Radar odometry using mmWave technology for SLAM applications.

Leveraging mmWave Radar Sensor Technology for odometry estimation.

Luis Fernando Rodriguez Gutierrez
Fachhochschule Dortmund
M.Eng. Embedded Systems Engineering
luis.rodriguez001@stud.fh-dortmund.de

***Abstract*—Radar technologies have recently emerged as a promising alternative to traditional odometry methods, which often rely on cameras, LiDAR, wheel encoders, inertial sensors, or GPS. Unlike these modalities, millimeter-wave (mmWave) radar offers robustness under adverse conditions such as fog, rain, or low light, while directly providing both range and Doppler velocity measurements. Previous work has demonstrated that radar can support odometry through global iterative closest point (ICP) alignment of point clouds. Building on this foundation, this work focuses on ego-motion estimation using a dual mmWave radar system complemented by an inertial sensor. The dual-radar configuration is designed to increase spatial coverage and reduce ambiguity in radar detections, addressing a key limitation of single-radar odometry approaches.**

Experimental evaluations were conducted in both enclosed environments (laboratory) and open environments (university parking lot). Results show that the proposed framework provides consistent and reliable ego-motion estimation, highlighting the potential of mmWave radar as a cost-efficient substitute or complement to conventional odometry sources in autonomous navigation.

***Index Terms*—Radar Odometry, mmWave Radar, Dual-Radar Configuration, Clustering, ICP, Doppler Velocity, Submap Aggregation, Ego-Motion Estimation**

I. Introduction

Accurate and reliable ego-motion estimation is a fundamental requirement for mobile robotic systems and autonomous vehicle solutions. It is the basis for localization, mapping, and navigation, and errors in this stage directly affect the overall performance of autonomous systems.

Traditionally, odometry has been estimated using a combination of wheel encoders, inertial measurement units (IMUs), and GPS. In more recent years, cameras and LiDAR have been widely used because they provide dense information about the environment, enabling precise feature extraction and recognition. However, these vision-based and LiDAR-based methods have significant drawbacks. Cameras are highly sensitive to illumination changes. LiDAR systems are costly, and their performance can degrade in adverse weather conditions such as fog, rain, or snow. Both methods also require high computational and memory resources. These limitations create the need for complementary sensing solutions that remain reliable under real-world conditions.

Millimeter-wave (mmWave) radar has emerged as a strong candidate to address these issues. Radar is compact, cost-efficient, and inherently robust to poor lighting and weather. A key advantage of radar is its ability to directly measure Doppler velocity. Unlike cameras or LiDAR, which require frame-to-frame comparisons to estimate motion, radar provides direct measurements of radial velocity. This not only indicates how fast the vehicle is approaching or moving away from an object, but also makes it possible to separate static structures from moving objects and to detect relative motion trends. These Doppler-based measurements provide additional constraints for ego-motion estimation, making radar a unique and valuable sensing modality.

Despite these advantages, radar data also presents challenges. The resulting point clouds are sparse and noisy, and they often include significant amounts of clutter. Radar also has lower angular resolution compared to LiDAR or cameras. These limitations make it difficult to directly apply traditional scan-matching techniques, which are usually designed for dense LiDAR point clouds. Previous work has shown that radar-only odometry and multimodal fusion can improve robustness, but challenges remain when dealing with sparsity, clutter, and the stability of scan registration.

This work investigates the use of mmWave radar sensors mounted on a Ninebot Go-Kart test platform. The system also integrates an IMU and an embedded processing unit. A visual overview of the setup, including sensor placement, is shown in Figure 1. This figure highlights the main components without going into detailed descriptions, which are provided later in the hardware description section.

Fig. 1: Ninebot test-vehicle system.

The contributions of this work can be summarized as follows:

1) A radar ego-motion pipeline using mmWave sensors and an IMU for rotation, minimizing hardware cost and system complexity.
2) Integration of Doppler velocity and RANSAC filtering to improve the separation of static and dynamic objects.
3) Submap aggregation to mitigate point cloud sparsity and improve alignment stability.
4) Object tracking via clusters to identify and filter dynamic objects from the ego-motion estimation.
5) Experimental validation using real-world data collected from a vehicle-mounted mmWave radar sensor.

II. System Design and Methodology

A. Research Design

The main objective of this work is to develop a radar-based odometry system that estimates vehicle ego-motion using front-mounted mmWave radar sensors, combined with an IMU for rotation compensation. The motivation is to investigate radar as a cost-effective and robust alternative to vision- or LiDAR-based odometry, especially in conditions where these methods are prone to failure. This builds on prior evidence that radar can provide instantaneous ego-motion estimation through Doppler velocity cues [13].

The system processes radar point cloud data, enriched with range, angle, and Doppler velocity, to extract ego-velocity and reconstruct the vehicle's trajectory. These outputs form a foundation for SLAM applications.

As shown in Figure 2, two mmWave radar sensors were mounted at the front of the vehicle with overlapping fields of view to improve coverage and reduce ambiguity. This configuration served as the baseline validation scenario, chosen as a simple and controlled setup to verify the dual-radar concept before moving to more complex outdoor environments.

The dual-radar arrangement increased point density and stability compared to single-radar approaches, aligning with multimodal strategies for robust state estimation [14], [15]. By fusing Doppler-derived radial velocities with rotational information from the IMU, the system was designed to remain resilient in conditions where LiDAR- or camera-based odometry would degrade. Each radar stream was processed independently and then merged into a combined representation before entering the odometry pipeline.

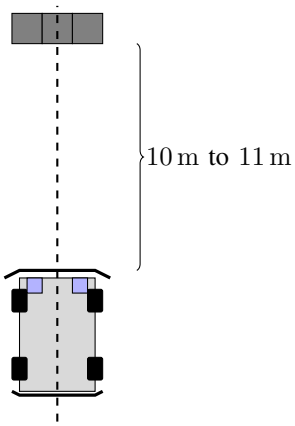


Fig. 2: Test scenario with dual front-mounted mmWave radar sensors. This setup served as the baseline validation environment before moving to more complex scenarios.

1) Sub-Tasks: The dual-radar objective implied several practical sub-tasks:

- Designing a modular pipeline to acquire and decode synchronized radar data from both sensors.
- Investigating sensor configurations to balance field of view, chirp bandwidth, update rate, and detection density.
- Developing mechanical mounts and selecting optimal sensor placement to ensure stability and maximize coverage.
- Applying RANSAC filtering on Doppler velocities to reject dynamic points and outliers.
- Implementing clustering methods to structure radar detections and isolate relevant features.
- Leveraging additional sensor information (e.g., SNR, RCS, range validity) to optimize point cloud reliability.
- Integrating Doppler velocity into the odometry estimation process.
- Employing submap aggregation to mitigate sparsity and improve stability.
- Performing ICP alignment between submaps aggregated from both sensors.
- Evaluating the influence of the dual-sensor arrangement on odometry accuracy and robustness.
- Validating the complete system on real-world driving scenarios.

Taken together, these sub-tasks define the modular pipeline through which radar and IMU data are processed into ego-motion estimates. Each block in the pipeline corresponds to one or more of the identified tasks, from raw data acquisition and transformation, to filtering, clustering, and alignment. This modularity ensures that individual stages can be validated and tuned independently, while still contributing to the overall odometry framework. The complete processing chain is summarized in Figure 3, which illustrates how the dual-radar inputs are merged with IMU measurements and progressively refined into robust ego-motion estimates.

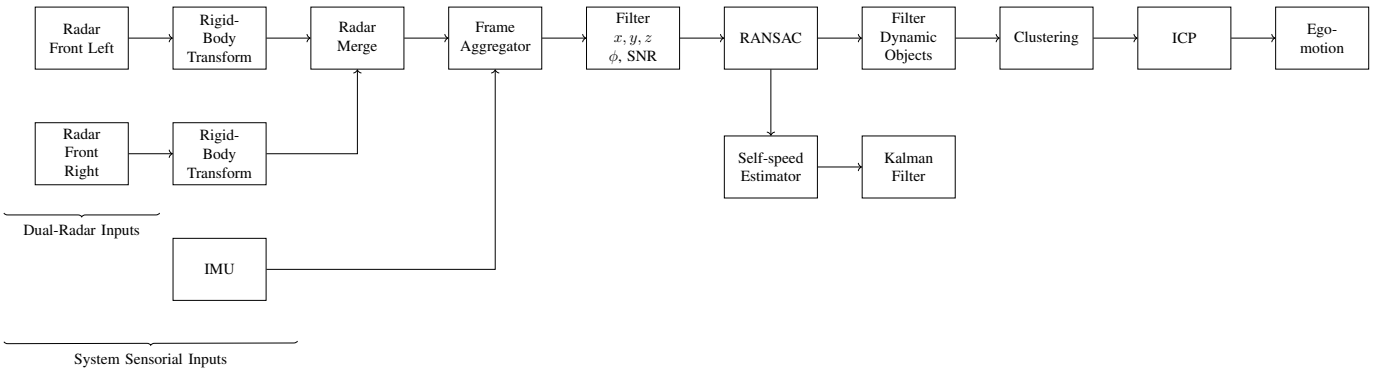


Fig. 3: Dual-radar pipeline showing how radar and IMU inputs are merged into the frame aggregator before subsequent processing.

The following sections detail the hardware setup, pipeline methodology, and experimental validation.

B. System Hardware

The hardware setup combines mmWave radars, an inertial sensor, and auxiliary components into a compact test platform. In this section, we describe the role of each element, how they were mechanically integrated, and how the radar configuration was selected for odometry tasks.

1) System Sensors:
The sensors that are implemented and used in this project are the following:

a) mmWave Radar (IWR6843AOP):
The IWR6843AOPEVM development board from Texas Instruments features the IWR6843AOP, a high performance 4D mmWave FMCW sensor with Antenna On Package (AOP) design. Although IWR6843AOP is intended for industrial applications and its complementary chip, AWR6843AOP, for automotive applications, IWR6843AOP was used in this project because it is available in the form of this development board and the two chips are identical in terms of their functionalities, only differing in compliance with automotive industry [2]. Its small physical size, due to its AOP design, makes it an optimal choice for the desired mounting position, the go-kart's steering column.

