Data Refining: ROSbag Filtration (RBF) Tool for Optimizing Autonomous Vehicle Sensor Data

Kaitlin E. Asch*†, Aaron Kingery*, Dezhen Song*

*Computer Science and Robotics Department of Engineering, Texas A&M University
College Station, TX, 77843 USA
†Computer Science & Applied Physics, Houston Community College
Houston, TX, 77082 USA

***Abstract*—This research presents a methodology for refining Autonomous Vehicle (AV) sensor data stored in Robot Operating System (ROS) bag files. The project focuses on filtering out noise and irrelevant data, thereby enhancing the quality and reliability of the sensor information critical for AV operations. By employing signal-to-noise ratio (SNR) calculations and the Kalman Filter, the developed node ensures that only high-quality, relevant data is retained. Additionally, a graphical user interface (GUI) is implemented to visualize the processed data, facilitating intuitive analysis and efficient utilization. The proposed approach, tested on ROS bag files gathered from AV data, demonstrated significant improvements in data quality.**

I. Introduction

Efficient data management is crucial for the reliable operation of autonomous vehicles (AV's). These vehicles rely on a multitude of sensors, including LiDAR, radar, cameras, and Inertial Measurement Units (IMU), to navigate and interact with their surroundings. The data collected from these sensors is stored in ROS (Robot Operating System) bag files. However, the raw data often includes irrelevant information, which can obscure vital information and degrade the performance of perception algorithms.

In autonomous driving scenarios, the accumulation of such redundant data poses significant challenges. For instance, consider an AV navigating different environments, from busy urban streets to rural areas. The vehicle captures extensive data, ranging from the movement of pedestrians and vehicles to stationary objects like trees and buildings. This mixture of useful and irrelevant data complicates the data analysis process and can lead to inefficient decision-making, as the system might waste valuable computational resources processing irrelevant data instead of focusing on real-time obstacles or changes in the environment. The implications of poor data management in AV systems are profound. Inefficient data processing can result in slower response times, reduced accuracy in obstacle detection, and ultimately, compromised safety. Moreover, developers and researchers analyzing ROS bag files for improvements in AV algorithms face increased difficulty when the data is cluttered with noise. This highlights the necessity for an effective solution to filter and refine the raw sensor data. To address these challenges, effective data filtration techniques are necessary. By refining the raw data, we can enhance the AV's ability to accurately perceive its environment and make informed decisions. This paper proposes the development of a ROSbag Filtration (RBF) node aimed at filtering out noise and irrelevant information from ROS bag files. The proposed solution leverages signal-to-noise ratio (SNR) calculations and the Kalman Filter for state estimation, ensuring that only high-quality and relevant data is retained. Signal-to-noise ratio (SNR) is a measure used to compare the level of a desired signal to the level of background noise. By calculating the SNR for the sensor data, we can effectively distinguish between useful information and extraneous data. High SNR values indicate data of good quality, which is essential for the accurate functioning of AV systems. The Kalman Filter, a recursive algorithm, is employed for state estimation. It predicts the future state of a system based on past measurements and updates the predictions with new measurements, reducing the uncertainty, allowing for more accurate tracking of the vehicle's position and movement. Additionally, a graphical user interface (GUI) is developed to visualize the processed data, providing an intuitive platform for data analysis and utilization. The GUI allows developers and researchers to interact with the data more effectively, facilitating a better understanding and easier debugging of the AV systems data files. This paper outlines the development of the ROSbag Filtration Node, detailing the technical challenges involved and the methodologies employed to address them. The effectiveness of the proposed approach is demonstrated through tests on ROS bag files, showcasing significant improvements in data quality and processing efficiency.

II. Related Work

The ROSbag Filtration (RBF) node isolates and retains only the most pertinent data, as determined by the user, enhancing the quality of data analysis in autonomous systems. The Argonne Lab's Perception and Connectivity Activity (ALPaCA) dataset offers a comprehensive collection of vision, laser, and GNSS data essential for developing and validating ROSbag filtration nodes. Its structured format facilitates seamless integration and testing of filtration algorithms within a ROS environment, supporting robust AV perception systems

in various settings [1], [2]. Leveraging the ALPaCA dataset allows for extensive validation of the ROSbag filtration node across various real-world scenarios, thereby ensuring that the algorithms are capable of effectively processing diverse data types.

