Radar and LiDAR Fusion for Scaled Vehicle Sensing

May 2020 | Final Report

SAFE-D
SAFETY THROUGH DISRUPTION

Texas A&M Transportation Institute

San Diego State University
Leadership Starts Here

VT | TRANSPORTATION INSTITUTE
VIRGINIA TECH

Disclaimer

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated in the interest of information exchange. The report is funded, partially or entirely, by a grant from the U.S. Department of Transportation's University Transportation Centers Program. However, the U.S. Government assumes no liability for the contents or use thereof.

TECHNICAL REPORT DOCUMENTATION PAGE

1. Report No. VTTI-00-025	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle LiDAR and Radar Fusion for Scaled Vehicle Sensing		5. Report Date May 2022 6. Performing Organization Code:
7. Author(s) Gregory T. Beale, Matthew D. Berkemeier, Zachary R. Doerzaph, Miguel A. Perez		8. Performing Organization Report No. VTTI-00-025
9. Performing Organization Name and Address: Safe-D National UTC Virginia Tech Transportation Institute 3500 Transportation Research Plaza Blacksburg, VA 24061		10. Work Unit No. 11. Contract or Grant No. 69A3551747115/00-025
12. Sponsoring Agency Name and Address Office of the Secretary of Transportation (OST) U.S. Department of Transportation (US DOT)		13. Type of Report and Period Final Research Report 14. Sponsoring Agency Code

15. Supplementary Notes
This project was funded by the Safety through Disruption (Safe-D) National University Transportation Center, a grant from the U.S. Department of Transportation – Office of the Assistant Secretary for Research and Technology, University Transportation Centers Program.

16. Abstract
Scaled test beds are popular tools to develop and physically test algorithms for advanced driving systems, but they often lack automotive-grade radars in their sensor suites. To overcome resolution issues when using a radar at small scale, a high-level radar and automotive-grade LiDAR sensor fusion approach that effectively leveraged the higher spatial resolution of LiDAR was proposed. First, radar tracking software (RTS) was developed to track a maneuvering full-scale vehicle using an extended Kalman filter (EKF) and a popular data association technique. Second, a 1/5th scaled vehicle performed the same vehicle maneuvers but scaled to approximately 1/5th the distance and speed. When considering the scaling factor, the RTS's positional error at small scale was over 5 times higher on average than in the full-scale trials. Third, LiDAR object tracks were generated for the small-scale trials using a second EKF implementation and then combined with the radar objects in a high-level track fusion algorithm. The fused tracks demonstrated a 30% increase in positional accuracy for a majority of the small-scale trials when compared to tracks using just the radar or just the LiDAR. The proposed track fuser could allow scaled test beds to incorporate automotive radars into their sensor suites more effectively by augmenting the radar output with LiDAR, overcoming the resolution issues that afflict radar when operating at small scale.

17. Key Words Radar tracking, LiDAR tracking, Kalman filter, track-to-track fusion, object tracking, sensor fusion, scaled test bed	18. Distribution Statement No restrictions. This document is available to the public through the Safe-D National UTC website, as well as the following repositories: VTechWorks, The National Transportation Library, The Transportation Library, Volpe National Transportation Systems Center, Federal Highway Administration Research Library, and the National Technical Reports Library.

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 40	22. Price $0

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

Abstract

Scaled test beds are popular tools to develop and physically test algorithms for advanced driving systems, but they often lack automotive-grade radars in their sensor suites. To overcome resolution issues when using a radar at small scale, a high-level radar and automotive-grade LiDAR sensor fusion approach that effectively leveraged the higher spatial resolution of LiDAR was proposed. First, radar tracking software (RTS) was developed to track a maneuvering full-scale vehicle using an extended Kalman filter (EKF) and a popular data association technique. Second, a 1/5th scaled vehicle performed the same vehicle maneuvers but scaled to approximately 1/5th the distance and speed. When considering the scaling factor, the RTS's positional error at small scale was over 5 times higher on average than in the full-scale trials. Third, LiDAR object tracks were generated for the small-scale trials using a second EKF implementation and then combined with the radar objects in a high-level track fusion algorithm. The fused tracks demonstrated a 30% increase in positional accuracy for a majority of the small-scale trials when compared to tracks using just the radar or just the LiDAR. The proposed track fuser could allow scaled test beds to incorporate automotive radars into their sensor suites more effectively by augmenting the radar output with LiDAR, overcoming the resolution issues that afflict radar when operating at small scale.

Acknowledgements

The research team would like to thank all the colleagues at Continental Automotive and the Virginia Tech Transportation Institute who aided in this project. Their support was instrumental in the development and completion of this project. The research team would also like to thank the anonymous reviewers who provided constructive feedback on earlier versions of this report.

This project was funded by the Safety through Disruption (Safe-D) National University Transportation Center, a grant from the U.S. Department of Transportation – Office of the Assistant Secretary for Research and Technology, University Transportation Centers Program.

Table of Contents

List of Figures

List of Tables

Introduction

In the last decade, large improvements in advanced driver assistance systems (ADAS) have increased the possibility of fully automated systems one day replacing most human driving. These systems have the potential to substantially mitigate the over 30,000 driving fatalities and over 3 million injuries that occur each year. These benefits have prompted every major automotive company, many tier 1 suppliers, and several specialized start-ups to invest in automated driving solutions [1]. Waymo, Tesla, Aptiv, NVIDIA, Continental Automotive, Ford, and GM, among others, already invest hundreds of millions of dollars into the research and development of highly automated driving (HAD) systems to reduce the deadly consequences of human behavior behind the wheel.

Despite these large investments, safety is still a concern when testing highly autonomous vehicles on public streets and highways. Engineers need to test and validate all autonomous vehicle systems under development robustly to ensure safety. This includes completing model-in-the-loop, software-in-the-loop (SIL), and hardware-in-the-loop (HIL) testing before moving on to controlled full-scale testing. Therefore, a full timeline of vehicle design, testing, and validation can take years before completion, requiring large investments to sustain a complicated product development life cycle.

