2024

DEVELOPING ROBUST AUTONOMOUS VEHICLES WITH ROS

Dylan J. Kangas
Michigan Technological University, dylank@mtu.edu

DEVELOPING ROBUST AUTONOMOUS VEHICLES WITH ROS

By

Dylan J. Kangas

A THESIS

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In Electrical and Computer Engineering

MICHIGAN TECHNOLOGICAL UNIVERSITY

2024

This thesis has been approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE in Electrical and Computer Engineering.

Department of Electrical and Computer Engineering

Thesis Co-advisor:	*Dr. Timothy Havens*
Thesis Co-advisor:	*Dr. Jin Choi*
Committee Member:	*Dr. Anthony Pinar*
Committee Member:	*Dr. Evan Lucas*
Department Chair:	*Dr. Jin Choi*

Contents

Author Contribution Statement

This work is the result of the collaboration between the author, Steven Senczyszyn, Kevin Li, Mohamed Salem, and is overseen by Dr. Anthony Pinar and Dr. Timothy Havens. The paper [1] has been revised and updated for this thesis, following publication in peer reviewed conference proceedings:

> Kangas, D. J.; Salem, M.; Li, K.; Ryynanen, T.; Senczyszyn, S.; Pinar, A. J.; Havens, T. C.; Price, S. R. In *Proc. SPIE Defense and Commercial Sensing*, 2024.

It was developed alongside a paper [2] with a related implementation, which focused on indoor localization and mapping:

> Senczyszyn, S.; Pinar, A. J.; Salem, M.; Donogue, E.; Wills, S.; Broestl, M.; Webb, A. J.; Havens, T. C.; Price, S. R. In *Proc. SPIE Defense and Commercial Sensing*, 2024.

Acknowledgments

I would like to express my sincere gratitude to Dr. Timothy Havens, my advisor, for his guidance and scholarly insight throughout the entire process of conducting this research and crafting this thesis. I'd like to thank my committee members Dr. Anthony Pinar and Dr. Evan Lucas for propelling my work further than I could imagine. I would like to acknowledge the members of the SENSE enterprise who have generously contributed their time and effort to this project. Lastly, I am indebted to my friends and family for their endless love, encouragement, and understanding during this academic journey.

The experiments described and the resulting data presented herein, unless otherwise noted, were funded by SOSSEC under ERDC-MECI-PLA-0003, managed by the US Army Engineer Research and Development Center, and by the US Navy under grant N00174-23-1-0003.

Abstract

The demand for autonomous vehicles (AVs) is rising across both military and civilian sectors. These unmanned systems offer numerous advantages, such as improved efficiency, safety, and adaptability. Addressing this demand requires the development of resilient and versatile autonomous vehicles crucial for the transport and reconnaissance markets.

The sensory perception of autonomous vehicles of any kind is paramount to their ability to navigate and localize in their environment. Factors such as sensor noise, erroneous readings, and deliberate attacks should all be considered when developing a robust autonomous system. This work aims to quantify the degradation of sensor data which causes mapping algorithms to fail and properly localize.

In this thesis, we explore five different simulated LIDAR perturbation models and their effects on mapping indoor and outdoor locations. The noise models are categorized into two types: *fake* and *real* point returns. A similarity metric is utilized to quantify the degradation of the resulting point clouds. An advantage of this approach, over implementing perturbations in physical environments, is the ability to test challenging or impractical perturbations on a simulated system.

Our findings confirm that increased levels of noise correlate with elevated errors in

mapping. We discuss the process of cascading failures and the additional overlaid topography that is produced. We also discovered that certain types of sensor noise affect indoor mapping more than outdoor, particularly when the noise is localized.

In future research, we plan to investigate methods to physically implement the noise models employed in this study and to develop strategies for mitigating their impact on autonomous navigation.

1 Introduction

Autonomous vehicles perform localization, mapping, and navigation tasks using multiple different types of sensors: LIDAR, radar, cameras, inertial sensors, and odometry are among the most used. Depth sensors are among the most important as they not only inform the system of potential obstacles, but also provide information about the world used for localization and mapping. Localization is the task of determining one's position within an environment, mapping creates a record of the objects and navigable paths within an environment. In unknown environments, autonomous systems must often perform *simultaneous localization and mapping* (SLAM), which will be discussed further in this paper. We will investigate the effects of sensor perturbations on SLAM in indoor environments, as well as unstructured, outdoor environments.

Depth measurements can be acquired in different ways [3], but are predominantly measured by active sensors, such as LIDAR, radar, laser-ranging, or passive ranging sensors, such as RGB-D cameras, which use sequences of images or multiple cameras (stereo-imaging). Machine learning approaches have also been proposed [4, 5, 6], which typically estimate depth from images collected from cameras [7]. These sensors can also be combined to provide more robust performance across different sensing scenarios—say rain, fog, or snow—or to simply improve the overall estimate by sensor fusion [8, 9, 10]. In this paper, we will focus on the use of LIDAR, with the experiments

we perform using a Velodyne VLP-16 LIDAR: a common LIDAR used in autonomous vehicles.

Sensor perturbations or degradation can have a significant effect on the performance of autonomous systems. Perturbations can take many forms: environmental causes such as fog or dust, human causes such a bright lights or reflective signage, and adversarial attacks such as patches or patterns, to name a few.

The literature abounds in work discussing perturbations on autonomy sensors and mitigation strategies for developing more robust systems. Two good surveys on the subject include the works by Thing and Wu [11], and Pham and Xiong [12]. A more general review of adversarial attacks on AI and mitigation strategies is the work by Qiu et al. [13], which is an important general concern as most current state-of-the-art autonomy approaches are data-driven, i.e., they rely on machine learning. In a previous paper [14], we focused on the effects of perturbations on ROS-implemented mapping algorithms in an indoor environment, specifically investigating Gmapping [15, 16] and HectorSLAM [17].

1.1 Related Work

The experiments presented in this paper are meant to be a fundamental treatment of how sensor perturbations or degradation affect autonomy in an unstructured, large-scale outdoor environment, specifically focusing on ROS-based SLAM. Numerous works have looked at the practical implications of erroneous sensor measurements in autonomous systems [18, 19, 20, 21, 22, 23, 24, 25, 26]. We now expand more on the relevant works to this paper.

Shin et al. [20] explored saturation effects on LIDAR by directing 905nm lasers of varying power levels toward a target LIDAR. Their experiments showed that various attack strategies with these lasers can lead to either erroneous LIDAR returns or to blind spots in the resulting point cloud. The erroneous returns could either add noise to the point cloud or "ghost" objects (i.e. objects that are not actually there); the blind spots could eliminate or disguise *actual* objects. This work underscores the fact that perception sensors, particularly LIDAR, are susceptible to various modes of noise other than "standard" measurement uncertainty, which demonstrates the need for the robust processing in the an autonomous agent's perception software.

Pekaric et al. [27] exploit vulnerabilities within ROS and Gazebo, specifically within

the MAVlink protocol, that demonstrate the potential for spoofing and jamming attacks. One such attack exploits LIDAR, and the framework is provided for malicious attacks to manipulate the object avoidance algorithm through injected LIDAR signals that then allows for the target to be controlled. This work is demonstrated in simulation and underscores the need for real-world experiments to further understand the impact that these attacks have on the performance and enhancement of autonomous systems.