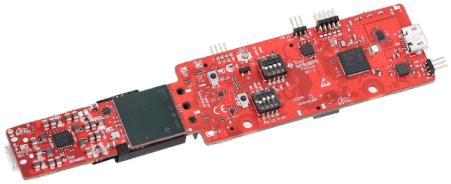


Fig. 4: IWR6843AOP sensor
Source: Texas Instruments, available at https://www.ti.com/ds_dgm/images/fbd_swrs219f.gif

The IWR6843AOP radar sensor operates within the frequency range of 60 GHz to 64 GHz and integrates 4 receive (RX) and 3 transmit (TX) antennas, radio frequency (RF) front-end stages, analog signal processing, and digital signal processing (DSP). It offers a wide range of communication interfaces including SPI, I2C, CAN-FD, UART and LVDS for raw data access and an Arm Cortex-R4F microcontroller for user-applications as shown in the Figure 5 [3].

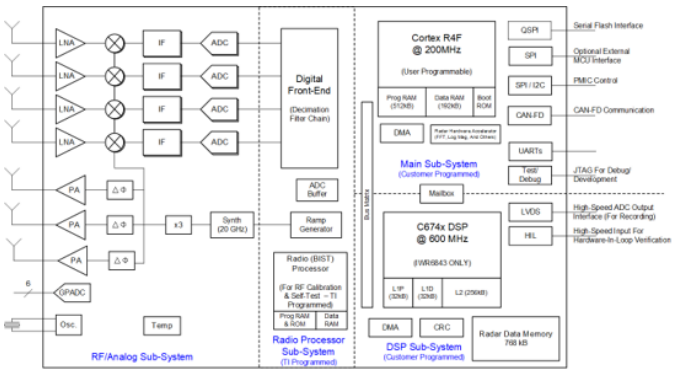


Fig. 5: IWR6843AOP internal block diagram.
Source: Texas Instruments, available at https://www.ti.com/ds_dgm/images/fbd_swrs219f.gif

As some operating parameters influence each other, their selection must be done carefully while observing the influence of the trade-offs involved. This could be referred to as "sensor tuning" and is a critical step because it directly impacts the system's accuracy and performance.

The following operating parameters can be tuned:

- Frame rate
- Range resolution
- Maximum unambigious range
- Maximum radial velocity
- Radial velocity resolution

Tuning these operating parameters introduces trade-offs by influencing each other in the following ways: The resulting overall accuracy of the velocity and distance measurements is again dependent on these operating parameters:

Tuning Parameter	Effect on Performance	Related HW Block	Trade-Off
Frame Rate	Higher FPS = faster updates but more processing load	C674x DSP, Radar Data Memory	Higher FPS reduces maximum range
Range Resolution	Higher resolution = better object separation	ADC, 1D FFT (Range FFT)	Higher resolution reduces max range
Maximum Range	Determines farthest detectable object	RF Front-End, PA, LNA, ADC	Higher range lowers resolution
Radial Velocity Resolution	Improves speed accuracy	DSP, 2D FFT (Doppler FFT)	Higher resolution requires more chirps
Maximum Radial Velocity	Detects fast-moving objects	Chirp rate, TX Antennas, 2D FFT	Higher max velocity reduces resolution

TABLE I: Radar System Tuning Parameters and Trade-offs

- **Radial velocity accuracy:** A fine balance between velocity resolution and frame rate must be maintained to ensure precise Doppler shift measurements. Lower resolution results in rounded velocity values, while an excessively high frame rate may introduce computational bottlenecks.
- **Distance accuracy:** Optimizing range resolution and maximum range ensures that detected objects are positioned accurately within the environment. Increasing range often sacrifices resolution, leading to potential inaccuracies in close-range detections.
- **Signal processing considerations:** The FFT calculation parameters directly affect both range and Doppler calculations, influencing the ability to distinguish between objects and detect small velocity variations.

b) Inertial Measurement Unit (IMU):

The MTi-G-710 from Xsens is a high-performance GNSS/INS module that integrates a 3D accelerometer, 3D gyroscope, 3D magnetometer, and barometer with an onboard GNSS receiver. Although designed as a complete navigation unit capable of delivering position and velocity estimates, in this project it was used specifically for its orientation outputs, provided in the form of quaternions, Euler angles, or rotation matrices [19]. Its compact form factor and proven sensor-fusion engine made it a suitable choice for integration into the test platform, complementing the radar subsystem by supplying real-time orientation references.

For the present setup, the device was configured to output quaternions, as this representation avoids singularities inherent in Euler angles and provides numerical stability for continuous tracking. These quaternions were used to track the vehicle attitude and to align radar point clouds within a consistent vehicle-centric coordinate frame. By relying on the MTi-G-710 internal fusion engine rather than implementing custom filtering, the system leveraged factory-calibrated algorithms for bias correction and drift reduction, minimizing processing overhead on the host platform.

To interface with the device and parse its MTData2 protocol, an open-source implementation from Scottapotamas was adopted [20]. This repository enabled robust packet parsing and quaternion extraction in Python, ensuring seamless integration of the IMU data stream into the radar processing pipeline.

c) Webcam:

A Logitech C270 HD webcam was included in the experimental platform for the sole purpose of video recording. Unlike the radar and IMU, the camera was not integrated into the sensing or processing pipeline and did not contribute any data to the odometry estimation system. Its role was limited to providing visual documentation of the test environment, allowing experiments to be reviewed and contextualized with synchronized video footage.

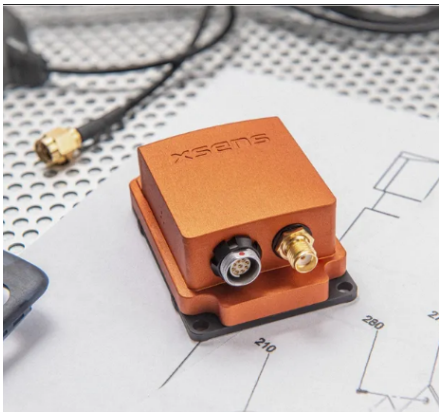


Fig. 6: MTi-G710 high-performance inertial navigation system (INS).
Source: Movella, available at https://www.movella.com/sensor-modules/xsens-mti-7-gnss-ins

2) Mechanical Integration:

The mechanical design and integration of the sensing suite were crucial to ensure stable operation, accurate data alignment, and reproducibility across experiments. This subsection is divided into three parts:

- 3D printed mounts for flexible radar positioning.
- 3D dedicated IMU mounting plate.
- Geometric placement of radars and corresponding coordinate transformations.

a) Radar Mounting Design:

A standard mounting bracket is provided by the manufacturer for the IWR6843AOP sensor, as shown in Figure 7a. While suitable for simple setups, this bracket only allows a fixed mounting position without the possibility of adjusting yaw or pitch. Such limitations restrict the ability to optimize the radar field of view for different vehicle placements and test scenarios.

To overcome this, a custom 3D-printed bracket was developed, enabling adjustable tilt and offering greater flexibility in sensor alignment (Figures 7 and 8). This design provides precise control over the radar orientation, which is essential for minimizing ground reflections, extending horizontal coverage, and ensuring that both radars could be consistently aligned with the vehicle's coordinate frame.

The bracket consists of three main components:

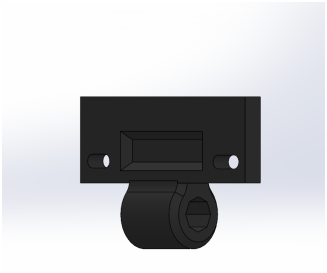
- A stable **mounting base** securely attached to the vehicle platform.
- A **sensor frame** that holds the IWR6843AOP in position.
- An adjustable **tightening screw** that locks the tilt angle once aligned.

The mounts were fabricated using PLA, chosen for its ease of printing and adequate stiffness for field use. They were

fixed to the vehicle platform with hot glue, a lightweight yet sufficiently rigid method to ensure stability during operation. This approach minimized vibration-induced artifacts and guaranteed repeatable positioning across experiments.

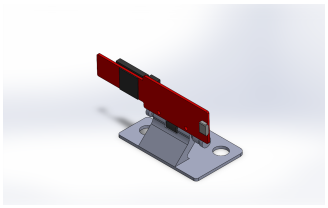
By introducing this adjustable design, the system achieved both mechanical robustness and alignment flexibility, enabling more accurate calibration of the dual-radar setup and improving consistency in downstream data fusion.

(a) Manufacturer-provided mount for the IWR6843AOP sensor. This design fixes the sensor in place without tilt adjustment, limiting flexibility. *Source: amicus engineering, available at amicus.com.sg*

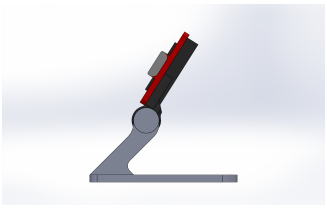


(b) Custom 3D-printed mount designed. It enables controlled tilt adjustments, ensuring repeatable alignment and optimized field of view.

Fig. 7: Comparison of sensor mounting options for the IWR6843AOP radar: (a) fixed manufacturer mount and (b) adjustable 3D-printed design. The custom bracket provides alignment flexibility and stability, improving calibration for dual-radar operation.



(a) IWR6843AOP sensor mounted on the custom 3D-printed bracket in its default upright orientation. The mount provides a stable base while allowing tilt adjustments.



(b) Side view showing the tilt functionality of the custom bracket. This adjustability was critical for compensating the 15° upward tilt used in the dual-radar configuration.

Fig. 8: Implementation of the custom 3D-printed mounting solution for the IWR6843AOP radar. The design combines stability with controlled tilt adjustment, enabling repeatable calibration and improved alignment with the vehicle frame.