Scaled test beds (STBs) can enhance the testing capabilities of automated driving functions and reduce the required investments. Introduced two decades ago as a bridge between SIL testing and full-scale system testing, STBs have been commonly used to test vehicle dynamics [2,3], controls [4], and advanced braking systems [5] with researchers taking a large amount of care, in some studies, to ensure vehicle similitude [6]. While SIL testing often focuses more on virtual, subsystem tests, STBs offer concrete visualizations of system performance and may expose weaknesses in the system under design related to real-world, physical limitations. More recently, scaled vehicle systems have advanced along with ADAS technologies, supporting research in efficient vehicle platooning [7], camera object detection [8], trajectory tracking [9], and vehicle-to-vehicle communication [7]. HIL testing often accompanies many of these STBs, and HAD developers can use STBs for real-world visualizations of driving functions and systems under design after SIL testing is complete. The more each HAD system on the STB parallels the system on the full-scale vehicle, the more quickly developers can identify safety concerns and validate driving functions in inexpensive, low-risk environments.

Consequently, STBs have proven useful for vehicle and safety testing, but apart from a few targeted studies [10,11], researchers have largely ignored two of the main perception sensors utilized in HAD driving: LiDAR and radar. Radar (radio detection and ranging) sensors detect objects in the field of view (FOV) with three attributes—range, angle, and range rate—based on radio wave emissivity and the Doppler effect. LiDARs (light detection and ranging) build point clouds of the surrounding environment with position, depth, and intensity based on concurrently layered laser scans. Each LiDAR scan can easily contain 10,000 spatial points, but, currently, most LiDARs used in HAD systems do not associate velocity with any returned point the way radar sensors do.

Reluctance to include perception sensors on STBs has been generally warranted, particularly for radar, since data resolution deteriorates as the scale of the sensing environment decreases. Using automotive production radars is ideal in these model systems to allow for comparable testing, but placing

an STB in a scaled environment (i.e., 1/5) can increase radar noise to a point where object tracking degrades substantially. While radars operating at full scale have multiple detection zones per radar cycle on a single vehicle (i.e., wheels, bumpers), radar returns may originate from only one area of the vehicle at small scale due to decreased angle resolution. Without multiple returns from the object, radar tracking software (RTS) cannot accurately model the size of a tracked vehicle. The RTS is thereby forced to model the radar return as a moving particle with an unknown size. Thus, barring any enhancement or augmentation of the sensor data, STBs may operate within large limitations regarding radar object tracking. To ensure robust STBs capable of testing HAD functions, these limitations on radar data must be addressed. Otherwise, there is a risk of the STB diverging from their comparable full-scale HAD system, particularly as vehicle technologies rely more on the detection, classification, and tracking of objects from perception sensors.

Background

This collaboration between the VTTI InternHUB and Continental Automotive aimed to generate a fusion technique that used radar and LiDAR sensor inputs to improve radar-tracking performance in an STB. Due to resolution issues encountered when using radars in small-scale environments, there can be fewer radar points returned per object, often resulting in less accurate tracking. This is particularly evident with vehicles and smaller objects; as they are physically scaled down in the STB environment, the reflection zones on these objects can shrink, and the number of points detected may be limited by the angular resolution of the radar sensor. Other larger roadway objects, such as bridges, overhead signs, and guardrails, do not suffer as much with detection resolution in the STB environment due to their size.

One working approach to this issue is to modify the underlying tracking algorithm to perform optimally in the scaled environment, such as changing the underlying tracking algorithms or adjusting angular resolution parameters if possible. However, this would gravely decrease the value of the small-scale tests as the algorithm would significantly change between small-scale testing and the implementation of its full-scale counterpart. This manipulation would make test results difficult, if not impossible, to compare. The proposed solution was to augment the traditional radar data returns in the small-scale environment with LiDAR. These sensors, while expensive and susceptible to noise in some environments, have much higher spatial resolutions than typical automotive radar sensors can achieve. Therefore, the LiDAR point cloud information could be used, through a sensor fusion approach, to augment depleted radar returns from an object in a small-scaled environment. A track-to-track fusion technique could then be used between the two sensors to help the radar better estimate the position and trajectory of the object or vehicle being tracked within the small-scale environment.

In this investigation, the research team assessed software techniques from four fields of study to generate a fusion technique using radar and LiDAR sensor inputs: (1) Kalman filtering for automotive radars, (2) point cloud filtering with LiDARs, (3) sensor fusion techniques, and (4) small-scaled vehicle implementations. Each of these fields has been the subject of much recent attention as automated driving functions have holistically improved, but there is a large gap in the research regarding the use of software solutions developed at full-scale to function with STBs. This study sought to offer a potential solution to bridge that gap.

The initial step was to review previous work in these fields: radar sensors and Kalman filtering; LiDAR point cloud filtering; sensor fusion; and STBs. The team used the knowledge obtained from this systematic review to develop an effective fusion approach and guide the acquisition of empirical data. In turn, the empirical data was used to (1) quantify the extent to which a reduction in environment scale reduced the accuracy and resolution in radar returns, and (2) quantify the extent, if any, to which the proposed fusion approach could generate trajectory data properly simulating full-scale vehicle returns within the scaled environment.

Radar Sensors and Kalman Filtering

Radar detects objects within its FOV by transmitting and receiving electromagnetic waves. A radar can report each object's position, radial velocity relative to the sensor, and an estimated radar-cross sectional (RCS) area. Automotive radars operate in two main frequency bands, 24 to 26 GHz and 77 to 81 GHz, corresponding to approximately 12 mm and 4 mm wavelengths, respectively. Each radar cycle results in a list of detection points that the radar can consolidate into data return values (i.e., radar "clusters"). In automotive radar, each cluster has four associated characteristics: lateral and longitudinal distance from the sensor[1], velocity radial to the sensor, and RCS. A typical radar cycle can detect anywhere between 10 and 75 clusters depending on the radar's software limits and the complexity of the environment [12]. Reported statistics of a common automotive-grade radar (Continental ARS430) can be found in Table 1.