Wang et al. [28] investigated malicious LIDAR-based deceptions using a limited number of injected artificial points. It is demonstrated that only 20 points are required to spoof an object detection algorithm into detecting a fake car with an 89% success rate. This work underscores the vulnerability that exists within LIDAR-based 3D object detectors and how susceptible they can be to spoofing without proper precautions.

Jin et al. [29] further explored physical attacks against commercial LIDAR systems by injecting malicious laser signals into the sensor on an autonomous vehicle. It demonstrated that the manipulation of point clouds can result in spoofed 3D objects by either generating artificial objects or hiding objects in the physical world. This work shows how the noise models used in the following study translate into potential real-world disturbances and emphasizes the need for robust countermeasures.

Cao et al. [30] explore methods of mitigating malicious LIDAR spoofing attacks, which are detailed in the previously referenced studies. Object detection algorithms

can prevent spoofing by treating ignored patterns as invariant physical features. This work first implements LIDAR attacks that achieve an 80% success rate, which is then mitigated to 5.5% through the implementation of sequential fusion views. This work emphasizes the benefit that can be provided by robust algorithm design and the need for further experimental testing.

Yang et al. [31] implement a novel 3D LIDAR-SLAM algorithm that is designed for optimal performance in degraded scenarios. Compared with several other popular LIDAR-SLAM algorithms, the presented algorithm quantitatively and qualitatively achieves better performance regarding state estimation, real-time performance, and mapping accuracy when ablations are applied to the sensor input. This work highlights the potential benefits that can be achieved when accounting for less-than-ideal sensor inputs.

2 Perturbation Models

The sensor perturbation models investigated here are designed to explore the behavior of autonomy in the face of sensor malfunctions or degradation. These models are crucial for testing the resilience and adaptability of autonomous systems when dealing with imperfect sensor data. Our research includes an expanded focus on 3D noise, adding a new dimension to the perturbations that can affect depth images and point clouds. This advancement allows for a more realistic simulation of sensor errors in environments that require three-dimensional spatial understanding. Tables 2-4 contain the parameters we used in our experiments, showing the minimum and maximum values used and the step used across that range. These parameter values were chosen to illustrate well the transition of good mapping performance to poor mapping performance. Figure 2.1 shows some example point clouds contaminated by these models.

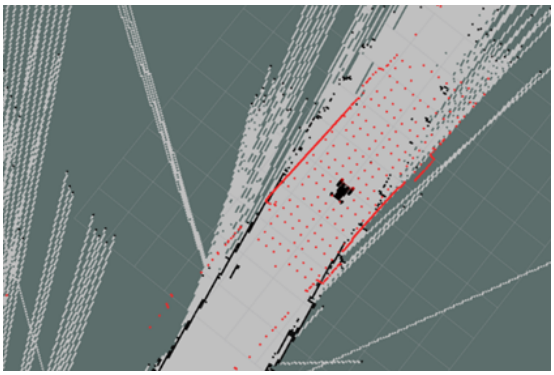
There are two types of models: (a) *fake* point returns, and (b) position-altered *real* point returns. In this updated approach, (a) *fake* point returns are added to or subtracted from the measured laser scan or point cloud data in three dimensions. This method simulates physical objects that either erroneously appear in or are missing from the sensor's field of view, thus testing the system's ability to cope with unexpected or missing data in a 3D space. The (b) *real* point returns in the laser scan

or point cloud are altered, reflecting changes in the 3D position of points. This can simulate scenarios where the accurate position of objects is distorted due to sensor inaccuracies, affecting the system's spatial understanding and decision-making processes. The 'Clouds Grid' noise model generates several point returns spatially clustered together to simulate clouds or blobs in the laser scan. This is implemented by first generating a 2D grid of independent (spatially-uncorrelated) noise centered on the robot, then filtering the noise with a 2D Gaussian filter, yielding spatially-correlated noise. Finally, we compare the values of this noise against a threshold and delete point returns below this level. Due to the nature of the Clouds Grid model it was not formatted to a 3D space.

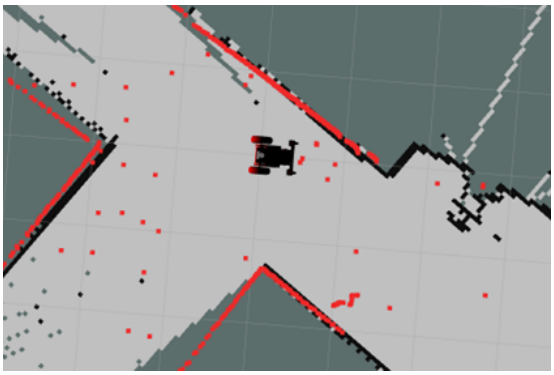
Table 1
Sensor Perturbation Models

Model	Type*	Description	Parameters
Clouds Grid (Figure 2.1a)	A	Spatially-correlated noise added in a square around robot	{width, threshold, no. points}
Snow (Figure 2.1b)	B	Spurious points sampled from a uniform distribution inside a cube centered on a robot	{no. points,}
Snow Local (Figure 2.1c)	A	Spurious points drawn from a normal distribution with mean located relative to robot frame (i.e., a "swarm of mosquitoes")	{no. points, mean, variance}
Modify Point Cloud (Figure 2.1e)	B	Uniform-distributed noise added to all points in measured point cloud	{width of distribution (cm)}
Modify Point Cloud Local (Figure 2.1f)	B	Uniform-distributed noise added to measured points within a radius around robot	{radius (m), width of distribution (cm)}

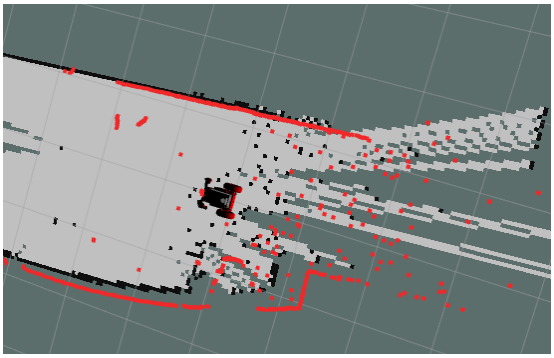
*Model type indicates: (A) points added to measured point cloud, or (B) measured point cloud is altered.



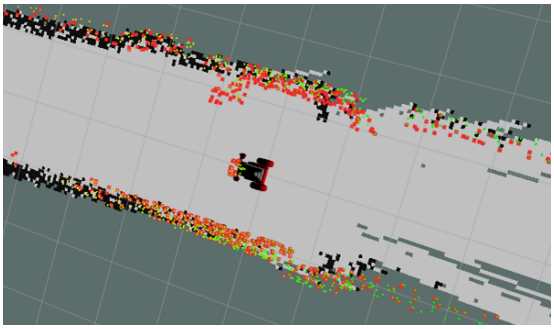
(a) Clouds Grid



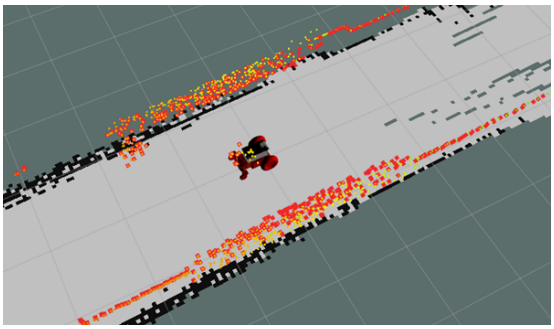
(b) Snow



(c) Snow Local



(d) Modify Point Cloud



(e) Modify Point Cloud Local

Figure 2.1: Examples of sensor perturbation models applied to a sample frame from the experiment arena. Red points indicate measured point cloud; black points indicate current map boundaries.