While the 3D model (Figures 7-8) illustrates the intended mounting geometry, Figure 9 shows an intermediate real-world implementation. This prototype installation provided a practical check of sensor tilt and positioning but was later replaced by the custom 3D-printed bracket, which ensured stability, repeatability, and precise control over yaw and pitch adjustments.

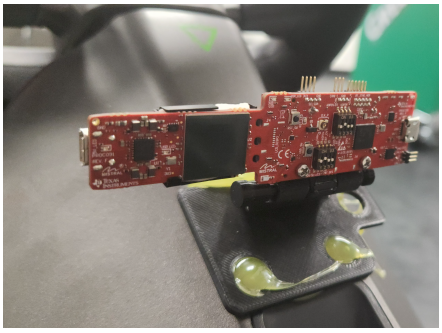


Fig. 9: Prototype installation of the IWR6843AOP radar sensor on the vehicle. This early mounting solution was used for visualization and comparison with the 3D design, but it was not adopted as the final configuration.

b) IMU Mounting Design:

The MTi-G710 IMU required a stable and horizontally aligned installation point to ensure that its quaternion orientation output could be directly mapped to the vehicle reference frame without introducing additional correction. To achieve this, a custom mounting solution was developed, illustrated in Figure 10.

The 3D design, shown in Figure 10, highlights the three main elements of the assembly:

1) The MTi-G710 sensor.
2) A flat mounting plate designed to secure the device.
3) The rear vehicle mounting spot, located near the spoiler.

This location was selected because it provided a rigid and vibration-resistant base, while also being naturally aligned with the horizontal plane of the vehicle. The design ensured that the IMU could be mounted flush, minimizing installation errors and avoiding unnecessary post-processing of orientation data.

The real-world implementation is shown in Figure 11, where the IMU was securely fixed to the vehicle. This placement offered both ease of integration and the stable reference needed for reliable sensor fusion with the radar system.

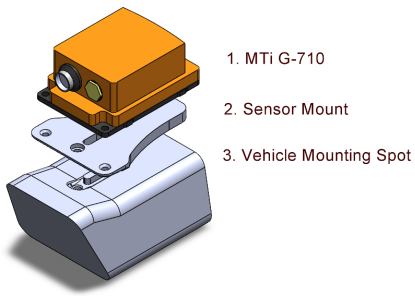


Fig. 10: 3D model of the MTi-G710 IMU installation. The design consists of (1) the IMU, (2) a custom flat sensor mount, and (3) the rear vehicle mounting spot selected for its stability and horizontal alignment.

Fig. 11: Real-world installation of the MTi-G710 at the rear vehicle mount point. This position near the spoiler ensured a rigid and level platform, reducing post-processing requirements for orientation correction.

3) Sensor Setup and Calibration:

An understanding of how the sensors operate is required before they can be correctly integrated into the system. This involves studying their configuration tools, calibration procedures, and data output formats. Only by understanding their internal operation and limitations can they be properly adapted for the precise requirements of this application.

Radar Setup and Calibration

a) Configuration Tools:

To configure and validate the radar sensors, two official Texas Instruments tools were employed: the *mmWave Demo Visualizer* and the *mmWave Sensing Estimator*. These tools allowed the rapid prototyping of chirp profiles and provided immediate feedback on their impact on range and velocity performance.

The **mmWave Demo Visualizer** [4], [5] is a browser-based application that enables basic sensor configuration and visualization. It allows the user to adjust frame rate, range resolution, maximum range, and velocity settings, while simultaneously observing output plots such as point clouds, noise profile, range profile, and Doppler heatmaps. This tool was particularly useful for testing standard chirp configurations and ensuring the correct operation of the radar hardware in a controlled environment. Figure 12 shows an example of the configuration interface.

Complementing this, the **mmWave Sensing Estimator** [7] provides a theoretical framework to design chirp profiles. By specifying parameters such as slope, bandwidth, ADC sampling rate, and frame timing, the tool outputs derived radar capabilities including range resolution, maximum detectable velocity, and unambiguous range. However, it must be emphasized that the estimator only generates feasible profiles. However, it does not guarantee hardware compatibility. Each configuration must be tested on the actual radar device to confirm correct execution. Figure 13 illustrates the chirp design interface of the estimator.

b) Sensor Transformations:

Each radar sensor in the dual-radar configuration was mounted with a yaw offset of $\pm 30°$, a pitch tilt of $15°$ upward, and a lateral displacement from the vehicle centerline. To ensure all

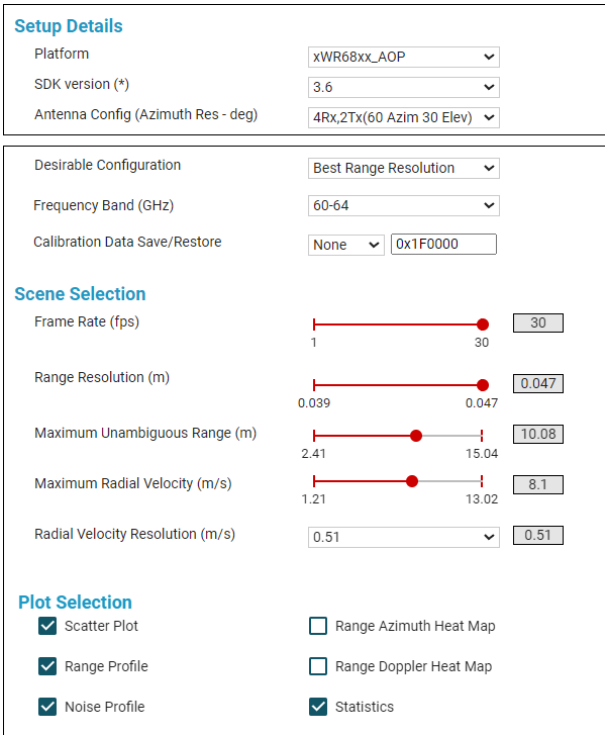


Fig. 12: mmWave Demo Visualizer interface for radar setup and visualization.
Source: Texas Instruments, available at dev.ti.com

Fig. 13: mmWave Sensing Estimator interface for chirp configuration design.
Source: Texas Instruments, available at dev.ti.com

detections could be expressed in a coherent vehicle-centric frame, the following extrinsic corrections were applied in sequence:

1) **Yaw rotation** R_{yaw}: compensates for the mounting angle around the Z-axis.
2) **Pitch rotation** R_{pitch}: corrects the upward tilt by rotating around the X-axis.
3) **Translation** $\vec{T}$: shifts the point clouds by the physical offsets of each radar.

The transformation from radar to vehicle coordinates is expressed as:

$$\vec{p}_{\text{veh}} = R_{\text{yaw}} \cdot R_{\text{pitch}} \cdot \vec{p}_{\text{radar}} + \vec{T} \tag{1}$$

These corrections ensured that detections from both radars overlapped coherently, avoiding duplicated or misaligned clusters. Without them, raw dual-sensor data appeared inconsistent, particularly when fusing Doppler-based velocity estimates.

After applying yaw, pitch, and translation adjustments, the fused point cloud served as a stable input for clustering, odometry, and tracking.

c) Chirp Configuration:

The IWR6843AOP operates as a Frequency-Modulated Continuous Wave (FMCW) radar, transmitting chirps whose frequency increases linearly over a bandwidth B during a duration T_c. Reflected signals return with a frequency shift that encodes target range, while Doppler shifts across successive chirps provide relative velocity.

The chirp is defined by:

- Start frequency f_0 [GHz]
- Frequency slope S [MHz/μs]
- Chirp duration T_c [μs]
- Idle time between chirps T_{idle} [μs]
- Sampling rate f_s [Msps]
- Bandwidth $B = S \cdot T_c$ [MHz]

From these parameters, the sensing performance is determined as:

$$\Delta R = \frac{c}{2B}\ [\text{m}], \qquad R_{\max} = \frac{c f_s}{2S}\ [\text{m}],$$

$$\Delta v = \frac{\lambda}{2N_c T_c}\ [\text{m/s}], \qquad v_{\max} = \frac{\lambda}{4T_c}\ [\text{m/s}],$$

where ΔR is **range resolution**, $R_{\max}$ the **maximum unambiguous range**, Δv the **velocity resolution**, and $v_{\max}$ the **maximum unambiguous velocity**.

Full-Bandwidth Chirp (Single 4 GHz Sweep), shown in Fig. 14, provides the finest possible range resolution by sweeping the entire 4 GHz spectrum in a single chirp. This configuration is well suited for precise spatial localization but requires high ADC sampling rates and is more prone to interference, making it practical mainly for single-radar setups.

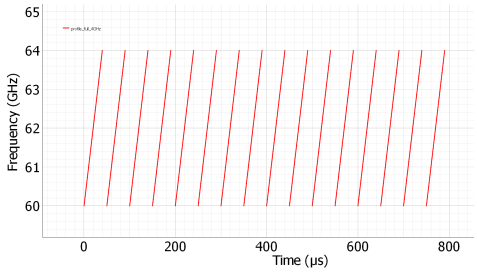


Fig. 14: Full-bandwidth chirp (single 4 GHz sweep). *Source: Texas Instruments, available in mmWave Demo Visualizer [4]*

Divided-Bandwidth Chirps (Two 2 GHz Sweeps), shown in Fig. 15, split the 4 GHz spectrum into two 2 GHz chirps. Although this reduces range resolution compared to a full-band sweep, it lowers ADC requirements and enables reliable operation of dual-radar systems.

In this setup, one chirp configuration was assigned to each radar (Config A $\rightarrow$ Radar A, Config B $\rightarrow$ Radar B). By operating on distinct frequency ranges, the radars avoided spectral overlap, preventing ghost targets and preserving point cloud quality.