The significance of sophisticated datasets and advanced filtration techniques is further underscored by Chan et al. (2022), who introduced the LIDAR De-Snow Score (DSS). The DSS combines quality and perception metrics to optimize data filtering, addressing challenges with noisy and degraded data [3]. Integrating DSS within our RBF node enhances the filtration process, ensuring that only high-quality, relevant data is retained for analysis. Additionally, Chen et al. (2023) introduce a method for extracting time-windowed ROS computation graphs from ROSbag files, enhancing static analysis capabilities in ROS-based systems. Their approach involves extracting ROS nodes and topic communication data, segmenting this data into user-defined time windows, and constructing computation graphs to illustrate a dynamic system reconfiguration during runtime [4]. By integrating Chen et al.'s methodology, the RBF node can effectively capture the dynamic interactions within the AV system, improving data analysis precision [4]. Wu et al. (2021) propose a variable dimension-based method for roadside LiDAR background filtering, which excludes irrelevant information while preserving objects of interest, reducing computational load and enhancing background filtering accuracy [5]. This approach is particularly beneficial for complex and densely populated environments, such as urban areas, and can be integrated into the RBF node to improve its efficiency in processing high-density sensor data. The development of a ROSbag Filtration (RBF) node is significantly informed by advancements in existing algorithmic approaches within the field. Polat et al. (2015) created the Digital Elevation Models (DEMs) through LiDAR data filtering, decimation, and interpolation. Their research highlights the critical role of data density and topographical accuracy in achieving precise DEMs, providing insights for optimizing data filtration in AV systems [6]. Kingery and Song (2023) developed a method for road boundary estimation using sparse radar signals. Their approach uses a homogeneous road model and incorporates a Dirichlet Process Mixture Model (DPMM) with Mean Field Variational Inference (MFVI) to derive accurate road boundary models, refined through a custom RANSAC algorithm [7]. This method can be adapted into our ROSbag Filtration (RBF) node to refine sensor data quality and ensure reliable boundary detection. Furthermore, the use of probabilistic methods is integral to the framework of the RBF node. Li et al. (2020) introduce a technique for verifying and updating lane markings in high-definition (HD) maps utilizing crowd-sourced images. Their methodology employs Bayesian belief models to address discrepancies between lane markings in HD maps and those detected in images, while accounting for camera pose uncertanties [8]. These algorithmic processes inform the design of the RBF node, ensuring effective filtering and processes of reliable AV sensor data.

In developing AV systems, tools like the integration of ROS with MATLAB through the Robotics Systems Toolbox (RST) enable leveraging MATLAB's prototyping and simulation capabilities alongside ROS's robust development environment [9]. The quality of ROS repositories is also critical for development of reliable autonomous vehicle systems. Santos et al. (2016) emphasized the importance of quality ROS repositories, introducing HAROS, a tool designed to evaluate code quality through static analysis, highlighting the need for high standard in ROS-based software development [10].

Weather condition, often underestimated in AV design, significantly impact sensor data reliability. Adverse weather such as rain, fog, or snow can degrade LiDAR and camera performance, affecting navigation and decision-making. Li et al. (2024) propose the Quasi-Density-Tree (QDTree) method to address noise and sparsity in millimeter-wave (mmw) radar point clouds, which are less affected by weather conditions [11]. This method enhances free space detection and computational efficiency, critical crucial for AV navigation. The core equation:

$$\hat{D} = w_1 \cdot D_1 + w_2 \cdot D_2 \tag{1}$$

where $\hat{D}$ represents the fused distance measurement, D_1 and D_2 are radar and LiDAR measurements, respectively. Weights w_1 and w_2 are coefficients that balance the contributions of each sensor based on reliability. This enhances the RBF node's data fusion, ensuring only relevant and accurate data is retained, thereby improving overall data quality and decision-making. Additionally, Zheng et al. (2024) provide advanced methods for filtering and denoising LiDAR data, forming a solid foundation for enhancing sensor data quality [12].

Future implementations for the RBF node aim to implement AI framework for data analysis. Qu and Zhuang (2023) present a data/model co-driven framework for scalable and dynamic cooperative perception for CAVs, optimizing resource allocation and enhancing perception through learning-assisted methods [13]. This framework supports our RBF node development by filtering out irrelevant information and improving data stream quality when processing AV data files. Na et al. (2024) propose a comprehensive approach to sensor fusion for ADAS applications through radar signal processing and LiDAR depth calculations, offering reliable means of measuring distances and filtering irrelevant data points, which can enhance the RBF node's functionality [14]. Li et al. (2023) explore rectifying dynamic intrinsic camera parameters using the DIME-Net framework. This neural network-based approach addresses variations caused by Optical Image Stabilization (OIS) and improves 3D reconstructions and camera pose estimations. Integrating a similar data-driven approach can dynamically adjust and rectify sensor data, enhancing the quality of fused data from multiple sensors [15].