Table 2
Parameters Used in EERC Experiment

Model	Var.*	Range	Other Parameters
Clouds Grid	no. points	[200, 480]	width = 20m, threshold = 98
Snow	no. points	[200, 1,600]	
Snow Local	no. points	[200, 1,600]	mean = (-2m, 0m), volume = $6m^3$
Modify Point Cloud	width (cm)	[4, 60]	
Modify Point Cloud Local	width (cm)	[5, 110]	radius = 5m

Table 3
Parameters Used in Fisher Experiment

Model	Var.*	Range	Other Parameters
Clouds Grid	no. points	[240, 520]	width = 20m, threshold = 98
Snow	no. points	[100, 1,500]	
Snow Local	no. points	[100, 1,500]	mean = (-2m, 0m), volume = $6m^3$
Modify Point Cloud	width (cm)	[5, 75]	
Modify Point Cloud Local	width (cm)	[25, 375]	radius = 30m

*Indicates perturbation model parameter that is varied in experiment.

Table 4
Parameters Used in Figure 8 Experiment

Model	Var.*	Range	Other Parameters
Clouds Grid	no. points	[280, 560]	width = 20m, threshold = 98
Snow	no. points	[800, 2,200]	
Snow Local	no. points	[1000, 2,400]	mean = (-2m, 0m), volume = $6m^3$
Modify Point Cloud	width (cm)	[5, 210]	
Modify Point Cloud Local	width (cm)	[25, 375]	radius = 35m

*Indicates perturbation model parameter that is varied in experiment.

3 Autonomy Approach

The autonomy algorithm that was applied for this work is HDL Graph SLAM, proposed by Koide et al. [32]. HDL Graph SLAM is an open-source ROS package that uses 3D LIDAR to produce a real-time 6 *degree-of-freedom* (DOF) localization estimate along with a simultaneous map. The algorithm can also support other sensor measurement constraints, such as *inertial measurement unit* (IMU), and GPS. In this work, we utilized 3D LIDAR, and an onboard IMU in our mapping solution.

The work of Koide et al. presents a robust system that leverages 3D LIDAR technology for addressing SLAM problems by combining two phases: i) offline environment mapping and ii) online sensor pose estimation. In the offline phase, a detailed 3D map of the area is generated using Graph SLAM which incorporates the ground plane constraints for indoor settings and GPS for outdoor environments to compensate for accumulated rotation errors from scan matching. In the pre-mapped environment, the system updates its position by merging the *normal distributions transform* (NDT) scan matching algorithm with a prediction of its angular velocity, processed via an *unscented Kalman filter* (UKF).

HDL Graph SLAM incorporates a loop detection algorithm similar to the work by

Nelson [33]. This involves the identification of loop candidates by analyzing translation distance and trajectory length between nodes. Nodes represent elements from the mapping process, signifying discrete sensor poses or landmark positions that the system has encountered. By handling the connection of these nodes that is derived from sensor data, such as LIDAR scans and GPS measurements, the system constructs a graph. The graph is then optimized to minimize discrepancies between observed and predicted state.

In indoor settings, where the ground is uniformly flat, the ground plane detection ensures that the pose estimate graph reflects a consistent plane. Conversely, outdoor environments, where the ground is not flat, the algorithm can utilize GPS-based positioning. GPS data provides an external reference that associates to each pose node, allowing for adjustments in elevation.

4 Robotics Platform

To collect data for the experiment, a versatile robotics platform for the various sensors was required. Initially experiments were conducted using a RoverRobotics Rover Zero 3, shown in Figure 4.1. This platform was used previously for experiments using a 2D LIDAR scanner [2]. However, challenges arose when attempting to integrate additional sensors, prompting the selection of a different platform.

Figure 4.1: RoverRobotics Rover Zero 3 2WD

The chosen robotics platform was developed by the US Army Combat Capabilities Development Command (DEVCOM) for use in research of small unmanned ground vehicles. The robot, shown in Figure [34], is called the GVR-BOT [34]. It serves as a modular platform for integrating various sensors and actuators. The robot features a robust chassis containing an ARM CPU, a network switch, wheel encoders, and an IMU.

Figure 4.2: NVIDIA Jetson AGX Orin embedded computer

For this application an additional processing computer was included to sufficiently collect the real-time data from the sensors. The NVIDIA Jetson AGX Orin in Figure 4.2 was selected and mounted externally using 3D printed fixtures. Further networking equipment was included for Ethernet communication between the companion computer, GVR-BOT, and the 3D LIDAR.

Two primary sensors were integrated into the GVR-BOT's platform for this experiment: the Velodyne VLP-16 3D LIDAR and the Phidgets 3/3/3 IMU. These sensors, as well as odometry from the wheel encoders, are essential for the selected SLAM algorithm.

The Velodyne VLP-16 3D LIDAR sensor was chosen for its ability to provide high-resolution 3D scans of the robot's surroundings. This sensor enables the GVR-BOT to

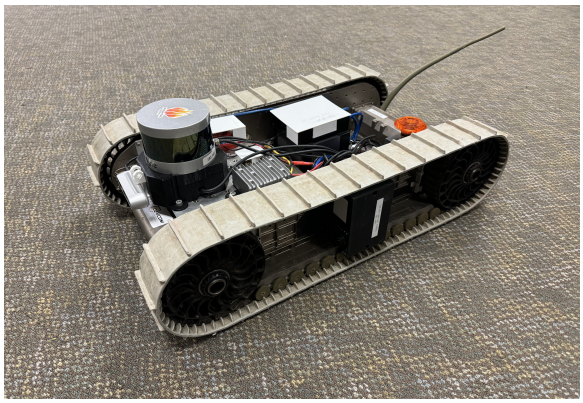


Figure 4.3: GVR-BOT robotic platform

create detailed maps of its environment, crucial for navigation and obstacle avoidance.

Additionally, a Phidgets 3/3/3 IMU was included in the sensor suite. While the GVR-BOT does come equipped with its own IMU, the external Phidgets IMU was integrated for redundancy and simplified integration within our ROS environment. This redundancy ensures robustness in orientation and motion sensing, critical for proper localization.