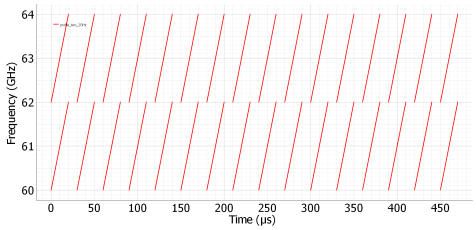


Fig. 15: Divided-band chirps (two 2 GHz sweeps). *Source: Texas Instruments, available in mmWave Demo Visualizer [4]*

The final parameters were derived using the *mmWave Sensing Estimator* tool [7]. The divided-band strategy provided a balanced trade-off between resolution and robustness, ensuring stable dual-radar operation while avoiding the interference issues inherent to full-bandwidth sweeps.

IMU Setup and Calibration

a) Configuration Tools:

The MTi-G-710 was first evaluated with *Xsens MT Manager* to understand the sensor outputs and axis conventions. Using the tool, quaternion orientation was visualized in real time while the unit was held still and under gentle rotations to verify stability and drift. Reference frames were checked so that the IMU heading and gravity vector aligned with the vehicle-centric frame used in the radar pipeline. Output rate, message content, and quaternion format were confirmed in this stage before any integration work. This step served only for validation and sanity checks; the final system consumed the IMU stream directly during experiments.

b) Calibration Aspects:

The IMU was mounted on a rigid, nearly horizontal plate at the rear of the vehicle to minimize vibrations and to avoid introducing tilt that would require post-processing. On power-up, the device was initialized with a short configuration sequence to enable quaternion output at the desired rate, as specified in the low-level protocol documentation [21]. This sequence places the device in configuration mode, sets the output configuration (quaternions at a fixed sample rate), and returns it to measurement mode for streaming. Because the MTi-G-710 is factory-calibrated and runs an internal sensor-fusion engine, no additional field calibration was required beyond correct mounting and a stable magnetic/metal environment [19]. GNSS and velocity outputs were not used in this work; only quaternions were consumed to provide a reliable attitude reference for aligning radar point clouds.

III. Processing Pipeline

Having defined the system hardware and sensor configurations, the next step is to establish the processing methodology that transforms raw sensor outputs into reliable odometry estimates. The proposed pipeline follows a modular structure, where each stage addresses a specific task: aligning radar frames through rigid-body transformations, merging dual-sensor data, filtering unreliable detections, clustering meaning-

ful structures, and finally estimating ego-motion using Iterative Closest Point (ICP).

The overall data flow is illustrated in Fig. 3, which summarizes how radar and IMU measurements are progressively refined before contributing to odometry estimation. This organization enables stepwise validation, as each module can be evaluated independently before integration into the complete system.

The following subsections present the mathematical formulations and algorithmic logic of each module in detail, supported by block diagrams to illustrate the transformations applied at each stage. By combining radar Doppler measurements with orientation inputs from the IMU, the pipeline is designed to deliver robust odometry even in environments where conventional vision- or LiDAR-based methods fail.

1) Rigid-Body Transformation: Each radar sensor in the dual-radar configuration is physically mounted with a specific orientation and tilt relative to the vehicle's forward direction. To ensure a common frame of reference for all points, it is necessary to compensate for:

- **Yaw rotation**: due to angled placement ($\pm 30^\circ$) of the radar modules.
- **Pitch tilt**: due to upward mounting tilt (15°), which must be compensated to recover horizontal geometry.
- **Sensor offset**: due to the physical separation of the radar sensors in the horizontal axis (X-axis).

a) Yaw Correction (Z-axis Rotation): The radar sensors are rotated relative to the vehicle frame:

- Radar A (Left): Mounted at $+30^\circ$ yaw $\Rightarrow$ compensated with -30° rotation.
- Radar B (Right): Mounted at -30° yaw $\Rightarrow$ compensated with $+30^\circ$ rotation.

The 2D rotation in the XY-plane is defined as:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

where $\theta = \pm 30^\circ$ depending on the sensor.

The yaw angles of $\pm 30^\circ$ were selected through a combination of physical measurement and simulation of the sensor field of view (FOV). Using the vehicle chassis centerline as a reference, the mounted sensors were visually measured to exhibit a yaw of approximately 28–32° outward (see Fig. 17), which confirmed the intended target of 30° opening from the vehicle center. This estimate was further cross-checked in simulation by plotting the radar FOV using the TI Demo Visualizer, which allows configuration of the theoretical horizontal FOV from its default 90° down to 60° or 30° respectfully. The 60° configuration was adopted as it provides improved angular resolution while discarding irrelevant detections outside the useful sector. Within this setting, the $\pm 30^\circ$ mounting maximizes the combined coverage of the dual-radar system, while creating a small blind zone of approximately 4 m directly in front of the vehicle. This trade-off was deliberately accepted to ensure the widest effective overlap of both radars for odometry tasks.

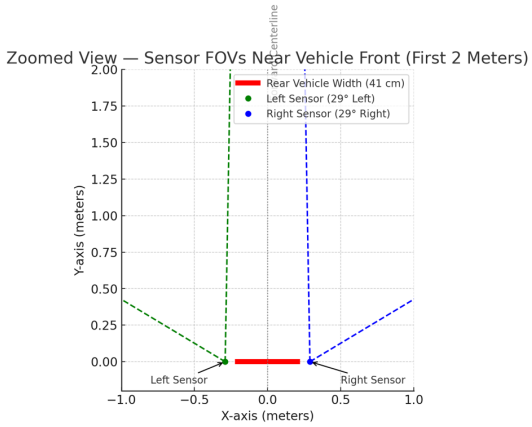


Fig. 16: Simulation of yaw rotation of sensors around the Z-axis ($\pm 30^\circ$).

Fig. 17: Real-world implementation of yaw rotation in the vehicle.

b) Pitch Compensation (X-axis Rotation): Since the radar sensors are tilted upward by 15°, a corrective rotation around the X-axis is applied to bring the points back to a horizontal perspective:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

where $\phi = -15^\circ$ (negative to reverse the upward tilt).

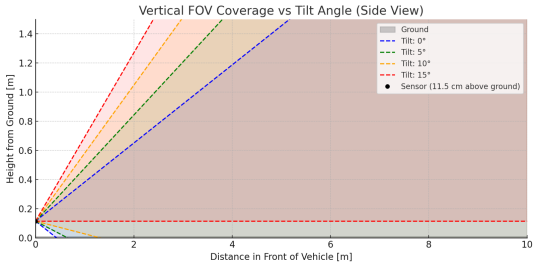


Fig. 18: Pitch compensation for 15° upward tilt.

The upward tilt of 15° was introduced to minimize clutter from the ground plane. With the sensor's vertical FOV of approximately 30°, this configuration reduced spurious reflections while maintaining sufficient detection of obstacles at the

intended height range. Both the yaw and pitch choices were tested through simulation and validated in real-world mounting experiments (Figures 16, 17, 18, 19).

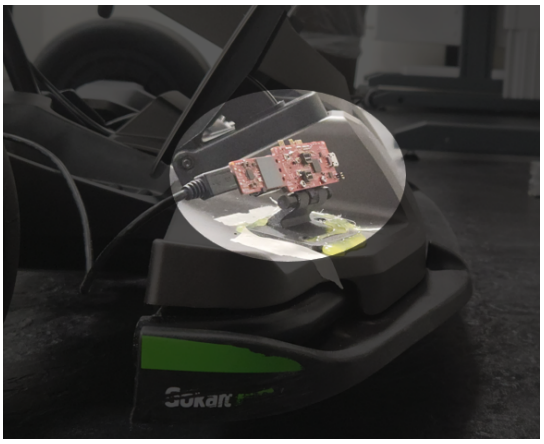


Fig. 19: Real-world implementation of radar tilt (15° upward).

c) X-Axis Offset Compensation: After rotation, each sensor's point cloud was translated along the X-axis to align with the vehicle's center:

- Radar A: $x \leftarrow x - 0.32$ meters
- Radar B: $x \leftarrow x - 0.28$ meters

This translation ensured both sensors were aligned in a common vehicle-centric frame.

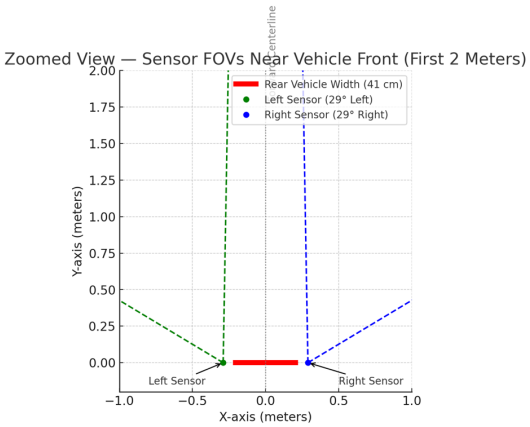


Fig. 20: Final extrinsic configuration combining yaw, pitch, and translation.

A yaw rotation R_{yaw} was first implemented, followed by a pitch correction R_{pitch}, and finally a translation vector $\vec{t}$ was applied to account for the physical position of the sensors with respect to the vehicle's coordinate origin.

The full transformation can be expressed as:

$$T_{\text{veh}} = R_{\text{yaw}} \cdot R_{\text{pitch}} \cdot \vec{p}_{\text{radar}} + \vec{T} \tag{2}$$

Where:

- $\vec{p}_{\text{radar}}$ is a radar point in sensor coordinates.
- R_{yaw} is a 2D rotation around the vertical axis ($\pm 30°$).
- R_{pitch} corrects for the upward sensor tilt ($-15°$).
- $\vec{T}$ is the translation vector.

d) Summary of Extrinsics:

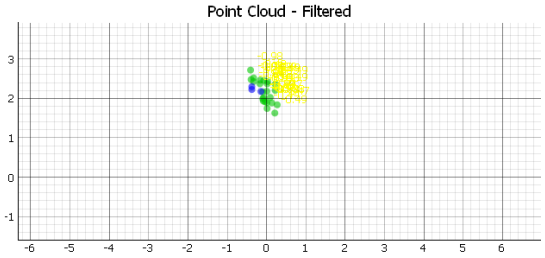
- **Left radar:** yaw $+30°$, pitch $-15°$, translation $(-0.32, 0, 0)$.
- **Right radar:** yaw $-30°$, pitch $-15°$, translation $(-0.28, 0, 0)$.