This research focuses on enhancing data quality by filtering out irrelevant sensor data, which is crucial for AV navigation and decision-making. Ensuring only essential data is utilized optimizes the vehicle's perception capabilities. Preliminary experiments have involved the development of initial data filtration algorithms, which were tested on ROSbag files

containing various sensor data types. This process includes setting criteria for data relevance based on movement and environmental factors. Expected outcomes include improved data quality and reduced computational load, leading to more efficient and reliable AV systems. These insights underscore the importance data filtration and provide a foundation for developing robust ROSbag filtration nodes, thereby advancing autonomous vehicle technology.

III. Problem Definition

As autonomous vehicles (AVs) operate in diverse environments, they continuously gather vast amounts of sensor data. This data, stored in ROS (Robot Operating System) bag files, includes inputs from various sensors such as odometry, LIDAR, and cameras. A ROS bag file is a versatile file format in ROS that records and stores message data. These files act as a comprehensive log of the data exchanged over sensor topics during the operation of an AV. Essentially, a ROS bag file captures and preserves the real-time data stream from all sensors and system nodes, making it an invaluable resource for analysis, debugging, and simulation. In the context of the AV data analyzed: Odometry data include the vehicle's position and movement over time, which is critical for understanding its trajectory and speed. LiDAR (Light Detection and Ranging) sensors generate 3D maps of their environment by emitting laser pulses and measuring the time it takes for the reflections to return. This data is essential for obstacle detection and environment mapping. Another example are the cameras on the AV, which capture video streams that provide a visual context to the vehicle's surroundings, assisting in object recognition. Consider an AV navigating a rural area: it captures data from every tree, fence, and livestock it encounters, as well as periods when the vehicle is stationary. This constant influx of data includes both useful and irrelevant information, which complicates data analysis and can degrade the performance of AV systems. The challenge itself lies in effectively filtering out the noise and irrelevant information while retaining high-quality, relevant data. Certain inputs such as odometry, are crucial for the AV to understand and navigate its surroundings. However, they often include redundant and irrelevant information, particularly when the vehicle is stationary or when the sensors capture static objects that do not contribute to meaningful analysis.

IV. Algorithms

Demonstrated in Fig. 1, the process for constructing the RBF tool begins with the raw sensor data includes inputs from odometry and IMU sensors. This sensor data undergoes SNR calculations to filter out noise and enhance data quality. Simultaneously, the data is processed through a Kalman Filter for state estimation. Both the refined odometry and IMU data are then integrated into the RBF Node graphical user interface (GUI), where users can visualize and further refine the information. The final output is a ROS Filtered Data File, containing high-quality, relevant sensor data.

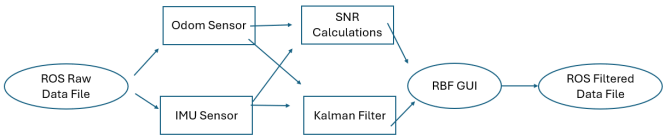

Fig. 1: Data refining process for AV sensor data.

A. RBF Tool Implementations

The core components of the RBF tool include the Kalman Filter for state estimation and SNR calculations for distinguishing between high-quality and low-quality data. To ensure the integrity and reliability of sensor data, it is essential to effectively filter out irrelevant information by developing a robust tool to organize and filter ROS bag files that aim to enhance data quality. This involves several key technical challenges: implementing Signal-to-Noise ratio calculations, and defining the Kalman filter algorithm into the RBF tool. The RBF tool's implementation involves several key classes and functions, as outlined in Fig. 2 of the UML diagram. The main function serves as the entry point for the RBF tool. It initializes the necessary parameters and orchestrates the data filtering process. Key attributes include the ROS version, duration's, and topics, which define the ROS environment and the specific data to be filtered.