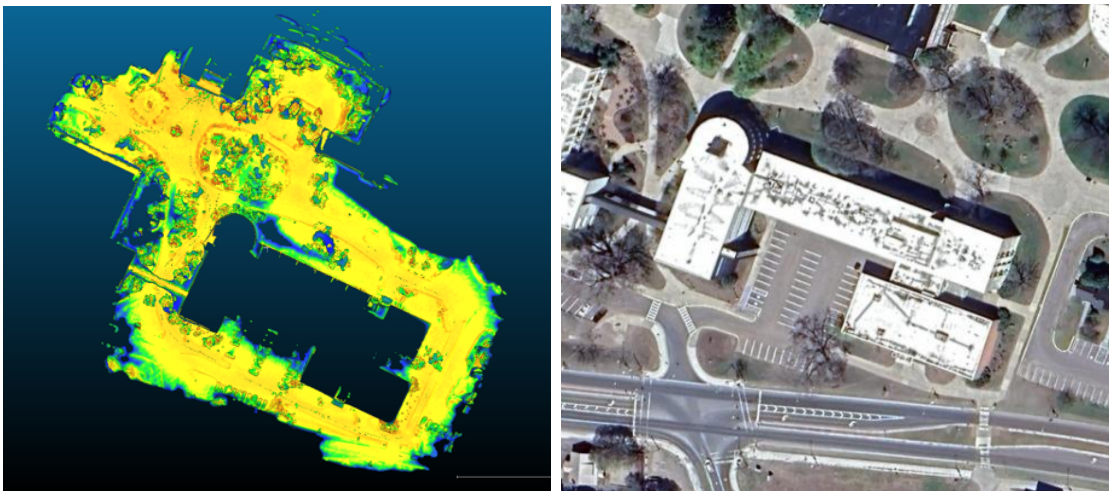
5 Experiment

The experiments performed were designed to quantify the effect of LIDAR sensor perturbations on the ability of an autonomous ground vehicle to localize and map an indoor scenario, shown in Figure 5.1, as well as two large-scale outdoor spaces, shown in Figures 5.2 and 5.3. The indoor location serves as a baseline for ideal mapping with well-defined walls and flat terrain. The outdoor locations were captured on a college campus and provide diverse surroundings analogous to a low density urban environment. All locations included light pedestrian traffic.

To compile a sufficient dataset, ROS bags of sensor data were collected as the robot was manually piloted through predefined courses. Multiple runs were made utilizing the same route, totalling up to 5 bags for each location. The bags contain data from the 3D LIDAR, IMU, and odometry from the onboard wheel encoders.

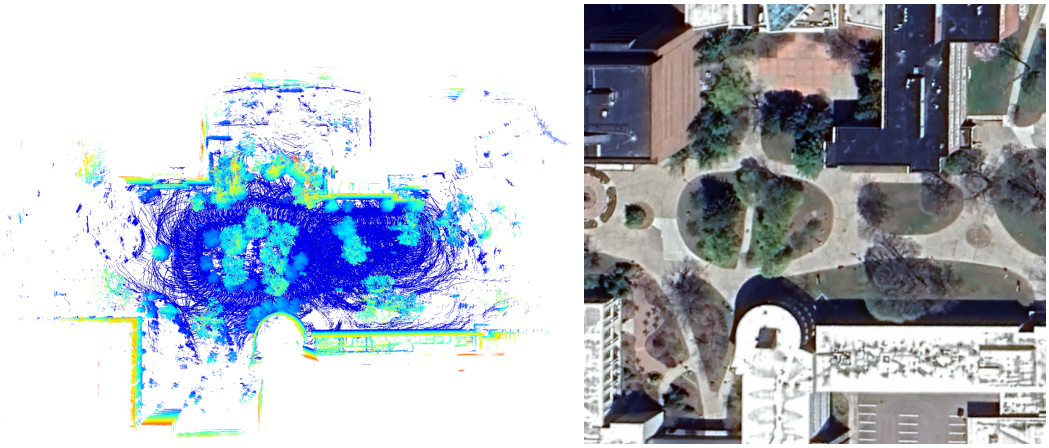
(a) EERC Ground Truth (b) EERC Satellite image

Figure 5.1: Image (a) shows the ground truth LIDAR scan of the EERC indoor scenario. Image (b) is the satellite image of the same location (EERC).



(a) Fisher Hall Building Ground Truth (b) Fisher Hall Building Satellite image

Figure 5.2: Image (a) shows the ground truth LIDAR scan, courtesy of the MTU Geospatial Research Facility (GRF), collected with a handheld GeoSLAM LiDAR. Image (b) is an aerial view of the same location.



(a) Figure 8 Ground Truth (b) Figure 8 Satellite Image

Figure 5.3: Image (a) shows the unperturbed map produced by HDL Graph SLAM of the Figure 8 scenario. Image (b) is an aerial view of the same location.

For each location, HDL Graph SLAM is used to generate point clouds without sensor perturbations. This initial step is crucial as it establishes a baseline reference point against which subsequent degraded point clouds are compared and evaluated. These

point clouds provide an accurate reflection of the physical environment, capturing distinct features, obstacles, and layouts.

Each of the perturbation models is then applied to the sensor data to quantify their effect on the map generation. The resulting "noisy" point clouds were then compared to the reference using the *iterative closest point* (ICP) algorithm [35]. On each iteration, the ICP algorithm attempts to register the source point cloud to the reference by transforming the source to minimize the error between them. Error is defined as the sum of squared differences for each point in the point clouds; a higher error corresponds with a larger difference between them. Error will be used as the metric to quantify the degradation of the resulting map produced for each noise model.

Since error is not required to be calculated in real-time, processing time is relatively abundant. More iterations of ICP will result in a more accurate registration to best quantify the degradation of the point cloud. The criteria for stopping iterations is set as a threshold of the difference in error between successive iterations. The threshold we use in this experiment is 0.001.

The main limitation of ICP in this case is its slow convergence time for the large number of points in our experiments. The point clouds reach upwards of 4 million points each, resulting in trillions of comparisons. This proves intractable with a pool of over 3,000 point clouds. In order to reduce the processing time from around 30 minutes per pair, the point clouds were decimated to 10,000 points each. Points are

uniform randomly selected from the measured clouds for removal. In our testing, the error produced in comparing the decimated maps and the original maps followed the same trends.

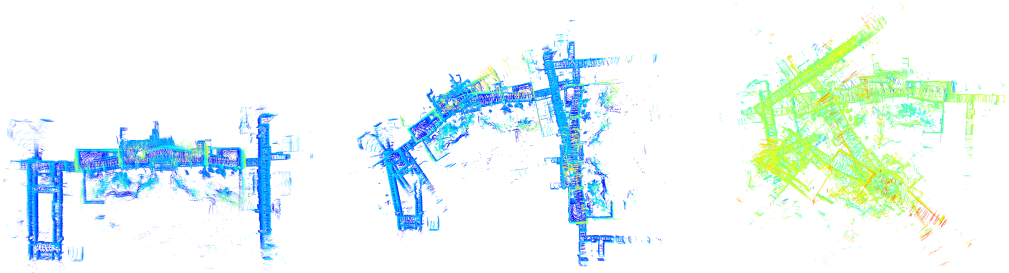
In some experiments we observed failed ICP registration at lower perturbation levels. Running the mapping process over multiple runs ideally mitigates these 'outliers.' Furthermore, we attempted to optimize the hyperparameters of the ICP algorithm to prevent such occurences, including the step size and convergence threshold. However, we are aware that outliers still could occur and may be present in our data. We now turn to the experiment results and discussion thereof.