This extrinsic calibration ensures that dual-radar data is expressed in a coherent vehicle-centric frame, which is critical for downstream modules such as clustering, odometry, and obstacle tracking.

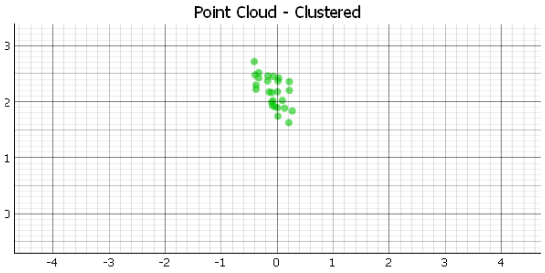
2) Radar Merge: Once each radar's detections were transformed into the common vehicle frame, the two datasets were merged into a single unified point cloud per frame. This step ensured that detections from both radars contributed consistently to the pipeline.

This fusion improves both density and coverage, particularly in regions where the individual radar fields of view overlap. It also reduces ambiguity during clustering, since targets detected by both sensors reinforce one another once aligned.

The effect of merging is visualized in Fig. 21, where the independent point clouds (top) are combined into a single dataset (bottom).



(a) Independent point clouds before merging.



(b) Unified point cloud after merging.

Fig. 21: Radar merge process: alignment of calibrated detections into a single dataset.

3) Frame Aggregator: The frame aggregator addresses the inherent sparsity of single radar frames by combining multiple consecutive frames into a denser and more reliable point cloud. While aggregation increases both useful information and noise, it ultimately improves object reconstruction, motion estimation, and detection stability.

a) Single Frame: Processing a single frame often leads to incomplete or fragmented detections:

- Objects may not be fully represented due to the low number of points.
- Close objects can merge or be misclassified because of insufficient separation in sparse data.

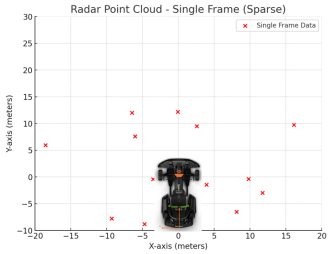


Fig. 22: Single-frame visualization.

This sparsity-induced ambiguity is not a limitation of the clustering algorithms themselves, but a direct consequence of limited raw data.

b) Multiple Frames: To mitigate these issues, multiple frames are aggregated in a fixed-size buffer, where older frames are removed as new ones arrive. This method increases point cloud density, reduces variance as per the Law of Large Numbers, and stabilizes detections:

$$\frac{\sigma^2}{N} \tag{3}$$

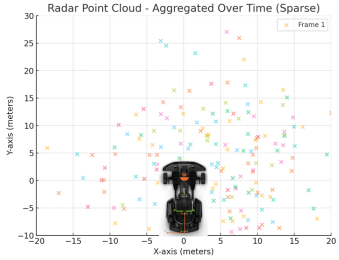


Fig. 23: Multi-frame aggregation visualization.

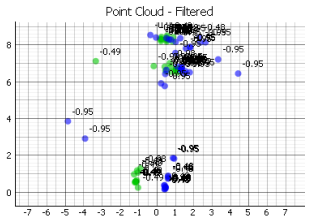
During experimentation, the number of aggregated frames was varied between 7 and 15. A value of 10 frames provided the best balance: enough temporal density to reduce noise and reveal stable structures, while minimizing the persistence of outdated points. This configuration was therefore adopted for all subsequent pipeline stages.

4) Filtering Stage: y-axis and Doppler: The raw radar point cloud exhibited systematic clutter in the region directly in front of the sensors, primarily due to reflections from the driver's feet and the ground. These unwanted detections were consistently observed up to approximately $1.5\,\mathrm{m}$ and persisted even when the vehicle was stationary. If left unfiltered, such clutter propagated through the pipeline, degrading clustering and motion estimation.

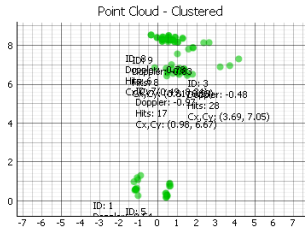
To address this, a static filtering stage was implemented with two criteria:

1) **y-axis filter:** Only points within the interval $1.5 \leq y \leq 15$ m are retained. This introduces a *dead zone* in front of the vehicle, removing ground clutter and very short-range reflections while keeping the operational range relevant for odometry.
2) **Doppler filter:** Only points with radial velocity within $0.01 \leq v_d \leq 8.0$ m/s are accepted. This rejects near-zero Doppler detections (often noise or static artifacts) and unrealistically large outliers.

This filtering approach significantly reduced spurious reflections (Fig. 24), producing a cleaner point cloud for downstream modules. The trade-off is that genuine obstacles appearing within the $1.5\,\mathrm{m}$ dead zone are excluded. Although this removes potentially useful information in very close range, the region is dominated by ground clutter and considered less critical for the odometry pipeline.



(a) Clutter representation (Radar A = blue, Radar B = green).



(b) Clustered view of clutter detections.

Fig. 24: Visualization of unwanted reflections (clutter) near the sensors, dominated by ground and driver-related reflections up to $1.5\,\mathrm{m}$. Blue dots correspond to Radar A (left), and green dots to Radar B (right).

5) RANSAC-Based Motion Filtering: A key stage in the pipeline is filtering static landmarks from dynamic objects to ensure ego-motion estimation is based only on stable references. To achieve this, a RANSAC-based model is fitted to the relationship between azimuth angle θ and Doppler velocity v_d.

a) Principle and Theory: Static reflections observed by a moving radar follow a predictable Doppler pattern:

$$v_d = v_e \cos(\theta),$$

where v_e is the vehicle's forward velocity. Objects directly in front exhibit $v_d \approx v_e$, while those at 90° azimuth have $v_d \approx 0$, and intermediate azimuths form a smooth curve. Dynamic targets deviate from this distribution, appearing as outliers.

Figure 25 illustrates this concept using simulated data.

b) Implementation: In practice, directly fitting the cosine model requires prior knowledge of v_e, which is not yet available at this stage of the pipeline. Instead, a polynomial approximation is used:

$$v_d = a\theta^2 + b\theta + c,$$

fitted with **RANSACRegressor**.

The algorithm iteratively:

1) Samples small subsets of (θ, v_d) pairs.
2) Fits a quadratic model.

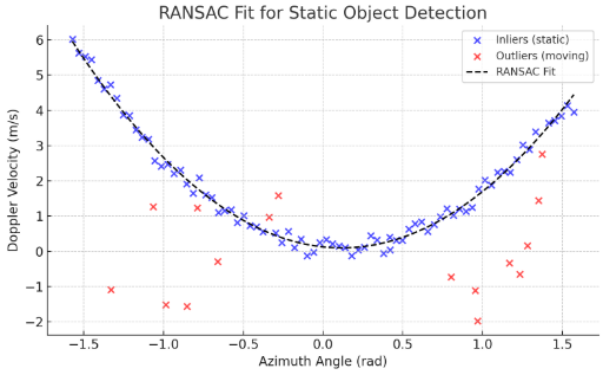


Fig. 25: Simulated data: RANSAC fit of Doppler vs. azimuth. Inliers = static (blue), Outliers = moving (red).

3) Evaluates inliers based on residual error.
4) Selects the model with maximum consensus.

This approach bends the quadratic to approximate the Doppler distribution of static objects, which in the radar's limited FOV resembles half of a cosine curve. Inliers are classified as static reflections, while outliers correspond to dynamic objects that must be excluded from ego-motion estimation.

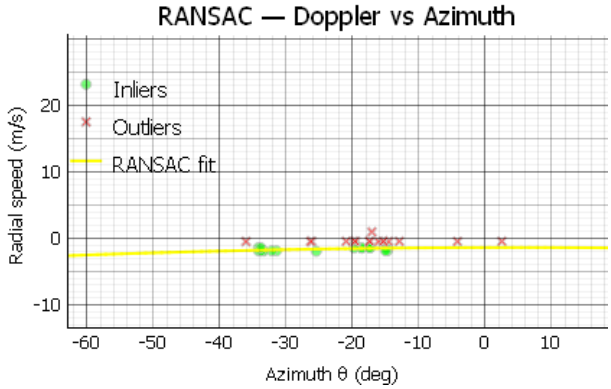


Fig. 26: Real implementation of RANSAC filtering: green = static inliers, red = dynamic outliers.

c) Pipeline Role: As illustrated in the system pipeline (Fig. 3), the RANSAC filter is positioned after static thresholding and before ego-velocity estimation. Its primary purpose at this stage is to separate dynamic objects from static landmarks, ensuring that subsequent modules operate only on consistent and reliable data.

Without this step, moving objects would propagate through clustering and ICP, corrupting the estimation of ego-motion. By filtering out dynamic outliers, RANSAC guarantees that only stable reflections are retained for trajectory alignment and speed estimation.

The output of this stage is therefore two-fold:

- A refined set of static landmarks, which serve as input for clustering and ICP-based odometry,
- A clean separation of dynamic actors, preventing them from biasing ego-motion estimation.

6) Clustering: Clustering is a key technique for grouping similar data points based on their spatial or statistical characteristics. In automotive sensing, clustering enables the detection and tracking of relevant objects such as vehicles, pedestrians, and obstacles by structuring raw radar detections into meaningful groups.