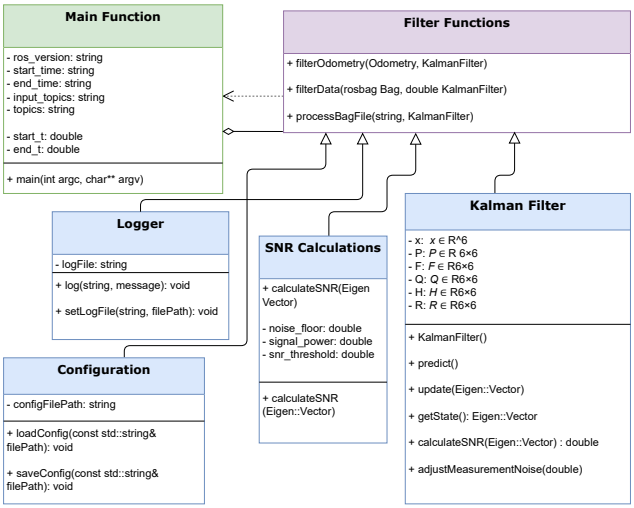

Fig. 2: UML diagram of the RBF tool.

The main function invokes filtering odometry data using the Kalman Filter, applying the Kalman Filter to the specified ROS bag file, and processes the ROS bag file with the necessary filters towards the data. The Kalman Filter is a critical component for enhancing data accuracy. This begins with initializing the filter, predicting the next state, updating the state with new measurements, returning the current state estimate, calculating the SNR for current state, and finally adjusting the measurement of extraneous data based on the

SNR. The SNR calculations class is responsible for determining the quality of the data based on signal strength, with the noise floor representing the baseline of extraneous data, the signal power, and the threshold for acceptable SNR to compute for a given vector of sensor data. The Logger class manages logging functionality, capturing key events and data during the filtration process. The Configuration class handles the loading and saving of configuration settings for the RBF tool. The implementation of these components ensures that the RBF tool can effectively enhance the quality of sensor data by filtering out irrelevant information. The tool's design, as illustrated by the UML diagram, highlights the interaction between various modules, providing a clear structure for future enhancements and scalability. The RBF tool's architecture, incorporating the Kalman Filter and SNR calculations, provides a robust framework for data filtration.

B. Signal-to-Noise Ratio (SNR)

To ensure the quality of the extracted data, the project uses the signal-to-noise ratio (SNR) as a metric. The SNR is defined as:

$$\text{SNR} = \frac{\mu_{\text{signal}}}{\sigma_{\text{noise}}} \tag{2}$$

where μ_{signal} is the mean of the signal and σ_{noise} is the standard deviation of the extraneous data. High SNR values indicate high-quality data, which is crucial for accurate AV operations. By calculating the SNR, we can effectively distinguish between useful sensor data and background noise, ensuring only high-quality data is processed.

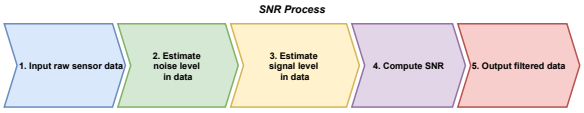

Fig. 3: Flow chart of the SNR calculation.

The RBF tool employs Signal-to-Noise Ratio (SNR) calculations to further refine the data. High SNR values indicate high-quality data, while low SNR values suggest that the data may be dominated by extraneous information, often referred to as noise. Demonstrated in Fig. 3, the RBF tool calculates the SNR for each data point and filters out those with low SNR values. This process is essential for removing low-quality data that could otherwise degrade the performance of autonomous vehicle systems. By focusing on high-quality data, the tool enhances the reliability and robustness of the processed information.

C. Kalman Filter for State Estimation

The project employs the Kalman Filter for state estimation, which is a recursive algorithm used to estimate the state of a dynamic system from noisy measurements. The Kalman Filter equations are as follows:

1) Prediction Step:

$$\hat{x}_{k|k-1} = \mathbf{F}_k \hat{x}_{k-1|k-1} + \mathbf{B}_k u_{k-1} \tag{3}$$

where $\hat{x}_{k|k-1}$ represents the predicted state estimate at time k given the state at time $k-1$. $\mathbf{F}_k$ is the state transition model, $\mathbf{B}_k$ is the control-input model, and u_{k-1} is the control vector. This equation predicts the vehicle's next state based on its current state and control inputs.

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_k \tag{4}$$

where $\mathbf{P}_{k|k-1}$ is the predicted estimate covariance, representing the uncertainty in the predicted state. $\mathbf{Q}_k$ is the process noise covariance.

2) Update Step:

$$\tilde{y}_k = z_k - \mathbf{H}_k \hat{x}_{k|k-1} \tag{5}$$

where $\tilde{y}_k$ is the innovation or measurement pre-fit residual, representing the difference between the actual measurement z_k and the predicted measurement $\mathbf{H}_k \hat{x}_{k|k-1}$. This step quantifies the discrepancy between the predicted state and the actual measurement.