Table 1. ARS408 Specifications

Measuring Performance	Comment	To natural targets (non-reflector targets
Distance range		0.20 ...250 m far range, 0.20...70m/100m@0…±45° near range and 0.20…20m@±60° near range
Resolution distance measuring	point targets, not tracking	Up to 1.79 m far range, 0.39 m near range
Accuracy distance measuring	point targets, not tracking	±0.40 m far range, ±0.10 m near range
Azimuth angle augmentation	(FOV)	-9.0°...+9.0° far range, -60°...+60° near range
Elevation angle augmentation	(FOV)	14° far range, 20° near range
Azimuth beam width (3 dB)		2.2° far range, 4.4°@0° / 6.2°@±45° / 17°@±60° near range
Resolution azimuth angle	point targets, not tracking	31.6° far range, 3.2°@0° / 4.5°@±45° / 12.3°@±60° near range
Accuracy azimuth angle	point targets, not tracking	±0.1° far range, ±0.3°@0°/ ±1°@±45°/ ±5°@±60°near range

[1] However, note that radars do not inherently report clusters in the typical *x-y* coordinate scheme, but instead do so in the polar coordinate system. A non-linear coordinate transformation is required to convert to Cartesian coordinates.

Measuring Performance	Comment	To natural targets (non-reflector targets
Velocity range		-400 km/h...+200 km/h
Velocity resolution	target separation ability	0.37 km/h far field, 0.43 km/h near range
Velocity accuracy	point targets	±0.1 km/h
Cycle time		app. 72 ms near and far measurement
Antenna channels /-principle	microstripe	4TX/2x6RX = 24 channels = 2TX/6RX far - 2TX/6RX near / Digital Beam Forming

Radars excel at collecting real-time data from a dynamic environment but need filtering and estimation techniques to translate radar clusters into object tracks over time, which is typical for any active perception sensor. Optimal state estimation, which tracking filters generally attempt, is a classic control theory problem with abundant literature. Early research in this field combined statistics to address one of the most important problems of the time: accurately estimating the state of a process in the presence of random noise, which was particularly important as electrical sensors became increasingly popular. This is the problem area where, in 1960, R. E. Kalman published his seminal work "A New Approach to Linear Filtering and Prediction Problems" and first introduced the famous Kalman filter (KF) [13].

The basis for the filter is a linear predict-and-update model able to recursively include all past data in the state estimation without needing to continually store that data. This greatly simplified the mathematics and computational requirements for filtering problems—something particularly relevant for processing within embedded systems. At the heart of the KF's predict-and-update model are Bayesian statistics, specifically Bayes theorem. Summarized in this context, the theorem concludes that given some knowledge about the current state of a system or process (expressed as a probability or underlying variance), it is more accurate to first, predict the state of the system in the future based on the current estimate and second, update that predicted state based on incoming noisy measurements; then, all that is required is to update the state based on the measurements alone [14]. In this sense, no matter how uncertain one can be about a measurement's true accuracy, the information it provides still contributes to improving the overall tracking accuracy [15]. In a KF, the noise in the system is often modeled as a Gaussian (normal) distribution with a mean value of zero and some variance value, σ^2. Due to this noise, the accuracy of the underlying system cannot be fully trusted, and the reported measurement is considered a combination of the true value skewed by some amount of unknown random noise. Therefore, the true state value is modeled as a Gaussian distribution over the state space with inherent standard deviation [13]. By modeling the noise, the system estimate provided by a KF can, given the known added biases of the sensor(s), seek the true state value instead of erratically jumping in a non-smoothed or "unfiltered" way between the reported values.

Consequently, there are two major sets of linear control equations included in a KF. One pertains to the prediction update (a priori estimate) and the other to the measurement update (a posteriori estimate) [14]. The prediction step requires three basic elements: (1) the magnitude of the time step used to predict

the future state, (2) the process noise expressed in variances and covariances (i.e., uncertainty values in the model predictions), and (3) the current covariances of the system. Only two calculations are needed. One calculation predicts the state at the future time step, and one updates the covariances of the system given the process noise. Similarly, the measurement update requires three elements: (1) a matrix to map the measurement to the state space, (2) the sensor noise expressed in variances and covariances, and (3) the measurement values themselves. With these equations, a linear KF used to track position and velocity can be implemented. Additionally, the linear KF model can be extended to track non-movement with a few modifications of the predict-and-update equations, resulting in the common extended KF (EKF) approach upon which many industry trackers heavily rely when tracking maneuvering objects. See Appendix A for more information regarding the KF and EKF processes.

Beyond the EKF modifications, a multi-object tracker must deal with the data association problem. Given that many perception sensors return multiple values per cycle (e.g., radar), these data returns need to be associated with objects in the environment before they can be passed into a KF. If there are multiple objects within the sensing area, the data from one object should not be conflated with the data return from another object because this would clearly result in erroneous and unusable tracking solutions from the KF.

The first step in successful data association is to create multi-track capability within the software program tracking the data measurements. In this manner, a set number of environmental objects can be observed and tracked over time, which is imperative for automotive or robotic uses in uncontrolled environments. Multi-track KFs require the addition of an object list—each running their own KF—and some version of a data association technique to match the data returns to specific objects (or create new ones where necessary). One adaption is the JPDA (joint probabilistic data association) technique. The JPDA technique weights measurements by how likely they are to belong to a certain track instead of outright assigning a measurement to only one track. This allows the JPDA to perform better than many other data association schemes when assigning measurements to tracked objects, particularly when those objects pass within proximity of each other, albeit at some cost in calculation time. Appendix B provides description of how the JPDA solves a typical data association problem.

LiDAR and Object Tracking

LiDAR is a time-of-flight sensor technique that uses pulsing infrared lasers (~900 nm wavelength) to map the surrounding environment. With cycle times as high as 20 Hz, modern LiDAR systems can report millions of data points per second. Each of those data points has four characteristics: distance values in the longitudinal (x), lateral (y), and vertical (z) directions, and an intensity value of the return. While both radars and LiDARs can detect obstacle position, LiDARs can sense the world at a much higher resolution than radars can.

In this investigation, a 16-channel Velodyne PUCK LiDAR was used to support the sensor fusion task. This LiDAR unit features a 360° horizontal FOV, 16 laser channels, a range of 100 meters, +/- 3 cm accuracy, 30° vertical FOV, horizontal angle resolution of 0.1° to 0.4°, operating speed of 5 to 20 Hz, 905 nm operating wavelength, microsecond data time-stamping, and a collection rate of up to 300,000 points per second [16].