Among commonly used clustering algorithms, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) stands out due to its robustness in handling irregular cluster shapes and its ability to automatically identify noise points without requiring a predefined number of clusters [16]. This contrasts with centroid-based methods such as K-Means, which require specifying K in advance and struggle with irregularly shaped clusters, and with hierarchical methods like agglomerative clustering, which can capture hierarchical structures but are computationally expensive and sensitive to noise.

Algorithm	Type	Strengths	Weaknesses	Best Use Case
DBSCAN	Density-Based	• Detects clusters of varying shapes and sizes • Identifies outliers (noise points)	• Computationally expensive for large datasets	Radar object detection with dynamic object counts and noise filtering
K-Means	Centroid-Based	• Fast and efficient for large datasets	• Requires fixed number of clusters (K) • Poor performance with irregular shapes	Structured radar data with known object counts
Agglomerative	Hierarchical	• No prior knowledge of cluster number required • Can detect hierarchical structures	• Sensitive to noise • High computational cost	Offline grouping of radar signatures

TABLE II: Comparison of clustering algorithms.

DBSCAN defines a cluster based on two parameters: a neighborhood radius ε and a minimum number of points MinPts. For a given point p_i with coordinates (x_i, y_i), its ε-neighborhood is defined as

$$\mathcal{N}_\varepsilon(p_i) = \{p_j \mid \|p_j - p_i\|_2 \leq \varepsilon\}. \tag{4}$$

A point p_i is called a *core point* if

$$|\mathcal{N}_\varepsilon(p_i)| \geq \text{MinPts}. \tag{5}$$

Clusters are then formed by connecting density-reachable points, while points that do not belong to any cluster are classified as noise. This density-based formulation makes DBSCAN well-suited for radar data, where detections of the same object tend to be locally dense, while clutter appears as isolated points.

a) Two-Stage DBSCAN Rationale: In practice, a single parameter set for $(\varepsilon, \text{MinPts})$ is insufficient because:

- A large ε merges nearby objects into one cluster.
- A small ε causes fragmentation or discards weak reflections.

To address this, a two-stage DBSCAN process was implemented:

Stage 1: Permissive Filtering

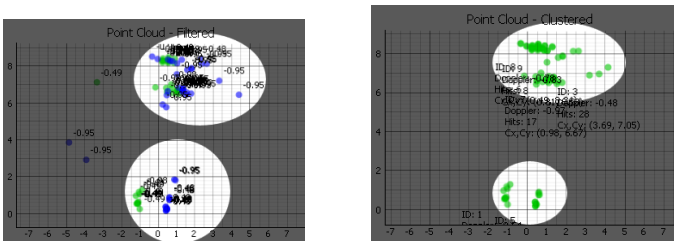
$$\varepsilon_1 = 2\,\text{m}, \quad \text{MinPts}_1 = 2 \tag{6}$$

This stage acts as a noise filter by discarding points that cannot form even small local clusters. Outliers are eliminated early, reducing computational load for the next stage.

Stage 2: Fine Clustering

$$\varepsilon_2 = 1\,\text{m}, \quad \text{MinPts}_2 = 4 \tag{7}$$

This stage applies stricter parameters to avoid merging distinct objects into a single cluster. It ensures compact clusters that more accurately reflect real objects in the environment.



(a) Pre-clustering view (Radar A = blue, Radar B = green).

(b) Post-clustering view showing grouped detections.

Fig. 27: Effect of clustering on radar point clouds. Before clustering, numerous unwanted reflections appear close to the sensors, dominated by ground and driver-related clutter. After clustering, these spurious points are mostly removed, but at the cost of discarding some valid detections that fail to meet the density criteria. Blue = Radar A (left), Green = Radar B (right).

The resulting process can be expressed as a composition of the two operators:

$$\mathcal{C}_{\text{final}} = \text{DBSCAN}(\varepsilon_2, \text{MinPts}_2,\ \text{DBSCAN}(\varepsilon_1, \text{MinPts}_1,\ P)), \tag{8}$$

where P denotes the raw radar point cloud and $\mathcal{C}_{\text{final}}$ the final set of clusters.

This formulation highlights the dual role of DBSCAN: first as a noise filter, then as a clustering method. By structuring the problem in two passes, the algorithm maintains robustness to noise while achieving finer object separation in dense scenarios.

7) Cluster Tracking: Once clusters have been identified, it is necessary to track them over consecutive frames to ensure temporal consistency. A cluster tracker assigns each detected cluster a unique identifier (ID) and maintains a history of its trajectory over time. This enables the system to distinguish between persistent static objects and transient detections caused by noise or dynamic obstacles.

a) Hits and Misses: Each tracked cluster is updated at every frame using two indices:

- **Hit Count (h):** The number of consecutive frames in which the cluster has been successfully matched to a detection.
- **Miss Count (m):** The number of consecutive frames in which the cluster has failed to find a corresponding detection.

A cluster is considered *active* if $m \leq M_{\text{max}}$, where M_{max} is the maximum allowed misses before deletion. This mechanism ensures that short occlusions or temporary sensor noise do not cause immediate loss of tracks.

b) Geometric Association: Let $c_t = (x_t, y_t)$ denote the centroid of a cluster at frame t. For each existing track, association with a new detection is performed by minimizing the spatial distance:

$$d(c_t, c_{t+1}) = \|c_t - c_{t+1}\|_2. \tag{9}$$

If a fitted motion model is available, the distance is instead evaluated relative to the predicted cluster trajectory. In this project, a simple line-fitting approach was applied to the history of each cluster's centroids:

$$y = mx + b, \tag{10}$$

where (m, b) are obtained via least-squares regression over the last N centroids. The orthogonal distance of a new detection (x_0, y_0) to this fitted line is computed as:

$$d_\perp(x_0, y_0) = \frac{|mx_0 - y_0 + b|}{\sqrt{m^2 + 1}}. \tag{11}$$

The detection is associated with the track if $d_\perp \leq d_{\text{max}}$, where d_{max} is a configurable threshold.

c) Track Management: The complete tracking procedure consists of:

1) **Initialization:** If no tracks exist, each cluster spawns a new track with a unique ID.
2) **Association:** Existing tracks are matched to current detections using the minimum distance criterion.
3) **Update:** Matched tracks increment their hit count ($h \leftarrow h + 1$), reset their miss count, and append the centroid to their history.
4) **New Track Creation:** Unmatched detections start new tracks with fresh IDs.
5) **Deletion:** Tracks exceeding M_{max} consecutive misses are deleted.

This process ensures that each stationary object is consistently represented by the same track ID across frames, while spurious detections are naturally filtered out.

To illustrate the method, Fig. 29 compares the clustering output before and after applying the tracker. In Fig. 29(a), clusters are detected but lack temporal identity across frames. In Fig. 29(b), the tracker assigns each cluster a unique ID and maintains attributes such as Doppler, centroid coordinates, and hit count. This ensures consistent representation across frames and enables reliable identification of stationary obstacles.

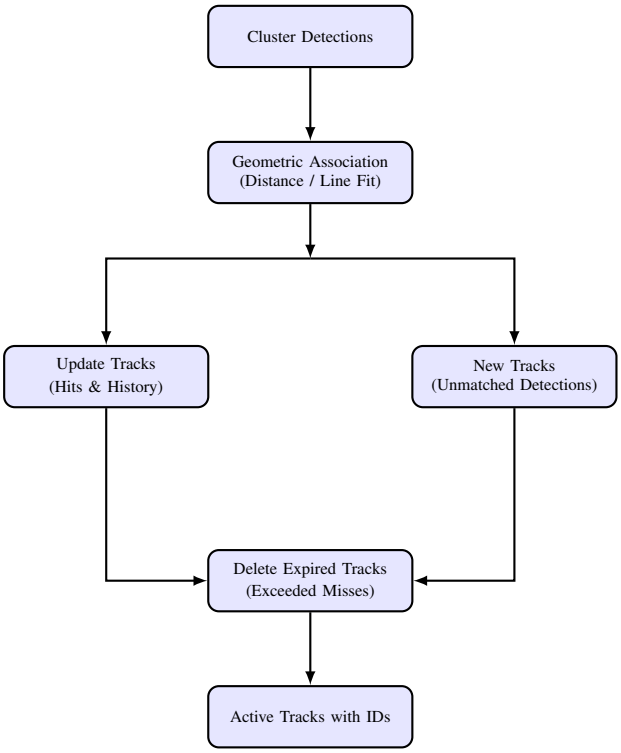
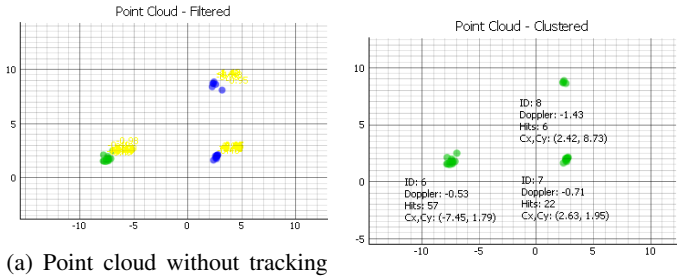


Fig. 28: Cluster tracking block diagram. Each detection is associated with existing tracks or used to initialize new tracks. Tracks are deleted after exceeding miss thresholds.



(a) Point cloud without tracking (Radar A = blue, Radar B = green).

(b) Tracked clusters with assigned IDs and attributes.

Fig. 29: Illustration of the cluster tracking process. (a) Clusters detected but without temporal tracking. (b) Tracked clusters with IDs, Doppler averages, centroids, and hit counts.

8) Iterative Closest Point (ICP): The Iterative Closest Point (ICP) algorithm is a classical method for estimating rigid-body transformations between two point sets. In the context of radar odometry, ICP provides a way to align consecutive frames of radar point clouds in order to estimate ego-motion. The main goal is to determine the relative rotation $R \in SO(2)$ and translation $t \in \mathbb{R}^2$ that minimize the alignment error between a source point cloud $P = \{p_i\}_{i=1}^{N}$ and a target point cloud $Q = \{q_j\}_{j=1}^{M}$.

a) Mathematical Formulation: The ICP algorithm can be separated into two main steps: correspondence search and transformation estimation.