6 Results & Discussion

Following the processing of sensor data with added perturbations, we collected the resulting point clouds and quantified the error for each perturbation model. The outcomes of our experiments, measuring the ICP error versus perturbation level for each model and scenario, are presented in Figures 6.4-6.6. These figures indicate the mean at each perturbation level with an $\times$ symbol, and the error bars indicate the maximum and minimum values. These results provide an overview of the impact of perturbation types and levels on the ability for HDL Graph SLAM to produce a coherent point cloud.

Overall, the results show noticeable point cloud degradation as the perturbation severity is increased. Moreover, the data highlights a substantial increase in the deviation of ICP error for experiments at higher perturbation levels. This trend highlights the sensitivity of the system to perturbations, particularly at higher severity, where HDL Graph SLAM fails to produce an acceptable map. An example of the degradation of maps proportional to perturbation level is show in Figures 6.1-6.3, which illustrate the effects of the Snow perturbation model.

For the Clouds Grid models, we observed high similarity of the perturbed map to the reference even at a considerable number of points added. Only a few outliers register



(a) EERC Add Snow - 500 (b) EERC Add Snow - 700 (c) EERC Add Snow - 1000

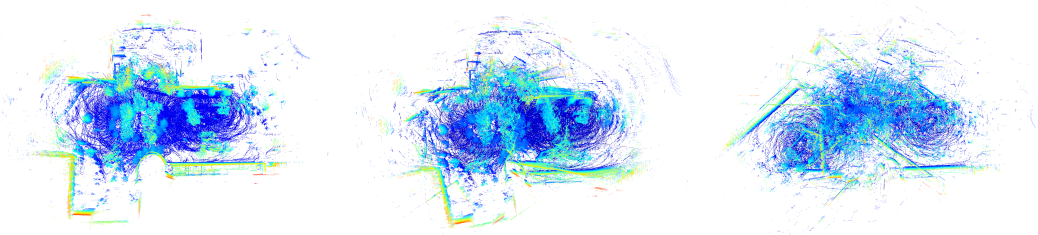
Figure 6.1: Comparison of mapping performance for the EERC indoor scenario at different perturbation levels.

(a) Fisher Add Snow - 700 (b) Fisher Add Snow - 1200 (c) Fisher Add Snow - 1400

Figure 6.2: Comparison of mapping performance for the Fisher outdoor scenario at different perturbation levels.

above an error of 600, and the deviation remained low. We expect this behavior comes from the SLAM algorithm perceiving the cloud points as 'dead zones,' either adding them to the map or ignoring them outright until the points are sufficiently dense.

The Modify Point Cloud model was found to be the most consistently disruptive for each of the locations, providing an almost linear response to increased distribution width. It is particularly interesting that the variation between runs stayed extremely

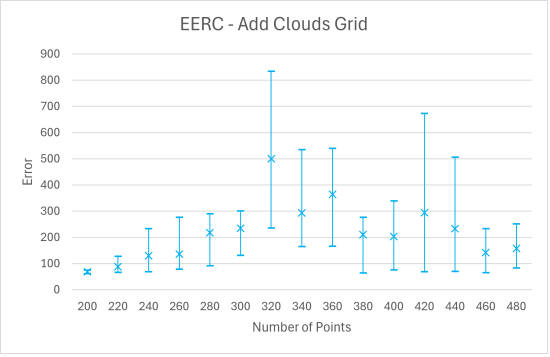


(a) Figure 8 Add Snow - 1000 (b) Figure 8 Add Snow - 1500 (c) Figure 8 Add Snow - 2000

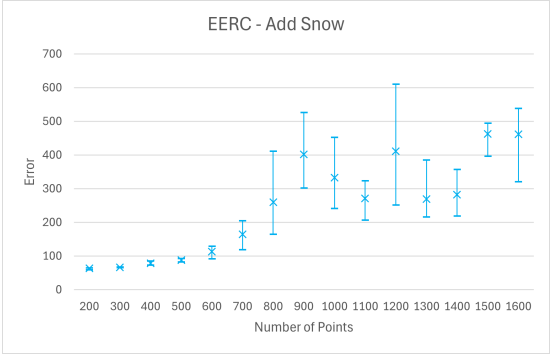
Figure 6.3: Comparison of mapping performance for the Figure 8 outdoor scenario at different perturbation levels.

consistent until significant perturbation occurs. When the width of the distribution exceeded 40 cm, the resultant map was significantly degraded, as shown in view (d) of Figures 6.4–6.6. Most notably in Figure 6.4(d), we recognize substantially increased ICP error with the Modify Point Cloud Local model, when the width of the distribution surpasses 35 cm. Interestingly, the results seem to indicate a difference in degradation for different perturbation models between indoor and outdoor locations, having an almost constant impact on the outdoor locations: see Figures 6.5(d) and 6.6(d).

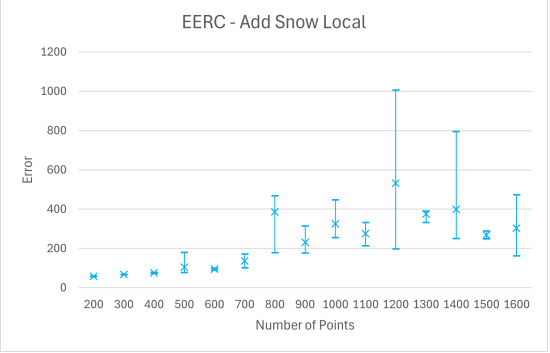
Further examination of the results shows that noise models incorporating local perturbations (such as Clouds Grid, Snow Local, or Modify Point Cloud Local) appear to exert a noticeably greater impact on point clouds in indoor environments. This conclusion seems logical given the relatively close features found in indoor settings, which are much more prone to registration errors when the only features available for localization are severely degraded. In contrast, outdoor locations benefit from containing features that are considerably farther apart, making alignment easier when