Substantial point cloud processing and filtering is needed to convert LiDAR point cloud data into useful information. This includes estimating the drivable road area, identifying traffic objects, classifying other environmental objects (e.g., buildings, fences, guardrails, lane lines, stop signs, trees), and performing object tracking. Effective LiDAR systems incorporate multiple techniques to process a single point cloud. An overview of a typical automotive LiDAR system includes algorithms designed to:

1. Down-sample the original point cloud into clusters, voxels [17], occupancy grids, or octrees [18] to reduce computational processing time;
2. Estimate the drivable area using RANSAC (i.e., random sample and consensus [19]; see Appendix C) or another ground plane estimator;
3. Convert clusters to objects and systematically assign data points to those objects using a density or nearest neighbor approach;
4. Track objects over time using KF or other similar techniques to identify dynamic and static obstacle tracks; and
5. Classify an object based on pre-specified parameters relative to the application and environment.

Down-Sample Process

For a LiDAR system to achieve near real-time system requirements, the hundreds of thousands of points present in a point cloud need to be greatly reduced. One way to achieve this is to down-sample the relatively high-density data returns into fewer points through voxel grids and clusters.

First, the *voxel approach* involves segmenting the three-dimensional space of the point cloud into small cubic elements. All points that lie within that volume of area are grouped, and the average position or the point closest to the average position is often returned as the single data point for that voxel [20]. The data within the point cloud is then reduced to the number of voxels, which are often several orders of magnitude less than the original number of points. Second, based on the voxels created in the first step, clusters can be formed. *Clustering methods* are ubiquitous and come in many different implementations, but in general they rely either on a specified Euclidean or statistical distance to determine if one return is part of the same object or cluster as another return in the point cloud [21]. This process usually results in the grouping of 100 clusters or less per data frame. The centroids for those clusters can be stored and tracked more efficiently than thousands of individual data points. Often, a certain defined threshold is set or dynamically maintained to cap the number of clusters, gate the number of points per cluster, and constrain the spacing between adjacent clusters.

Ground Plane Estimation

Because automotive LiDARs are deployed in areas where driving is possible, a large percentage of the data points within the point cloud originate from the road surface. The correct identification of this "ground" plane benefits the system in two major ways. First, the data points that constitute the ground plane can be removed from the point cloud once identified, and the resultant reduction in data points increases computational efficiency. Second, the height of the ground plane can be used as a gate to identify other environmental objects. A direct comparison between the height of the estimated ground plane and

the height of a cluster can therefore be used to determine certain classification levels for an object in the surrounding environment [22].

The simplest ground plane removal method involves cutting off all points below a certain height value, usually informed by the placement of the sensor itself. This is not, however, a dynamic approach, and it only works when the road plane is relatively flat. In real-world applications, more robust estimators are needed to account for changing road plane angles.

One such option is to use RANSAC (random sample and consensus) to identify a general ground plane. A simplified outline for RANSAC entails (1) identifying random planes within the point cloud data, and (2) assigning points as either outliers or inliers for that plane based on thresholds. The plane with the most inliers is most likely to be the ground plane. For cases where the ground plane changes slope across the FOV, multiple iterations of RANSAC can be used to identify the multiple ground planes.

Object Tracking

As previously noted in the discussion pertaining to radar, object tracking using time-discrete sensors is an optimal estimation and data association problem where object existence, object state, and sensor data are all commonly modeled as Gaussian distributions within the state space. Many of the same principles and solutions applicable to radars apply to LiDAR sensor systems as well. Like radar sensors, LiDAR sensor systems often incorporate KFs and data association techniques to accurately track objects within the sensing environment. The process and equations are the same, except that these KFs do not initially track velocity components of clusters as they first become visible. However, LiDAR KFs can derive an object's velocity over time as they move within the FOV over consecutive timesteps. Often, the inputs to the LiDAR object tracking algorithms are the centroids of the cluster, which are tracked over time.

Sensor Fusion

Perceptual sensors, such as radar, LiDAR, and camera vision, are rarely used alone to provide perception information for a vehicle. Generally, these sensors are combined within the same application to provide redundancy and robustness: the strengths of one sensor can help overcome the weaknesses of another sensor. Therefore, common automotive system architectures often fuse some combination of camera, radar, and LiDAR; any combination between two or more of these perception sensors can exist depending on the application and resource limitations.

Sensor fusion, in the context of environmental perception, implies combining the data streams or object outputs of multiple sensors to ultimately reduce the uncertainty about the objects and their current state. If two sensors perceive the same object in the same global space, the system can be more certain of its existence and can combine the data from each sensor to better track the object.

A common goal of sensor fusion in automotive applications is to support object detection and tracking tasks. There are two primary methods to achieve sensor fusion for this application: de-centralized and centralized fusion. *De-centralized fusion* focuses on the individual sensors themselves. Each sensor within the system architecture captures and pre-processes its own data. The tracks from all sensors are input into a fusion module that then combines them. Often, a second layer of tracking is applied to fuse

tracks to support global object tracking. The process is often referred to as track-to-track fusion (a.k.a., high-level fusion, object list level fusion). The only extra modules needed for high-level fusion are those to combine tracks after sensor processing, which often use commonplace data association techniques to manage the global list of objects [23]. Some drawbacks of track-to-track fusion stem from sensors providing data at asynchronous times (which can require extra processing downstream to align them), needing adequate processing and computing power for each sensor, maintaining the proper alignment and calibration of sensors, and implementing techniques to resolve duplicate detection for sensors with overlapping FOVs.

In contrast, *centralized fusion* uses all sensor data as input before any object detection and tracking is performed. Incorporating all data from the sensors before running detection and tracking algorithms ensures that the system has as much data as possible to make decisions. A small amount of pre-processing is usually done for some sensors (e.g., ground plane removal and general clustering for LiDARs), leaving only the most important features of each sensor's measurement frame. These features form the input from which a central tracking algorithm can associate and develop tracks. As a result, centralized fusion is often referred to as low-level or feature-level fusion [23]. Centralized fusion is much easier to implement with sensors that have similar output structures, such as radar and LiDAR. However, large system communication bandwidth is required, and significant software modifications are generally required for any unique combination of sensors for low-level fusion. Hybrid approaches that combine features of both high-level and low-level fusion have also been developed [24].