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k \tag{6}$$

where $\mathbf{S}_k$ is the innovation covariance, which quantifies the uncertainty in the innovation. $\mathbf{R}_k$ is the measurement noise covariance.

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T \mathbf{S}_k^{-1} \tag{7}$$

where $\mathbf{K}_k$ is the Kalman gain, which determines the weight given to the innovation. This step adjusts the state estimate based on the measurement uncertainty and the predicted uncertainty.

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + \mathbf{K}_k \tilde{y}_k \tag{8}$$

where $\hat{x}_{k|k}$ is the updated state estimate, computed by adjusting the predicted state estimate with the weighted innovation.

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1} \tag{9}$$

where $\mathbf{P}_{k|k}$ is the updated estimate covariance, reflecting the uncertainty in the updated state estimate.

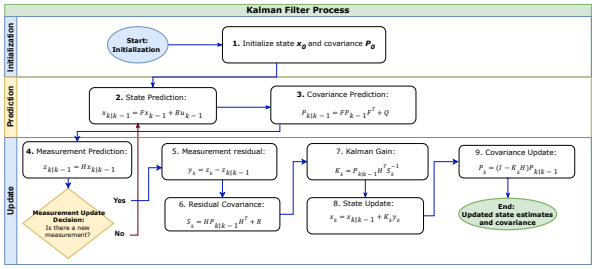

Fig. 4: Flow chart of the Kalman filter process.

The Kalman Filter plays a crucial role in the RBF tool by improving data accuracy through predictive modeling and state estimation. As shown in Fig. 4, it works by predicting the state of the vehicle based on previous measurements and then updating this prediction with new measurements. This recursive algorithm is particularly effective when the vehicle is stationary, such as at a red light or stop sign, where sensor data can include a significant amount of irrelevant information due to lack of movement. By accurately predicting and updating vehicle motion through state estimate, the Kalman Filter ensures that only relevant data is retained during the filtration process. During the prediction step, the filter estimates the current state of the vehicle and the uncertainty associated with this estimate. In the update step, the filter incorporates new measurements to refine the state estimate of the vehicles motion. This continuous process of prediction and correction results in a highly accurate estimates, which are critical for effective data filtration.

V. Experiments

The experiments conducted to evaluate the effectiveness of the ROSbag Filtration (RBF) tool involved multiple stages, each aimed at validating and demonstrating the tool's capabilities. These stages included the implementation and validation of the Kalman Filter and Signal-to-Noise (SNR) calculations, the development and assessment of the features of the Graphical User Interface (GUI), performance comparisons before and after filtration using Matplotlib for visualizations, and comprehensive testing through various test cases.

A. RBF Graphical User Interface (GUI)

To facilitate user interaction with the ROSbag Filtration (RBF) tool, a Graphical User Interface (GUI) was developed using the QT framework. The GUI provides an intuitive platform that enables users to efficiently manage and filter data within ROS bag files, enhancing the overall usability and functionality of the RBF tool.

The primary interface of the GUI includes several key features designed to streamline the data filtration process Fig. 5. The GUI provides a straightforward mechanism for selecting ROS bag files or folders containing multiple bag files. Users can navigate through their file system using the built-in directory structure, allowing easy access and organization of the files to be processed. Users can specify the start and end times for data extraction, allowing for precise control over the duration of interest. This functionality is particularly useful when dealing with large datasets, as it enables the filtration of data relevant to specific events or time frames. The GUI allows users to select specific sensor topics such as odometer, IMU data, and front camera inputs. This customization enables users to focus on the data most pertinent to their analysis, excluding irrelevant information and optimizing the filtration process. The central control panel of the GUI features essential buttons for data processing. The 'Filter' button initiates the data filtration based on the user-defined parameters. Additional control buttons such as 'OK' and 'Cancel' provide further management options, allowing users to confirm their settings or reset the parameters if needed. Once the filtration process is initiated, the GUI provides real-time feedback and visualization of the progress. This feature helps users monitor the status of the data processing and ensures that they are informed of the ongoing operations. The results of the filtration are displayed within the GUI, allowing for immediate review and verification of the refined data. The development of the GUI using QT involved several critical steps to ensure a robust and user-friendly interface. The design focused on creating a clean and intuitive layout, minimizing complexity while maximizing functionality. The integration of file handling capabilities and real-time processing feedback enhances the overall user experience, making the RBF tool accessible to users with varying levels of technical expertise. Additionally, to further simplify access to the GUI, a shortcut command 'rbfplay' was implemented. This command can be executed in the command line to open the GUI, making it all the more user-friendly. By providing this shortcut, the tool ensures that users can quickly and easily access the GUI without needing to navigate through directories or execute complex commands. This feature enhances the tool's accessibility, catering to both novice and experienced users. The GUI was implemented with a modular approach, allowing for future enhancements and scalability. This design choice ensures that additional feature or modifications can be easily incorporated as the tool evolves.