Correspondence Search. For each point $p_i \in P$, the closest point in Q is selected according to the Euclidean distance:

$$q_i^* = \arg\min_{q_j \in Q} \|p_i - q_j\|_2. \tag{12}$$

This step establishes a set of tentative correspondences $\{(p_i, q_i^*)\}$ between the two frames. Efficient implementations use spatial data structures such as KD-trees to accelerate nearest-neighbor searches.

Transformation Estimation. Once correspondences are found, the rigid-body transformation (R, t) is estimated by minimizing the least-squares error:

$$E(R,t) = \sum_{i=1}^{N} \|p_i - (Rq_i^* + t)\|^2. \tag{13}$$

The optimal solution can be obtained using singular value decomposition (SVD) of the cross-covariance matrix between the two point sets. In 2D, the rotation can be parameterized by an angle θ, such that:

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{14}$$

The transformation is then applied to Q, and the process is repeated iteratively until convergence, typically when changes fall below a threshold or a maximum number of iterations is reached.

b) Global ICP: In the global variant, all available points of the radar point cloud are used to compute correspondences and estimate the motion. This maximizes the use of sensor information but can be sensitive to noise and dynamic objects, since outliers directly affect the transformation estimate.

c) Cluster-wise ICP: To increase robustness, a cluster-wise approach was also investigated. Instead of using the entire point cloud, correspondences are restricted to clusters identified in the previous stage (see Section III-7). Among these clusters, priority is given to those with the highest *hit count* and lowest *miss count*, since they are more likely to represent stable, stationary objects. For each cluster C_k, an independent ICP alignment is performed:

$$E_k(R,t) = \sum_{p_i \in P_k} \|p_i - (Rq_i^* + t)\|^2, \tag{15}$$

where P_k and Q_k denote corresponding cluster subsets across two consecutive frames. The resulting transformations are then aggregated, typically via weighted averaging based on cluster stability, to obtain the final ego-motion estimate.

d) Practical Considerations: Several practical aspects influence ICP performance:

- **Initialization:** ICP assumes small inter-frame motion. Poor initialization may cause convergence to local minima.
- **Nearest-neighbor search:** KD-trees enable efficient correspondence search in $\mathcal{O}(N \log N)$ time.
- **Outlier rejection:** Correspondences with distances exceeding a threshold $d_{\max}$ are discarded to reduce the influence of spurious points.

- **Cluster selection:** Restricting ICP to stable clusters improves robustness against moving objects and noise.

IV. Odometry Results

This section presents the result of the implemented pipeline as shown in Section III with the dual-radar setup. Two experimental environments were selected: a controlled laboratory environment and an outdoor scenario.

A. Laboratory Environment

1) Scenario Description: The first experiments were conducted in an underground laboratory space (Fig. 30), which provided a controlled but reflective environment. The room contained multiple devices and metallic structures from other projects, which acted as additional landmarks.

The main obstacle was created using a set of modular cushions arranged into a wall (black and orange blocks in Fig. 30). This configuration allowed us to test both straight-line odometry and more complex trajectories involving more complex set of trajectories.

Two tests were performed:

- **Straight-line test:** The vehicle was driven directly toward the cushion wall, covering a distance of approximately 10–11 m from the start point to the wall.
- **Obstacle-avoidance test:** The vehicle followed a path around the cushion wall and returned to its starting location, introducing turning maneuvers that add stress to the tested system and additional opportunities for clutter-induced reflections.

2) Odometry Results: Resulting trajectories from the recordings are shown from the indoor tests in Fig. 30. In this scenario a **U-Turn trajectory** was performed to test the robustness of the pipeline in an enclosed environment with high reflections. This test provide the oportunity of a tight turning tracking of the ego-motion estimation. As shown in Fig. 31, this test had 2 results, in this case the ICP operation on the whole point-cloud resulted in a more accurate ego-motion estimation. While the clustered one provided a trajectory not far from the one performed, it was still lacking in accuracy and consistency. As the total distance from the starting point to the wall that we are avoiding is around 10 meters.

B. Outdoors Scenario

1) Scenario Description: The second evaluation environment was the outdoor parking lot of Fachhochschule Dortmund (see Fig. 32). The tests were conducted shortly after rainfall, which introduced additional radar clutter from wet ground reflections, making this a challenging stress test for the pipeline.

The test area corresponds to a rectangular section of the parking lot, bounded by the building facades and a row of parked vehicles. Key dimensions were measured as shown in Fig. 33, with the vertical segment $L_1 \approx 10.7\,\mathrm{m}$ and the horizontal segment $L_2 \approx 54.1\,\mathrm{m}$. Combined, these form an L-shaped trajectory of approximately $62.3\,\mathrm{m}$ (Fig. 34).

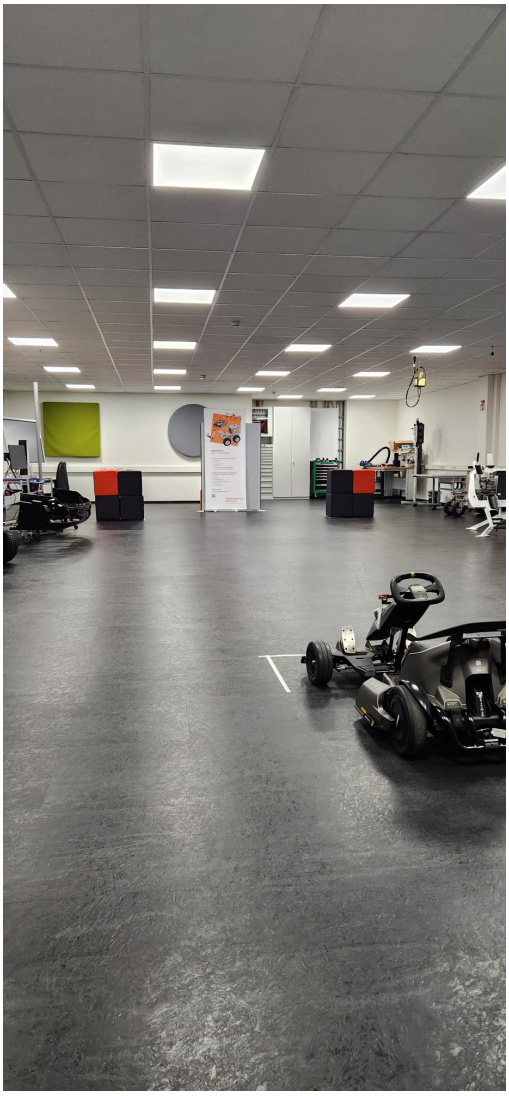
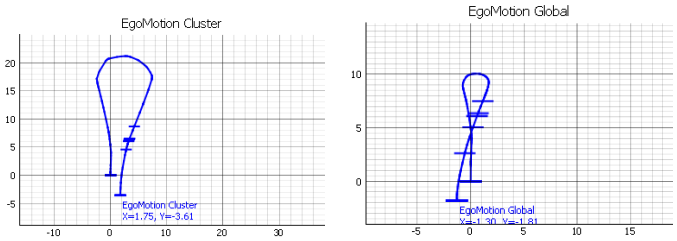


Fig. 30: Laboratory environment setup. The cushion wall (black and orange) served as the main obstacle, while surrounding equipment contributed additional reflective surfaces.



(a) Vertical segment $L_1 \approx 10.7$ m.

(b) Horizontal segment $L_2 \approx 54.1$ m.

Fig. 31: Measured distances of the parking lot test area used for defining the L-shaped trajectory.
Source: Google Maps, available at https://maps.google.com [22]

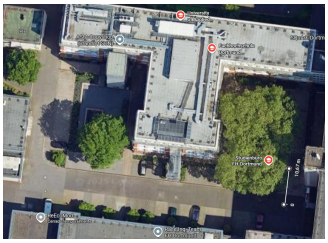
During the experiments, between 6 and 10 vehicles were parked in the area, providing strong reflective structures that contributed to odometry alignment.

Two trajectories were tested:

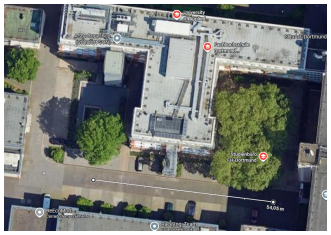
- **L-shape:** Starting along L_2, followed by a turn into L_1.
- **L-shape with return:** Same as above, but including a U-turn to return to the starting point.

All tests were driven exclusively in forward motion, as Doppler-based differentiation between forward and reverse trajectories was not implemented at this stage.

Fig. 32: Overview of the outdoor parking lot test environment at Fachhochschule Dortmund.
Source: Google Maps, available at https://maps.google.com [22]



(a) Vertical segment $L_1 \approx 10.7$ m.



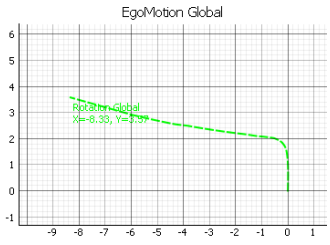
(b) Horizontal segment $L_2 \approx 54.1$ m.

Fig. 33: Measured distances of the parking lot test area used for defining the L-shaped trajectory.
Source: Google Maps, available at https://maps.google.com [22]

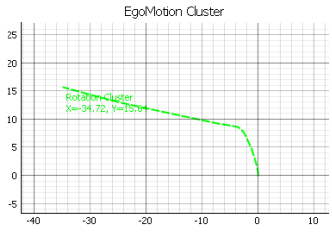
2) Odometry Results: Resulting trajectories from the recordings are shown from the outdoor tests in Fig. 35 and in Fig. 36. During this test the non idealistic weather allowed us to perform a perfect performance test for both of the solution in the outdoors environment. In this testing period light rainfall was present, however nothing to damage the hardware. In both scenarios an **L shape trajectory** was performed, however both where run at different lengths of the testing area. As shown in Fig. 35, the test was stoped around the metal emergency stair, which is around of 30 meters of the L1 section. In Fig. 36 the full length was covered in which we get a pretty close

Fig. 34: Combined L-shaped trajectory of approximately 62.3 m ($L_1 + L_2$).
Source: Google Maps, available at https://maps.google.com [22]

estimation of the ego-motion of the vehicle converting this into a close estimation of the vehicle odometry.