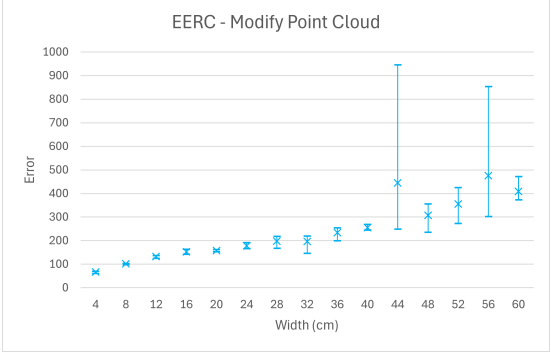
(a) Add Clouds Grid



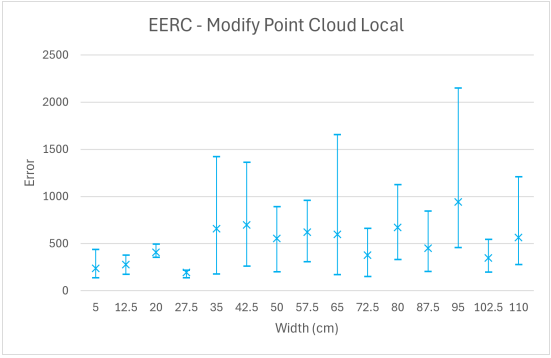
(b) Add Snow



(c) Add Snow Local

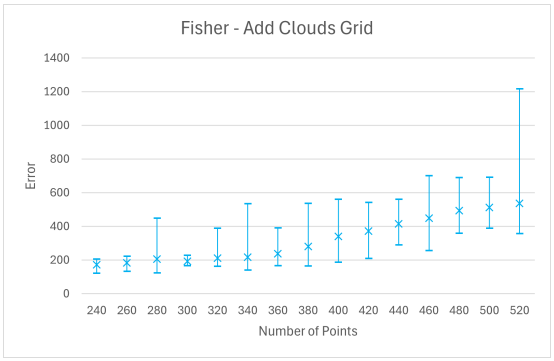


(d) Modify Point Cloud

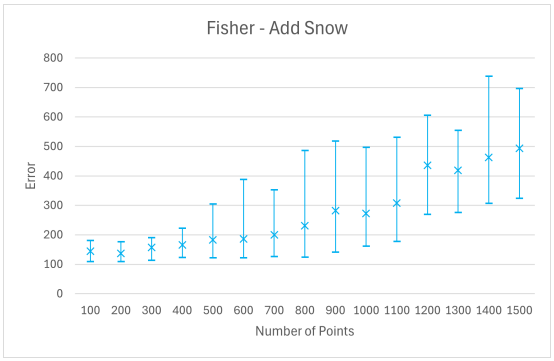


(e) Modify Point Cloud Local

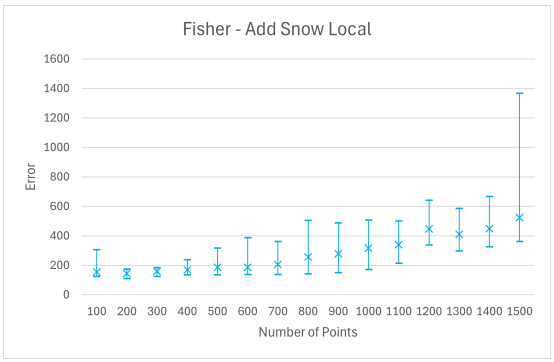
Figure 6.4: Plots of ICP error versus perturbation level for each perturbation model in the EERC indoor scenario. Error bars indicate mean, minimum, and maximum ICP error values for each noise level.



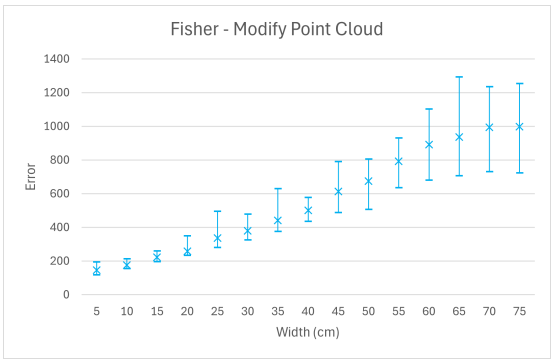
(a) Add Clouds Grid



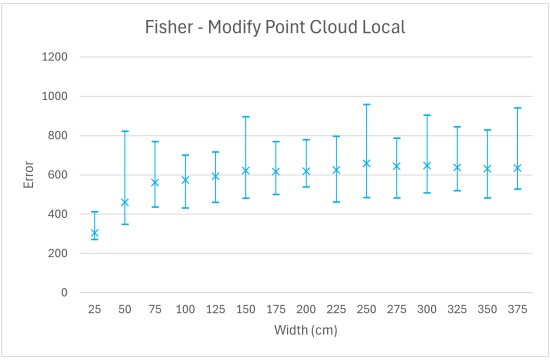
(b) Add Snow



(c) Add Snow Local

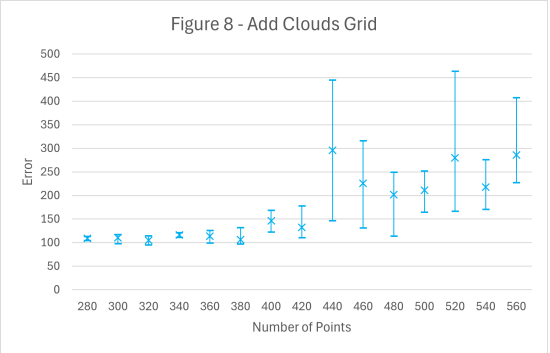


(d) Modify Point Cloud

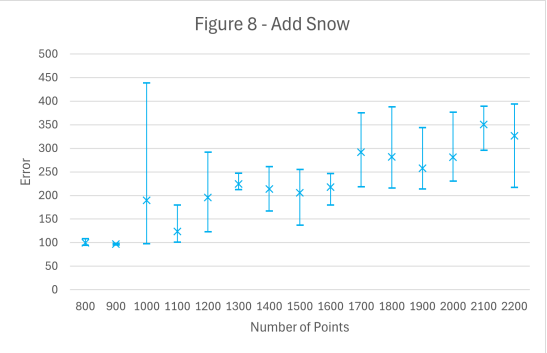


(e) Modify Point Cloud Local

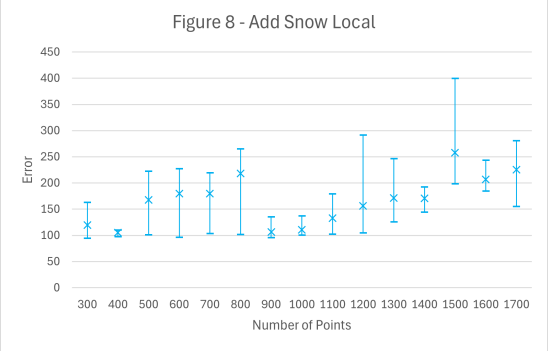
Figure 6.5: Plots of ICP error versus perturbation level for each perturbation model for the Fisher outdoor scenario. Error bars indicate mean, minimum, and maximum ICP error values for each noise level.



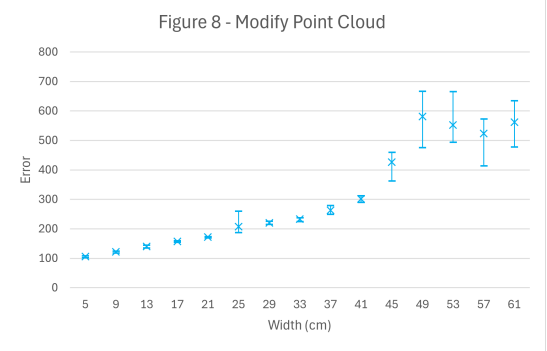
(a) Add Clouds Grid



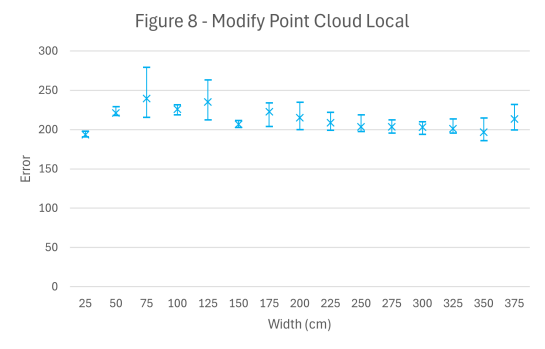
(b) Add Snow



(c) Add Snow Local



(d) Modify Point Cloud



(e) Modify Point Cloud Local

Figure 6.6: Plots of ICP error versus noise level for each perturbation model for the Figure 8 outdoor scenario. Error bars indicate mean, minimum, and maximum ICP error values for each noise level.

adjacent features are disturbed.