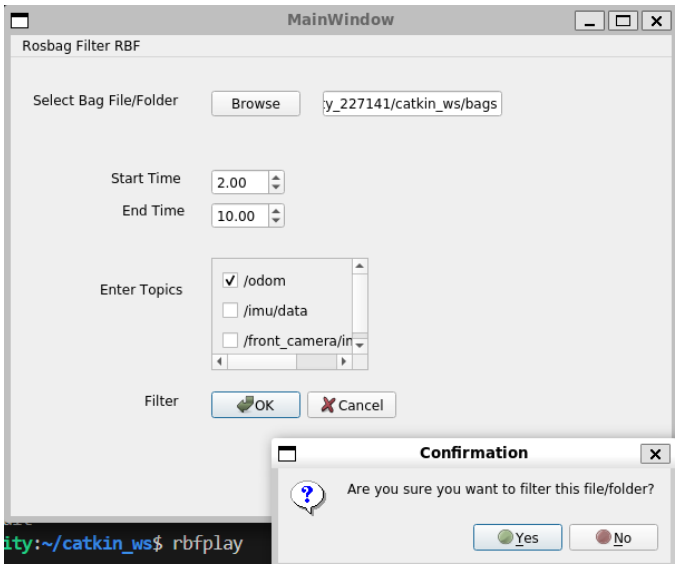

Fig. 5: Prototype of the RBF GUI . The GUI allows users to select a ROS bag file or folder, set start and end times for data extraction, and specify the sensor topics to filter.

Overall, the GUI plays a crucial role in making the RBF tool more accessible and efficient, enabling users to manage and filter ROS bag data with greater ease and precision.

B. Performance Before and After Filtration

To evaluate the performance of the RBF tool, a comprehensive analysis was conducted comparing sensor data before and

after the filtration process. This comparison aimed to demonstrate the tool's ability to enhance data quality by reducing excess data and filtering out irrelevant information. The graphs displayed in Fig. 7 show the RBF tool successfully removed extraneous data points, leading to a cleaner and more reliable dataset. This improvement is crucial for autonomous vehicle (AV) systems, where high-quality sensor data is essential for accurate perception and decision-making. First, raw sensor data from various sources were collected from ROS bag files. These files contained a mix of useful and extraneous information, providing a suitable test-bed for assessing the RBF tool's filtration capabilities. The collected data was then processed using the RBF tool, which applied the Kalman Filter and Signal-to-Noise Ratio (SNR) calculations to refine the data. The Kalman Filter improved state estimation, particularly when the vehicle was stationary, while the SNR calculations helped distinguish high-quality data from background noise. To visually represent the improvements in data quality, the raw and filtered data were graphed using Matplotlib. This visualization tool allowed for a clear and detailed comparison of the data before and after filtration, highlighting the effectiveness of the RBF tool.

Fig. 6: Images from experiments with our AV.

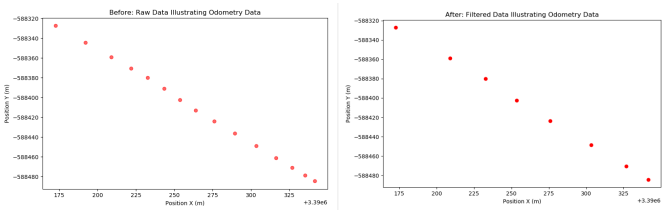

Fig. 7: Before and after filtration process.

C. Test Cases and Validation

To ensure the robustness and reliability of the RBF tool, a comprehensive set of test cases were conducted. These tests covered basic functionality, boundary and edge cases, performance, error handling, and integration with other ROS nodes, displayed in Fig. 8. First, the basic functionality of the RBF tool was tested by verifying its ability to filter data by time range, specific topics, and a combination of both. These tests ensured that the tool could correctly filter data within user-defined parameters. Next, boundary and edge case tests were performed to validate the tool's behavior under various challenging scenarios. These included testing with an empty ROS bag file, large ROS bag files, non-existent topics, and overlapping time ranges. The tool handled the cases gracefully, successfully maintaining stability and providing appropriate responses. Performance tests were conducted to ensure that the tool provides clear and informative error messages for invalid inputs, such as invalid time ranges or corrupted ROS bag files. The tool's robust error handling mechanisms contributed to a user-friendly experience. Furthermore, integration tests were performed to verify the seamless functionality of the RBF tool with other nodes in the ROS ecosystem. The end-to-end functionality was validated using real-world datasets, confirming the tool's capability to process and refine the validated data, and the tool's behavior with unsupported topics was examined to ensure a comprehensive coverage. All tests were executed successfully, demonstrating the robustness and reliability of the RBF tool. The complete documentation and source code are available on the RBF GitHub repository for further collaboration and development [16].

