than standard ICP or 3 DoF ICP. Consequently, it may theoretically fail when the drone is moving too quickly.

Chapter 4 Conclusion

In this project, we aimed to develop a robust method for geometric reconstruction of steel structures using data from onboard sensors. The primary objective was to create accurate 3D models of steel infrastructures. For this task, the Iterative Hough Transform showed significant potential due to its robustness to noise and effectiveness in detecting lines within complex 3D point clouds.

A major challenge identified was the sensitivity of the 3D Iterative Hough Transform to noise, particularly in the drone positioning. This noise can lead to errors in detecting and mapping structural elements, reducing the overall efficiency and reliability of the segmentation process. To address this, we focused on improving Visual Odometry (VO) using existing Time of Flight (ToF) sensors on the Starling drone. The project was implemented using the Robot Operating System (ROS) and simulated on Webots.

The first experiment was to reduce this positional noise using the segmentation from the 3D Hough transform. For this we used an algorithm capable of detecting parallel segment, compute the distance error between these segments and readjust the drone position based on these. This simple algorithm showed good results but could suffer from inaccuracy when the drone lose sight on the structure.

The second part of this project was to experiments odometry using the Iterative Closest Point (ICP) algorithm and variants like 3 DoF ICP, and Generalized ICP (GICP) to align point clouds and reduce noise in drone positioning. The standard ICP often failed due to incorrect orientation corrections, which compounded over time, leading to significant errors. The 3 DoF ICP performed better but still suffered from incorrect convergence. The GICP, while computationally more intensive, provided excellent noise reduction and resolved the structure deformation issues seen with standard ICP. It also appears to be more robust against yaw noises (fig. 5.2).

In summary, our findings demonstrate the potential of the 3D Iterative Hough Transform and GICP for real-time segmentation and geometric reconstruction of steel structures. Despite the challenges related to noise and computational demands, these methods significantly enhance the accuracy and reliability of structural inspections using drones. Future work could focus on path planning in order to avoid the drone losing sight on the structure. This would enhance the drone odometry and the overall segmentation.

Bibliography

[1] C. Dalitz, T. Schramke, and M. Jeltsch, "Iterative Hough Transform for Line Detection in 3D Point Clouds," *Image Processing On Line*, vol. 7, pp. 184–196, 2017. https://doi.org/10.5201/ipol.2017.208.

[2] R. Mur-Artal, J. M. Montiel, and J. D. Tardos, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[3] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, pp. 2564–2571, 2011.

[4] S. Wang and Y. Wang, "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," *arXiv preprint arXiv:1709.08429*, 2017.

[5] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 5, pp. 698–700, 1987.

[6] P. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[7] C. Stachniss, "Iterative closest point (icp) - 5 minutes with cyrill." YouTube, 2020. Images by Igor Bogoslavskyi.

[8] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp.," in *Robotics: science and systems*, vol. 2, p. 435, Seattle, WA, 2009.

[9] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal of Computer Vision*, vol. 81, pp. 155–166, February 2009.

[10] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[11] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pp. 145–152, 2001.

[12] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time.," in *Robotics: Science and systems*, vol. 2, pp. 1–9, Berkeley, CA, 2014.

[13] O. Michel, "Webots: Professional mobile robot simulation," *Journal of Advanced Robotics Systems*, vol. 1, no. 1, pp. 39–42, 2004.

[14] Webots, "http://www.cyberbotics.com." Open-source Mobile Robot Simulation Software.

[15] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system." http://www.ros.org, 2009. Accessed: 2024-05-27.

[16] N. Bolourian and A. Hammad, "Lidar-equipped uav path planning considering potential locations of defects for bridge inspection," *Automation in Construction*, vol. 117, p. 103250, 2020.

[17] R. Almadhoun, T. Taha, J. Dias, L. Seneviratne, and Y. Zweiri, "Coverage path planning for complex structures inspection using unmanned aerial vehicle (uav)," in *Intelligent Robotics and Applications* (H. Yu, J. Liu, L. Liu, Z. Ju, Y. Liu, and D. Zhou, eds.), (Cham), pp. 243–266, Springer International Publishing, 2019.

[18] A. Jonnarth, J. Zhao, and M. Felsberg, "Learning coverage paths in unknown environments with reinforcement learning," 2024.

Chapter 5 Annex

5.1 Installation

The setup works as follows:

- Create catkin workspace to work with ROS
- "src" folder must contains the repo :
  - auto_pilot
  - pointcloud_segmentation
- Launch webots and

To launch the segmentation part using odometry correction:

- Open Webots the simulation world 'flying_arena_ros_tower.wbt' and starts it.
- Launch segmentation in a terminal with 'roslaunch pointcloud_segmentation all.launch'
- Lauch the pointcloud ICP with 'roslaunch pointcloud_segmentation pc_align.launch'

Quick Notes

1. **Webots Simulation**:
   - Ensure the Webots simulation is open and running before executing any `roslaunch` commands.
2. **ICP Method Selection**:
   - To switch between different ICP methods (ICP, GICP, DOF3), edit the `pointcloud_alignment_node.cpp` file and set the desired method.
3. **Noise Management**:
   - Noise handling is configured in the `drone_controller_node.py` file, located in the `auto_pilot` folder.

4. **Path Planning**:

    - An initial implementation for path planning has been coded in Python. It moves the drone using a velocity vector instead of waypoints, which should facilitate further development of the path planning algorithm.

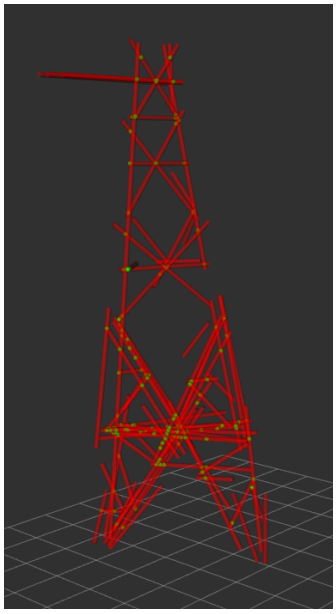


Figure 5.1: Issues of the segmentation method for YAW drift

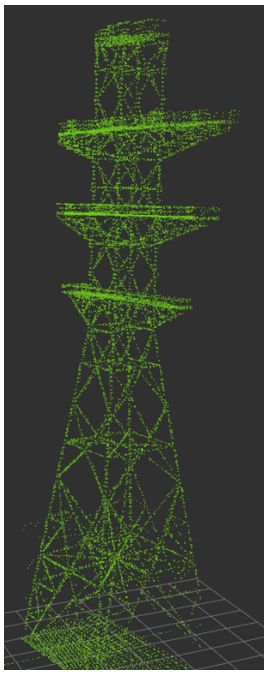


Figure 5.2: Robustness of GICP with yaw drift, view 1

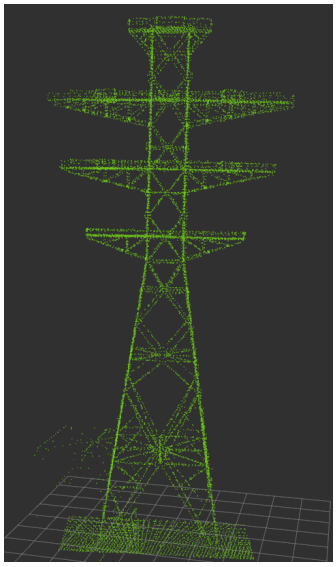


Figure 5.3: Robustness of GICP with yaw drift, view 2