Scaled Test Beds

For a full-sized vehicle, the methods presented in the previous sections offer enough background to initiate advanced development of a fusion system in support of autonomous driving systems. However, our intention in the current investigation was to implement a fusion solution for a small-scaled vehicle in a small-scaled environment. This STB vehicle and its scaled environment present novel challenges for fusion methods, especially those involving radar.

Equipping STBs with the same sensors and controls as full-sized vehicles is the first step to achieving vehicle similitude. Most small-scaled vehicles adapt remote-control car platforms at either 1/4th, 1/5th, or 1/10th the actual size because powertrain components at that scale are readily available. Required sensors and computer hardware are then added to the existing platform. STBs generally feature inertial measurement units, accelerometers, gyroscopes, a central electric motor with differential or individual motors for each wheel, wheel speed sensors, steering servo motors, on-board central processing units (CPUs) for motor and steering control, and ethernet or Wi-Fi connection ports. Some sophisticated STBs may also feature additional ADAS sensors such as laser scanners, LiDAR, stereo vision hardware, cameras, ultrasound sensors, GPS modules, and the additional microcontrollers and computing power required to deploy these sensors.

In general, STBs offer several advantages for engineers and developers. First, *expense*: STBs are less expensive to create than a full-sized prototype vehicle. Second, *time*: STBs require less time to build; tests using STBs are easier and faster to conduct than those on full-sized vehicles. Third, *risk*: STBs reduce the physical risk for all people involved in the testing, which ultimately reduces business risk. Fourth,

reality: STBs can more easily expose a vehicle's physical limitations and more intuitively demonstrate vehicle functionality than software simulations. Overall, STBs are one tool in a large toolchain available for systems development engineers, and they efficiently bridge virtual testing and full-scale implementation. When used in combination with other approaches, STBs can accelerate the development of driving functions, especially when used for physical "smoke" tests of certain ADAS aspects [25-29].

Despite the impressive configurations of previous STBs in research and commercial spaces over the past two decades, no STB was discovered that incorporated automotive-grade radar into its functionality, as was required by the current investigation. Thus, all currently available STBs lag in usefulness when modeling ADAS functions with radar. This represents a sizeable problem because many ADASs and other HAD systems already rely heavily on radar object detection. Without including radar, STBs cannot be fully utilized by engineers and developers of automated driving algorithms.

The solution proposed in the current investigation directly addressed this gap in functionality for STBs. The fusion method between LiDAR and radar used the high spatial resolution of the former sensor to inform the latter sensor about potential object locations that the radar cannot reliably detect at a small scale. This fusion process thus allows an STB to fully implement a production radar for general purpose vehicle tracking within a scaled environment.

Research Questions

The gaps identified via the literature review suggest several important research areas related to the effectiveness of radar sensors in scaled environments and the ability of other sensors (e.g., LiDAR) to supplement any shortcomings in radar performance. Based on these gaps, this investigation intended to answer the following questions regarding scaled vehicle tracking and sensor fusion:

- Compared to full-scale radar tracking, how much does the positional accuracy of the radar tracking software (RTS) diminish when tracking objects in a 1/5th scaled environment?
- Can the RTS tracking accuracy in a 1/5th scaled environment be quantified using standard error metrics?
- Using a LiDAR sensor and a custom track-to-track sensor fusion technique, can the tracking accuracy in a 1/5th scaled environment be improved?
- If present, can the accuracy improvement in the 1/5th scaled environment match the accuracy obtained from the radar at full scale?

Methods

To answer these research questions, five elements needed to be implemented. The first was the radar and associated tracking software; the second, LiDAR clustering of point cloud data; the third, a software fusion technique; the fourth, a ground truth measurement; and the fifth, a data recording system with sensor implementation for full-sized and STB vehicle trial runs.

Sensor Implementations and Trial Configurations

The first step in developing tracking filters was to implement a data collection system in the Robot Operating System (ROS) to capture radar data returns, LiDAR point clouds, and DGPS ground truth values for each vehicle under surveillance (VUS).

ROS can inherently record and time-stamp all messages passed within the system, so the only input information required was the sensor configurations. Using an ROS driver developed for the Continental ARS430 series radar running at 14 Hz, the Velodyne ROS driver provided for the Velodyne PUCK LiDAR set to run at 10 Hz, custom DGPS ROS drivers, and an NVIDIA Nano CPU running Linux tasked with collecting data for the trial runs. The radar and LiDAR were statically mounted 10 and 20 cm, respectively, off the ground, similar to their height when mounted on a small-scale vehicle.

Each trial run included a single VUS running through vehicle maneuvers within the radar and LiDAR's FOV in a cluttered, but mostly static, environment. Curbs, light posts, fences, and other stationary vehicles were present, but the VUS was the only moving object within the FOV. There were three main types of vehicle maneuvers: driving directly toward or away from the radar sensor, driving diagonally across the radar's FOV, and performing S-turns within the radar's FOV. The first two sets of vehicle maneuvers represented normal driving conditions that the perception sensors and their trackers should easily be able to handle, while the third type of vehicle maneuvers, the S-turns, should challenge the trackers because of the large non-linear movement. The "S-turn" trials were intended to represent driving conditions that would be less apparent within the FOV of the radar or LiDAR sensors than the "Diagonal" and "Straight" trials, particularly if these sensors were installed on a vehicle driving in real-world conditions. More extreme vehicle maneuvers, such as "figure 8s" or circles, were not considered because these maneuvers would rarely be present in real-world driving conditions. Appendix D shows sensor FOVs for both radar and LiDAR as well as the three test trial configurations and the STB used.

All maneuvers were repeated for the small-scale and full-scale vehicles. Because the 1/5th scaled vehicle was built to model a low-speed shuttle, the velocities of each trial were determined by typical operating speeds of those shuttles, usually 10 to 15 mph. Thus, the full-scale vehicle traveled at speeds between 10 and 15 mph (4.5–6.7 m/s) while the small-scale vehicle traveled at speeds between 2 and 4.5 mph (0.89–2 m/s). To accurately scale the small vehicle maneuvers, particularly during the S-turn trials, tape was placed at roadway locations where the vehicle should start to change its direction from left-to-right or right-to-left. However, exact positional scaling was not possible due to the use of a manual driving approach—thus, the small-scale S-turn trials extended to positions slightly outside the full-scale S-turn trial when accounting for scale. This study focused on modeling and tracking only a small-scale VUS. Other small-scale objects, such as bridges or overhead signs, were not included in the environment when conducting trials.