Test Cases	Basic Functionality Tests	Boundary Edge Case Tests	Performance Tests	Error Handling	Integration Tests
TC01	Verify RBF by time range ✓				
TC02	Verify RBF by specified topics ✓				
TC03	Verify RBF with a combination of time range and topics ✓				
TC04		Verify behavior with an empty Rosbag File ✓			
TC05		Verify behavior with a very large Rosbag file ✓			
TC06		Verify behavior with non-existent topics ✗			
TC07		Verify behavior with overlapping time ranges ✗			
TC08			Measure performance with increasing sizes of Rosbag file ✓		
TC09			Measure performance with increasing numbers of topics ✓		
TC10				Verify error messages for invalid input (e.g., invalid time range) ✗	
TC11				Verify error handling for corrupted Rosbag file ✗	
TC12					Verify integration with other nodes in the ROS ecosystem ✓

Pass: File Created
Fail: File Not Created

Fig. 8: Test cases conducted to ensure RBF tool functionality.

VI. Conclusion and Future Work

The development and implementation of the ROSbag Filtration (RBF) tool have demonstrated significant improvements in the quality and reliability of sensor data for autonomous vehicle systems. By integrating the Kalman Filter and Signal-to-Noise Ratio (SNR) calculations, the RBF tool effectively filters out noise and irrelevant information, providing a cleaner dataset for further analysis and decision-making.

A. Automation through PyBind11

A key enhancement to the RBF tool involves the automation of data processing tasks using PyBind11. PyBind facilitates seamless interoperability between C++ and Python. By leveraging this framework, automating the data filtration process would significantly enhance the performance and efficiency of the RBF tool. The integration of PyBind11 allows for several key benefits. By automating data processing tasks, computational overhead is reduced and the filtration process is accelerated. This leads to faster data processing times, enabling the RBF tool to handle larger datasets more efficiently.

PyBind11 specifically enables integration between C++ and Python, allowing us to leverage the performance benefits of C++ while maintaining the ease of use and flexibility of Python. This ensures that the tool could utilize advanced algorithms and libraries from both languages, enhancing its overall functionality. Ultimately, automation through PyBind simplifies the data processing workflow, making the RBF tool more user-friendly. Users could benefit from automated data filtering without needing to manually configure and execute complex commands, thereby improving the overall user experience.

B. Future Enhancements to the User Interface

Future developments plans for the RBF tool include several enhancements to the graphical user interface to further improve usability and functionality. One of the key features planned is the ability to preview the filtration results before confirming the creation of a new file, allowing users to see a visual representation of the data before and after filtration. This would serve to further enable users to verify the effectiveness of the selected filters and make any necessary adjustments before committing to the final data processing. By providing a preview, users could ensure that the filtration parameters are correctly configured, thereby reducing the risk of data loss or incorrect filtering. Additionally, enhancing the UI with more interactive controls could provide users with greater flexibility when selecting and adjusting filtration parameters. For example, sliders and input fields could be used to set the duration and sensor topics more precisely. Interactive graphs and charts can also be incorporated to provide real-time feedback on the filtration process, allowing users to make informed decisions about their data. These future enhancements will focus on improving the navigation and accessibility of the RBF GUI, including better file management options, such as the ability to organize and sort ROS bag files, as well as streamlined access to frequently used features. Enhancements in navigation would ensure that users can quickly locate and process the data they need. Additionally, allowing users to customize the GUI according to their preferences would enhance the overall user experience. Customized dashboards, themes, and layouts can be introduced to make the RBF tool more adaptable to different user requirements. This accommodation would cater to a wider range of users, ensuring that the tool meets their specific needs. These improvements for the RBF tool through automation and GUI enhancements would significantly elevate the overall functionality and usability. The integration of PyBind11 could optimize performance, making the data filtration process more efficient. Future GUI enhancements could provide users with a more interactive experience, allowing them to better manage and filter their ROS bag data. Overall, the RBF tool has shown great promise in improving data quality for autonomous vehicle systems. By continuing to enhance the tool's performance and usability, we aim to provide researchers and developers with a powerful and efficient tool for managing sensor data, ultimately contributing to the advancement of autonomous vehicle technology.