We hypothesize that the majority of the error comes from the SLAM algorithm failing to properly register the current scan with the previous, with the perturbation points (or deletions) causing erred registration. Registration errors can create a secondary layout of points that are improperly aligned, which can cascade into further mapping failure and additional overlaid topography.

For a comprehensive discussion on additional results, please refer to Appendix A. In this section, we provide an alternative scenario and explain the unexpected results of an inconsistent increase in error level.

7 Conclusion & Future Work

This work explored the effects of perturbed LIDAR measurements on an autonomous vehicle mapping a complex, large-scale outdoor environment. The mapping algorithms investigated was HDL Graph SLAM. Map quality was measured by first registering the perturbed map with a ground-truth map, then quantifying the difference between the two using the final ICP error.

The perturbations tested in this work were fundamental types of sensor perturbations, either adding to or deleting from the measured point cloud, or altering the measured point cloud. These perturbations were accomplished by "highjacking" the sensor measurements within ROS, altering them within a perturbation ROS node, and then publishing the perturbed measurements for use by the mapping algorithms. The advantage of this approach, as compared to implementing perturbations in the real world, is that perturbations that are difficult or potentially impossible to physically realize can be tested on a real-world system.

One conclusion from this study is that indoor and outdoor mapping are affected differently by perturbations. Indoor mapping was significantly affected by local perturbations close to the autonomous system. We surmise that this is because the features of the indoor space are close to the system and do not extend beyond that local region

(the robot cannot "see" through walls). Local perturbations were not as relatively disruptive in the outdoor scenarios as the system could "see" over longer distances, with mapping enabled by both close-in point cloud features as well as features at a further distance.

In the future, we will examine real-world instantiations of perturbations to both validate our hybrid *in-situ*/real-world experiments and also to examine the complexity of producing such perturbations. Furthermore, these experiments will lead to algorithm enhancements to improve the robustness of mapping.

Another area of future work is to look at metrics that examine the local quality of map point clouds. We observed that some of the larger ICP error levels were seen when only one section of the map failed, causing a cascaded error on the rest of the map creation, even if the remaining map was locally correct. We aim to address this in future work.

References

[1] Kangas, D. J.; Salem, M.; Li, K.; Ryynanen, T.; Senczyszyn, S.; Pinar, A. J.; Havens, T. C.; Price, S. R. In *Proc. SPIE Defense and Commercial Sensing*, 2024.

[2] Senczyszyn, S.; Pinar, A. J.; Salem, M.; Donogue, E.; Wills, S.; Broestl, M.; Webb, A. J.; Havens, T. C.; Price, S. R. In *Proc. SPIE Defense and Commercial Sensing*, 2024.

[3] Hebert, M. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, Vol. 1, pages 102–110 vol.1, 2000.

[4] Zhao, C.; Sun, Q.; Zhang, C. *Sci. China Technol. Sci.* **2020**, *63*, 1612—1627.

[5] Garg, R.; Wadhwa, N.; Ansari, S.; Barron, J. T. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7628–7637, 2019.

[6] Godard, C.; Mac Aodha, O.; Firman, M.; Brostow, G. J. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3828–3838, 2019.

[7] Matthies, L. *Journal of Robotic Systems* **1992**, *9*(6), 787–816.

[8] Song, H.; Choi, W.; Lim, S.; Kim, H. In *2014 CACS International Automatic Control Conference (CACS 2014)*, pages 144–149, 2014.

[9] Dieterle, T.; Particke, F.; Patino-Studencki, L.; Thielecke, J. In *2017 IEEE SENSORS*, pages 1–3, 2017.

[10] Fu, C.; Mertz, C.; Dolan, J. M. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 273–278, 2019.

[11] Thing, V. L. L.; Wu, J. In *IEEE Int. Conf. Internet of Things (iThings)*, pages 164–170, 2016.

[12] Pham, M.; Xiong, K. *ArXiv* **2020**, *abs/2007.08041*.

[13] Qiu, S.; Liu, Q.; Zhou, S.; Wu, C. *Applied Sciences* **2019**, *9*(5), 909.

[14] Pinar, A. J.; Webb, A. J.; Brown, J. L.; Havens, T. C.; Alvey, B.; DeSouza, G. N.; Anderson, D. T.; Price, S. R. In *Proc. SPIE DSS*, number 11746, page 117461F, 2021.

[15] Grisetti, G.; Stachniss, C.; Burgard, W. *IEEE Transactions on Robotics* **2007**, *23*, 34–46.

[16] Grisetti, G.; Stachniss, C.; Burgard, W. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005.

[17] Kohlbrecher, S.; Meyer, J.; von Stryk, O.; Klingauf, U. In *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, 2011.

[18] Cheng, C. H.; Chen, C. Y.; Chen, J. D.; Pan, D. K.; Ting, K. T.; Lin, F. Y. *Optics Express* **2018**, *26*(9), 12230–12241.

[19] Petit, J.; Stottelaar, B.; Feiri, M.; Kargl, F. In *Black Hat Europe*, 2015.

[20] Shin, H.; Kim, D.; Kwon, Y.; Kim, Y. In *CHES*, pages 445–467, 2017.

[21] Changalvala, R.; Malik, H. *IEEE Access* **2019**, *7*, 138018–138031.

[22] Changalvala, R.; Malik, H. In *IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1219–1225, 2019.

[23] Yan, C.; Xu, W.; Liu, J. *DEF CON* **2016**, *24*(8), 109.

[24] Sun, J.; Cao, Y.; Chen, Q.; Mao, Z. M. *ArXiv* **2020**, *abs/2006.16974*.

[25] Cao, Y.; Xiao, C.; Cyr, B.; Zhou, Y.; Park, W.; Rampazzi, S.; Chen, Q.; Fu, K.; Mao, Z. M. *Proc. ACM SIGSAC Conf. Computer and Communications Security* **2019**.

[26] Hau, Z.; Co, K.; Demetriou, S.; Lupu, E. C. *ArXiv* **2021**, *abs/2102.03722*.

[27] Pekaric, I.; Arnold, D.; .; Felderer, M. In *International Conference on Software Quality, Reliability, and Security Companion*, pages 44–53. IEEE, 2022.

[28] Wang, J.; Li, F.; Zhang, X.; Sun, H. *IEEE Transactions on Multimedia* **2024**, *26*(6), 2686–2699.

[29] Jin, Z.; Ji, X.; Cheng, Y.; Yang, B.; Yan, C.; .; Wu, X. In *IEEE Symposium on Security and Privacy (SP)*, pages 1822–1839. IEEE, 2023.

[30] Sun, J.; Cao, Y.; Chen, Q. A.; Mao, Z. M. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 877–894. USENIX Association, 2020.

[31] Yang, L.; Ma, H.; Wang, Y.; Xia, J.; Wang, C. *Sensors* **2022**, *22*(8), 3063.