All tests were completed during the day with partly cloudy weather conditions and no rain or water on the ground. No effects of sunlight or other environmental conditions were seen in the collected data from any sensor. Each run typically lasted for 20 to 30 seconds. Tests of this duration usually generated 280 to 430 radar frames and 200 to 300 LiDAR point clouds.

ArduSimple DGPS boards provided ground truth values for each trial run. One of the boards, configured as a DGPS base station through a multi-hour survey, sent GPS timing corrections to the other

board, configured as a rover, and installed on the VUS. The ArduSimple system was able to calculate highly accurate positions, with error measurements no larger than 12 cm at any given time. For each trial run, ROS recordings for all the sensors were time-stamped and stored in .bag files for easy playback.

Radar and LiDAR Tracking

RTS was developed and implemented in the MATLAB environment as part of this investigation. The RTS centered on the EKF and the principles previously summarized. In general, the RTS needed to obtain the radar data returns, filter those returns based on the sensor speed to identify moving objects, cluster returns from the same object, perform object tracking, and save the resultant track information for the duration of each trial. A JPDA point tracker was selected for tracking maneuvering objects within MATLAB as pilot testing suggested it to be the most consistent, noise-resistant, and tunable tracker out of the alternatives considered.

To minimize radar clutter within the environment, radar returns were filtered based on speed. Because the radar was statically mounted and the tracker was only interested in moving objects, any measurement that had a range-rate value of 0, or very close to 0, most likely originated from a static object in the environment and was discarded. Usually, when a radar is mounted to a moving vehicle, all data returns from a cycle need to be transformed into the vehicle frame with velocity components in the x and y directions (where they can then be similarly filtered based on ego vehicle speed). Here, that step was not needed.

After identifying dynamic returns, the radar data was clustered based on a threshold for the density of surrounding points (points needed to be close in distance and in range rate to be considered part of the same cluster). The representative centroid along with cluster measurement noise was then passed to the JPDA to start tracking. The JPDA filter reported object tracks with an EKF in the tracking frame: x and y dimensions with velocity components and the radar sensor at the origin facing along the x-axis. The EKF itself had several important attributes that were defined and tuned based on pilot testing before any tracking started: the process noise, the state transition model, and the mapping (measurement) function.

As with tracking for radar measurements, important objects needed to be identified in the LiDAR point clouds. To first down-sample the point cloud, voxels cubes with sides of 5 cm were used. RANSAC was then used to identify and remove the ground plane. The voxels remaining after ground plane removal progressed to the object clustering algorithm. The clustering algorithm followed a similar process to the approach described for the radar data. Here, however, a Euclidean distance threshold of 10 cm was used to form clusters. With both the centroids and associated points identified, tracking was performed downstream on the LiDAR clusters. This required a separate JPDA filter for the LiDAR clusters to generate tracks similar to the RTS. With tracks available for both sensors, the next step was to fuse those track outputs.

Sensor Fusion

High-level, or track-to-track, fusion was selected to complete sensor fusion between the radar and the LiDAR tracks. Given multiple trackers outputting confirmed object tracks at similar or varying time stamps, the track fuser maintained a list of central tracks based on the states and covariances of the sensor

tracks. The track fuser largely behaved as its own tracking filter, predicting central tracks forward with a process model, assigning measurements to tracks (here the measurements were the sensor tracks), and updating covariances. Data assignment functions, process noise values, confirmation and deletion thresholds, and state transition functions were all supplied to the track fuser in a manner similar to the previous JPDA EKF filters, although the track fuser used a simpler data association technique.

The output of the track fuser, along with the independent outputs of tracks from both the radar and LiDAR sensors, could then be directly compared to each other and to the DGPS ground truth values obtained during the data recordings.

Error Metrics

Several error metrics were used to compare accuracy between the ground truth and sensor/fusion tracks. These metrics included the mean squared error (MSE), the root-mean squared error (RMSE), the mean absolute error (MAE), linear correlations, the squared error (SE), and the estimation error squared (EES) at a given time step. Interpolated ground truth values from the DGPS points were used to compare the track positions at each of the time steps and calculate the specified error metric. Appendix E provides more detail for each error metric.

The MAE, RMSE, MSE, EES, SE, and linear correlation values all characterized a sensor tracking accuracy over time. Moreover, these values were calculated for the full-scale radar trial runs and then compared to the same metrics calculated for the radar sensor in the 1/5th environment to measure the tracking degradation. Comparisons across experimental conditions leveraged "increase factors," which are simply the final accuracy (i.e., after an experimental manipulation) divided by the original accuracy (i.e., before an experimental manipulation). The resultant value from the ratio reveals by what factor the value has increased. For example, if the RMSE of a track increased from 0.2 in a full-scale test to 0.6 in a small-scale test, the RMSE has increased by 3 times, and the increase factor is 3.

These metrics were also calculated across track fusion method to quantify the tracking accuracy increase compared to using only the radar or only the LiDAR sensor. Percent change between the error metrics of the two input tracks and the error metrics of the fusion tracks was also calculated to show the tracking accuracy increase of using fusion. Similarly, the final fusion track metrics were compared to the original full-scale radar track metrics to quantify the track fusion method's usefulness at small scale.

Error metrics were calculated for each trial run. The full-scale trial runs featured radar tracks compared to DGPS ground truth values for all three vehicle maneuvers. The small-scale trials featured radar tracks, LiDAR tracks, and fusion tracks for similar vehicle maneuvers, also compared to DGPS ground truth values. Recall that, represented in the 1/5th environment, vehicle size, vehicle speed, and driving distance were all scaled to values at approximately 1/5th the magnitude of their full-sized counterparts.

Each full-scale and small-scale trial produced video recordings of the sensor data and tracks plotted against the DGPS ground truth values at each time step for visual reference. For the sensor track data, there were three plots generated: an overall x versus y graph, an x versus time graph, and a y versus time graph. The error metrics (i.e., MSE, RMSE, MAE, linear correlation) were also calculated for each trial

run for both the *x* and *y* values of the track. Additionally, graphs depicting the EES and SE as a function of time were generated.