VII. Acknowledgment

This research is developed as part of a larger effort to improve data quality for autonomous vehicle research, in collaboration with Aaron Kingery and Dez Song's team. Thank you to Aaron Kingery for his insights on the algorithms and structure of the entirety of the RBF tool, Agah Duzenli for his contributions to the functionality of the tool, Dez Song for his expertise in software implementation, and Jason O'Kane for his guidance on the overall project framework. Their thoughtful conversations and support have been invaluable to the development of this software.

References

[1] U.S. Department of Transportation, "Argonne Lab's Perception and Connectivity Activity (ALPaCA) Raw Sensor and Vehicle Data," 2023. [Online]. Available: https://livewire.energy.gov.

[2] U.S. Department of Transportation, "Raw data Batch I: Garfield Ridge," 2023. Available: https://livewire.energy.gov/ds/alpaca/garfield-ridge.

[3] P. H. Chan et al., "LIDAR De-Snow Score (DSS): Combining Quality and Perception Metrics for Optimized Data Filtering," 2022. [Online]. Available: https://www.techrxiv.org/doi/pdf/10.36227/techrxiv.171259681.14859329.

[4] Z. Chen, M. Albonico, and I. Malavolta, "Automatic Extraction of Time-windowed ROS Computation Graphs from ROS Bag Files," *arXiv preprint arXiv:2305.16405*, 2023. [Online]. Available: https://arxiv.org/pdf/2305.16405.

[5] Y. Wu et al., "Variable Dimension-based Method for Roadside LiDAR Background Filtering," *IEEE Transactions on Intelligent Transportation Systems*, 2021. [Online]. Available: https://ieeexplore.ieee.org/stampPDF/getPDF.jsp?tp=&arnumber=9605615.

[6] N. Polat, M. Uysal, and A. S. Toprak, "An investigation of DEM generation process based on LiDAR data filtering, decimation, and interpolation methods for an urban area," *Measurement*, 2015. [Online]. Available: https://doi.org/10.1016/j.measurement.2015.08.008.

[7] A. Kingery and D. Song, "Road Boundary Estimation Using Sparse Automotive Radar Inputs," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2023, Brisbane, Australia. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8461080.

[8] B. Li et al., "Lane Marking Verification for High Definition Map Maintenance Using Crowdsourced Images," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 2324–2331. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9340923.

[9] P. Corke, "Integrating ROS and MATLAB," *IEEE Robotics & Automation Magazine*, 2015, pp. 19–20. [Online]. Available: https://ieeexplore.ieee.org/stampPDF/getPDF.jsp?tp=&arnumber=7124623.

[10] J. Santos, J. Lourenço, and F. Martins, "A Framework for Quality Assessment of ROS Repositories," 2016.

[11] Y. Li et al., "QDTree: Quasi-Density-Tree Accelerates Free Space Detection with MMW Radar Point Cloud," 2024. [Online]. Available: file:///C:/Users/kaitl/Downloads/ATDE-50-ATDE240043%20(2).pdf.

[12] H. Zheng et al., "Long-Range Imaging LiDAR with Multiple Denoising Technologies," *Applied Sciences*, vol. 14, no. 8, p. 3414, 2024. [Online]. Available: https://doi.org/10.3390/app14083414.

[13] K. Qu and W. Zhuang, "Scalable and dynamic cooperative perception: A Data/model co-driven framework," *IEEE Transactions on Intelligent Transportation Systems*, 2023. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10400178.

[14] D. Na et al., "Depth Information Fusion using Radar-LiDAR-Camera Experimental Setup for ADAS Applications," *IEEE International Conference on Communication Systems and Network Technologies (CSNT)*, 2024, pp. 59–63. [Online]. Available: https://ieeexplore.ieee.org/stampPDF/getPDF.jsp?tp=&arnumber=10546006.

[15] Y. Li et al., "DIME-Net: Dynamic Intrinsic Matrix Estimation Network for Mobile Cameras," *arXiv preprint arXiv:2303.11307*, 2023. [Online]. Available: https://arxiv.org/pdf/2303.11307.

[16] K. Asch, "ROSbag Filter Project," GitHub Repository, 2024. [Online]. Available: https://github.com/kaitlin-a/rosbag-filter-project/tree/master