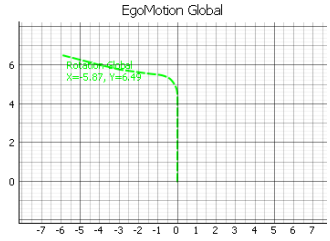


(a) Vehicle full point-cloud solution ICP ego-motion estimation. $L_1 \approx 10$ m., $L_2 \approx 30$ m.

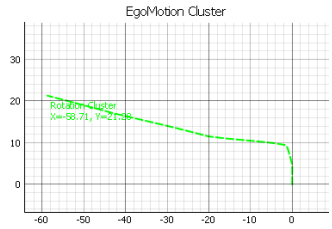


(b) Vehicle clustered solution ICP ego-motion estimation. $L_1 \approx 10$ m., $L_2 \approx 30$ m.

Fig. 35: Outside test recording, reduced trajectory due to weather. $L_1 \approx 10$ m., $L_2 \approx 30$ m.



(a) Vehicle full point-cloud solution ICP ego-motion estimation. $L_1 \approx 10$ m., $L_2 \approx 60$ m.

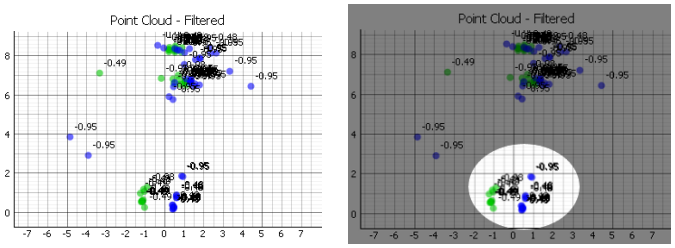


(b) Vehicle clustered solution ICP ego-motion estimation. $L_1 \approx 10$ m., $L_2 \approx 60$ m.

Fig. 36: Outside test recording, full planned trajectory due to weather. $L_1 \approx 10$ m., $L_2 \approx 60$ m.

3) Observations: The outdoor experiments highlighted the following:

- Wet ground introduced additional clutter, especially near the dead zone as shown in Fig. 37, reinforcing the relevance of y-axis filtering.
- Parked vehicles provided strong static references that stabilized ICP alignment, particularly during turning maneuvers.



(a) Full point-cloud of the observed clutter to aid the visualization of the whole point-cloud.

(b) Highlighted clutter in the point-cloud to aid the visualization of the clutter.

Fig. 37: Outside test recording, full planned trajectory due to weather. $L_1 \approx 10$ m., $L_2 \approx 60$ m.

Overall, the parking lot tests confirmed the pipeline's robustness in real-world outdoor conditions with environmental noise and complex reflectors, while also exposing drift-related challenges over extended distances.

C. Discussion

The two evaluation environments highlight different strengths and weaknesses of the proposed odometry pipeline.

In the **laboratory environment**, performing ICP directly on the full point cloud delivered more stable results. The confined space, with multiple reflective surfaces and repeatable landmarks, produced a dense and consistent set of static detections. In this context, using all available points allowed ICP to maximize geometric alignment and minimize drift, even without the need for aggressive filtering of dynamic elements.

In contrast, the **outdoor parking lot scenario** posed greater challenges. Non-ideal environmental conditions, such as recent rainfall, made this setting an effective stress test for the system. Sparse long-range detections and the presence of moving vehicles further contributed to a less consistent point cloud. Applying ICP on the entire point set in this environment led to instability and drift, as spurious or dynamic points distorted the alignment process. Instead, the clustered ICP approach—selecting the most consistent tracked cluster as the alignment reference—proved more effective. By anchoring to a reliable static object, the system achieved better trajectory consistency across extended distances.

This comparison underlines a fundamental trade-off:

- **Full point cloud ICP** excels in structured, controlled environments with dense and reliable landmarks.
- **Cluster-based ICP** provides robustness in unstructured or outdoor environments, where clutter and dynamics reduce the reliability of the full point cloud.

Although improvements are possible in both settings (e.g., more refined filtering, adaptive ICP weighting), the experiments suggest that cluster-based ICP is more suited for outdoor odometry, while full point cloud ICP can be leveraged indoors where landmark consistency is higher.

V. Summary and Outlook

Insert conclusion here

REFERENCES

[1] Segway Inc., "Ninebot Go-kart PRO product page", 2025, Webpage. [Online]. Available: https://de-de.segway.com/products/ninebot-gokart-pro

[2] Texas Instruments, "IWR1642: difference between AWR and IWR parts", 2025, Webpage. [Online]. Available: https://e2e.ti.com/support/sensors-group/sensors/f/sensors-forum/742730/iwr1642-difference-between-awr-and-iwr-parts

[3] Texas Instruments, "IWR6843AOPEVM product page", 2025, Webpage. [Online]. Available: https://www.ti.com/tool/IWR6843AOPEVM

[4] Texas Instruments, "User's Guide mmWave Demo Visualizer," 2020, Online Document. [Online]. Available: https://www.ti.com/lit/ug/swru529c/swru529c.pdf?ts=1742817596204.

[5] Texas Instruments, "mmWave Demo Visualizer", 2025, Webpage. [Online]. Available: https://dev.ti.com/gallery/view/mmwave/mmWave_Demo_Visualizer/ver/3.6.0/

[6] Texas Instruments, "Understanding UART Data Output Format", 2025, Webpage. [Online]. Available: https://dev.ti.com/tirex/content/radar_toolbox_2_20_00_05/docs/software_guides/Understanding_UART_Data_Output_Format.html

[7] Texas Instruments, "mmWave Sensing Estimator", 2025, Webpage. [Online]. Available: https://dev.ti.com/gallery/view/mmwave/mmWaveSensingEstimator/ver/2.5.1/

[8] ub4raf, "Ninebot-PROTOCOL", 2025, GitHub Repository. [Online]. Available: https://github.com/ub4raf/Ninebot-PROTOCOL

[9] -, "Ninebot ES Communicaton Protocol", 2019, Webpage. [Online]. Available: https://cloud.scooterhacking.org/release/nbdoc.pdf

[10] -, "numpy.polyfit documentation", 2025, Webpage. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.polyfit.html

[11] Ç. Önen, A. Pandharipande, G. Joseph, and N. J. Myers, "Occupancy Grid Mapping for Automotive Driving Exploiting Clustered Sparsity," *IEEE Sensors Journal*, vol. 24, no. 7, pp. 9240-9250, 2024. [Online]. Available: https://doi.org/10.1109/JSEN.2023.3342463.

[12] D. Casado Herraez, M. Zeller, L. Chang, I. Vizzo, M. Heidingsfeld, and C. Stachniss, "Radar-Only Odometry and Mapping for Autonomous Vehicles," *arXiv preprint*, 2023. [Online]. Available: https://arxiv.org/abs/2305.12409

[13] S. R. Bhatt, B. S. Nadiger, R. Parthasarathy, and H. M. Shetty, "Instantaneous Ego-motion Estimation Using Doppler Radar," *IEEE Sensors Letters*, vol. 7, no. 5, pp. 1–4, 2023. [Online]. Available: https://doi.org/10.1109/LSENS.2023.3244030

[14] C. E. Beal, T. Williams, J. Pauli, M. Mukadam, and B. Boots, "Robust Off-Road Autonomy Using Multimodal Sensor Fusion," in *Proc. of the Conference on Robot Learning (CoRL)*, 2023. [Online]. Available: https://openreview.net/forum?id=kmiZqSgoAt

[15] B. Sundaralingam, C. E. Beal, and B. Boots, "Robust High-Speed State Estimation for Off-Road Autonomous Vehicles," in *Proc. of Robotics: Science and Systems (RSS)*, 2023. [Online]. Available: https://openreview.net/forum?id=3JpFLY3ihix

[16] GeeksforGeeks, *DBSCAN Clustering in ML — Density Based Clustering*, 2023. [Online]. Available: https://www.geeksforgeeks.org/dbscan-clustering-in-ml-density-based-clustering/. [Accessed: 19-Mar-2025].

[17] MathWorks, *Understanding Kalman Filters, Part 3: Optimal State Estimator*, 2017. Available at: https://la.mathworks.com/videos/understanding-kalman-filters-part-3-optimal-state-estimator--1490710645421.html (Accessed: March 23, 2025).

[18] Texas Instruments, *Radar Toolbox – mmWave Sensor Configuration and Demos*, 2024. Available at: https://dev.ti.com/tirex/explore/node?node=A__ADnbI7zK9bSRgZqeAxprvQ__radar_toolbox__1AslXXD__2.20.00.05 (Accessed: March 23, 2025).

[19] Movella (Xsens), *MTi User Manual*, 2023. [Online]. Available: https://www.xsens.com/hubfs/Downloads/usermanual/MTi_usermanual.pdf. [Accessed: 26-Sep-2025].

[20] Scottapotamas, *Xsens MTi Device Interface and Parser*, GitHub repository, 2020. [Online]. Available: https://github.com/Scottapotamas/xsens-mti. [Accessed: 26-Sep-2025].

[21] Movella (Xsens), *Low-Level Communication Protocol Documentation*, 2023. [Online]. Available: https://www.xsens.com/hubfs/Downloads/Manuals/MT_Low-Level_Documentation.pdf. [Accessed: 27-Sep-2025].

[22] Google Maps, *Fachhochschule Dortmund - Parking Lot Overview*, 2025. [Online]. Available: Google Maps. [Accessed: 01-Oct-2025].