Finally, there are two important caveats related to this analysis. First, the ArduSimple board self-configured for this project provides accurate global positioning but does not provide any speed data. Therefore, all comparison metrics could only be calculated from positions, not velocities. Second, the small-scale environment tracking had to be directly compared to full-scale tracking measurements. Therefore, for compatibility, the track errors in the 1/5th environment were scaled by a factor of 5 before any error metrics were calculated. Thus, for example, errors of 1 m in the 1/5th environment would be equivalent to 5 m at full-scale.

Results

Full-Scale Tests

The track error metrics for all three trials (Straight, Diagonal, and S-turn) were calculated for the full-scale tests using the DGPS sensor as a ground truth comparison. The MSE, RMSE, MAE, and correlation values are shown for each trial (Table 2). The radar tracking results overlayed on the ground truth position appear in Figure 1 for the S-turn trial.

Table 2. Error Metrics for Full-Sized Vehicle Radar Tracking, Straight, Diagonal, and S-turn Trials

	Full-Scale Vehicle Straight Trial Run x	*Full-Scale Vehicle Straight Trial Run y*	*Full-Scale Vehicle Diagonal Trial Run x*	*Full-Scale Vehicle Diagonal Trial Run y*	*Full-Scale Vehicle S-turns Trial Run x*	*Full-Scale Vehicle S-turns Trial Run y*
MSE (m^2)	0.5078	0.2161	0.3726	0.1757	3.2372	0.7122
RMSE (m)	0.7126	0.4648	0.6104	0.4192	1.7992	0.8439
MAE	0.5275	0.3378	0.5051	0.3369	1.5917	0.6988
Linear Correlation	0.9987	0.9987	0.9990	0.9990	0.9987	0.9987

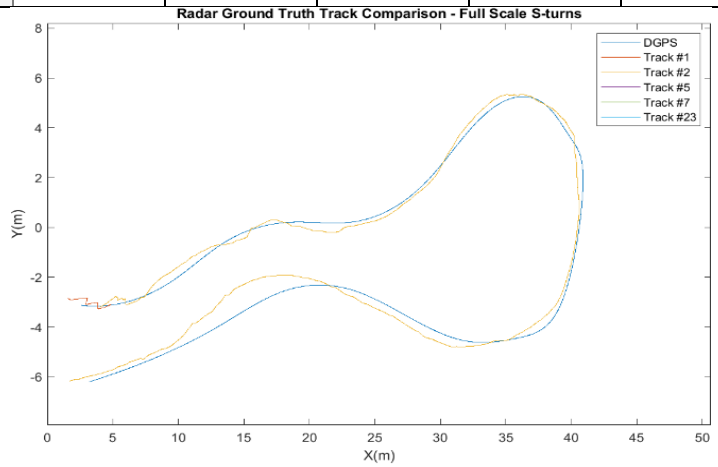

Figure 1. Graph. Radar track position and ground truth overlay, S-turn full-scale trial.

Small-Scale Tests

The same three test runs (Straight, Diagonal, S-turn) were conducted at small scale and the error analysis recorded. In these trials, there were MSE, RMSE, MAE, and linear correlation values calculated for three different sensor modalities: radar only, LiDAR only, and the fused tracks. As an example of track outputs, Figure 2 shows the radar, LiDAR, and fused object tracks for the small-scale S-turn trial. The error metric tables associated with each run as well as the track position and ground truth overlays broken into the *x* and *y* positions can be found in Appendix F.

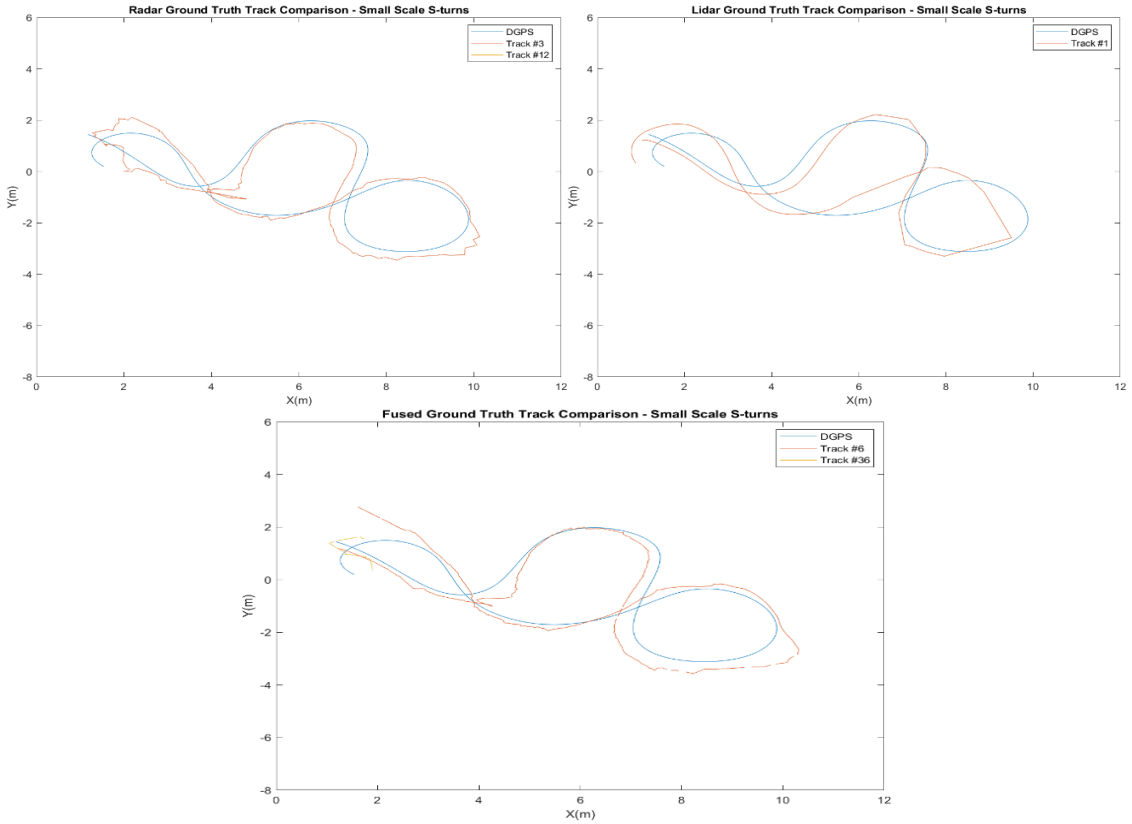

Figure 2. Graph. Radar, LiDAR, and fused track position overlays with ground truth, small scale S-turn trial.