[32] Koide, K.; Miura, J.; Menegatti, E. *International Journal of Advanced Robotic Systems* **2019**, *16*(2), 1729881419841532.

[33] B(erkley) L(ocalization) A(nd) M(apping). Nelson, E. `https://github.com/erik-nelson/blam`.

[34] Gvr-bot user's guide. US Army CCDC, **2019**.

[35] Zhang, J.; Yao, Y.; Deng, B. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2022**, *40*(7), 3450–3466.

[36] Yadav, D. H. S.; Ansari, A. Autonomous vehicles camera blinding attack detection using sequence modelling and predictive analytics Technical Report 2020-01-0719, SAE, **2020**.

[37] Lewis, G. D.; Santos, C. N.; Vandewal, M. In *Proc. SPIE*, page 1116108, 2019.

[38] Ritt, G.; Eberle, B. *Opt. Eng.* **2017**, *56*(3), 033108.

[39] Chowdhury, A.; Karmakar, G.; Kamruzzaman, J.; Jolfaei, A.; Das, R. *IEEE Access* **2020**, *8*, 207308–207342.

[40] Hau, Z.; Demetriou, S.; Muñoz-González, L.; Lupu, E. C. *ArXiv* **2020**, *abs/2008.12008*.

[41] Bahirat, K.; Prabhakaran, B. In *IEEE Int. Conf. Multimedia and Expo (ICME)*, pages 679–684, 2017.

[42] Shah, S.; Dey, D.; Lovett, C.; Kapoor, A. In *Field and Service Robotics*, 2017.

[43] Unreal engine. Epic Games.

[44] Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. In *Proc. Int. Conf. NIPS*, pages 2672–2680, 2014.

[45] Das, N.; Shanbhogue, M.; Chen, S.-T.; Hohman, F.; Chen, L.; Kounavis, M. E.; Chau, D. H. *arXiv preprint arXiv:1705.02900* **2017**.

[46] Rublee, E.; Rabaud, V.; Konolige, K.; Bradski, G. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[47] Alcantarilla, P.; Nuevo, J.; Bartoli, A. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2013.

[48] Alcantarilla, P. F.; Bartoli, A.; Davison, A. J. In Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C., Eds., *Computer Vision – ECCV 2012*, pages 214–227, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[49] Wang, Z.; Bovik, A.; Sheikh, H.; Simoncelli, E. *IEEE Transactions on Image Processing* **2004**, *13*(4), 600–612.

Appendix A: Additional Results

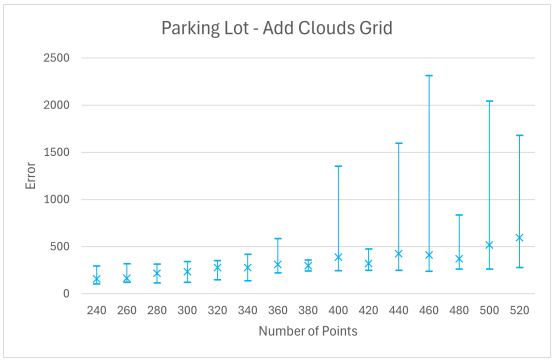
In addition to the results presented in this paper, supplementary LIDAR data from an alternate location is provided in this appendix. These additional findings offer a different perspective on the results of the perturbation techniques and the impact on point cloud coherence.

The additional location chosen for this study was a parking lot. This locale featured numerous highly reflective surfaces on vehicles and several complex structures. These divergent conditions could possibly contribute to the observed disparities in the outcomes, as depicted in Figure A.1.

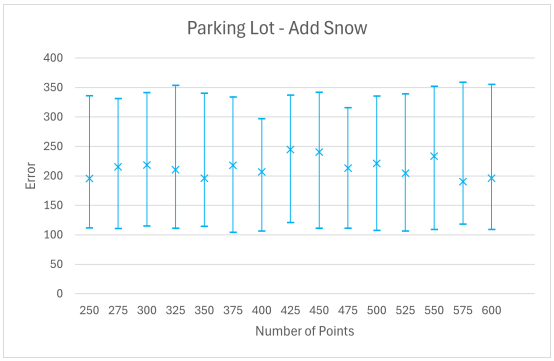
Our analysis reveals similar results to those observed in other locations for all noise models, with one exception. The "Modify Point Cloud Local" model exhibits deviations, showcasing an inconsistent escalation in error rates with increased noise levels. Interestingly, the anticipated pattern of monotonically increasing error is not present. Instead, we observe a noticeable elevation in noise and variance within the data, particularly around the 50cm mark. Beyond a width of 125cm, we regain a constant error level.

This behavior may be attributed to a previously discussed scenario, where a failure

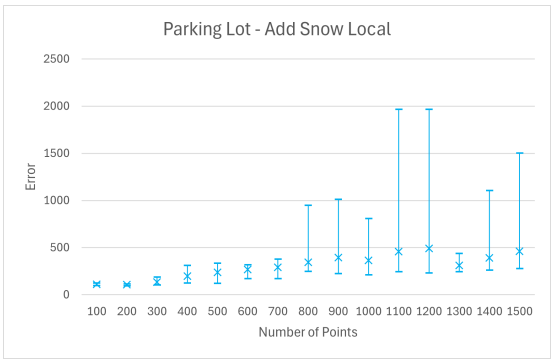
to register a prior scan results in subsequent failures. In this instance, we believe that registration errors initially contributed to an elevated error level. However, as the width increased, successful registration was recovered, and the previous failed registrations are eventually decimated to a point where they become insignificant.



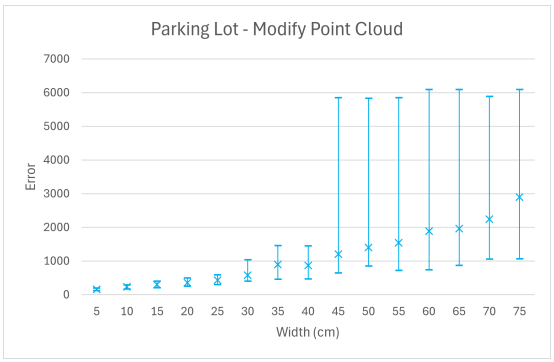
(a) Add Clouds Grid



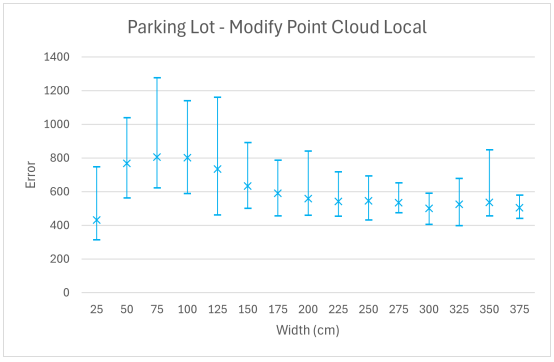
(b) Add Snow



(c) Add Snow Local



(d) Modify Point Cloud



(e) Modify Point Cloud Local

Figure A.1: Plots of ICP error versus noise level for each perturbation model for the Parking Lot outdoor scenario. Error bars indicate mean, minimum, and maximum ICP error values for each noise level.