Increase factors were also calculated to compare the radar track error metrics from the full-scale and small-scale trials (Table 3). The same increase factors were computed to compare the radar performance at full scale and the LiDAR performance at small scale (Table 4).

Comparison between the fusion technique tracking metrics and the sensor tracks metrics was aided by calculating the percent change in the tracking error metrics (Table 5). The table breaks down into three main sections for each small-scale test and displays the percent change in the specified error metric cell for the two dimensions assessed within the radar and LiDAR sensor tracks. A positive percent change implied that the fusion approach had the smaller error when compared to the ground truth compared to the sensor track itself. Red shading has been added in the table to highlight those that had a negative percent change.

Table 3. Increase Factors Between Full-Scale Radar Tracks and Small-Scale Radar Tracks

	Straight: *x* dimension	Straight: *y* dimension	Diagonal: *x* dimension	Diagonal: *y* dimension	S-turn: *x* dimension	S-turn: *y* dimension
MSE (m^2)	11.2	2.5	40.9	65.6	0.6	2.9
RMSE (m)	3.4	1.8	6.4	8.8	0.8	1.7
MAE (m)	4.2	1.7	6.4	7.0	0.7	1.7

Table 4. Increase Factor Between the Full-Scale Radar Tracks and Small-Scale LiDAR Tracks

	Straight: *x* dimension	Straight: *y* dimension	Diagonal: *x* dimension	Diagonal: *y* dimension	S-turn: *x* dimension	S-turn: *y* dimension
MSE (m^2)	18.1	5.7	49.2	22.2	1.8	14.0
RMSE (m)	4.3	2.4	7.0	4.7	1.3	3.7
MAE (m)	5.1	3.0	7.2	4.5	1.2	3.8

Table 5. Percent Change Decrease in Tracking Error Metric for the Fusion Method

	Small-Scale Straight: Radar Tracks *x*	Small-Scale Straight: Radar Tracks *y*	Small-Scale Straight: LiDAR Tracks *x*	Small-Scale Straight: LiDAR Tracks *y*	Small-Scale Diagonal: Radar Tracks *x*	Small-Scale Diagonal: Radar Tracks *y*	Small-Scale Diagonal: LiDAR Tracks *x*	Small-Scale Diagonal: LiDAR Tracks *y*	Small-Scale S-turn: Radar Tracks *x*	Small-Scale S-turn: Radar Tracks *y*	Small-Scale S-turn: LiDAR Tracks *x*	Small-Scale S-turn: LiDAR Tracks *y*
MSE (m^2)	36%	31%	59%	71%	36%	42%	47%	-72%	40%	-30%	95%	73%
RMSE (m)	20%	17%	36%	46%	20%	24%	27%	-31%	22%	-14%	53%	48%
MAE (m)	41%	32%	5%	62%	39%	25%	46%	-17%	60%	1%	64%	53%

The fusion track metrics were also compared to the original track errors from the full-scale trials. Again, increase factors were used to compare the radar track metrics at full scale for each trial to the fusion track metrics from the paired small-scale trials (Table 6). The table breaks down into three main sections of four columns for each trial. On the left side of each section are two columns that represent the error metric increase factor between the radar-only small-scale tracking scenario and the paired full-scale radar tracking trial (see Table 3). For direct comparison, the right two columns in each section are the increase factor when using the track fusion metrics. Instances when the fusion technique increased track error metrics when compared to only the radar at small scale have been marked in red.

Table 6. Track Metric Error Increase Factors Between the Radar Tracks at Full Scale and the Small-Scale Tracks from the Radar Only and Fusion Technique (Separated Between the *x* and *y* Dimensions)

	Straight Small-Scale Radar only *x*	Straight Small-Scale Radar only *y*	Straight Small-Scale Fusion Tracks *x*	Straight Small-Scale Fusion Tracks *y*	Diagonal Small-Scale Radar only *x*	Diagonal Small-Scale Radar only *y*	Diagonal Small-Scale Fusion Tracks *x*	Diagonal Small-Scale Fusion Tracks *y*	S-turn Small-Scale Radar only *x*	S-turn Small-Scale Radar only *y*	S-turn Small-Scale Fusion Tracks *x*	S-turn Small-Scale Fusion Tracks *y*
MSE (m^2)	11.2	2.5	7.6	1.7	40.9	65.6	26.3	38.2	0.6	2.9	0.4	3.8
RMSE (m)	3.4	1.8	2.7	1.3	6.4	8.8	5.1	6.2	0.8	1.7	0.6	2.0
MAE (m)	4.2	1.7	2.5	1.1	6.4	7.0	3.9	5.3	0.7	1.7	0.4	1.6

Compared to the full-scale tracking, the radar tracking at small scale saw higher metrics, including the EES and SE values, across all runs except the S-turn trial (Table 5). There was a clear increase in track errors for the *x* and *y* dimensions in the Straight and Diagonal small-scale trials, along with a slight increase in error in the *y* dimension of the S-turn small-scale trials. This indicates that overall tracking accuracy decreased when tracking in a small-scale environment using the radar sensor, even when the vehicle was performing simple maneuvers. The Diagonal trial posed problems for the radar and indicated substantial error increases. The *x*-dimension radar track correlation with ground truth was also the lowest for this trial. Compared to the LiDAR sensor at small scale, the radar followed the vehicle better during maneuvers, and even successfully followed the vehicle during the S-turns despite the high number of directional changes. On the other hand, the LiDAR track lagged behind the vehicle maneuvers several times, causing errors to increase during turning events. MSE, RMSE, and MAE metrics were all higher for the LiDAR for all runs, when compared to the small-scale radar tracks, except in the Diagonal trial, where the LiDAR was more accurate in the *y* dimension (Table 6).

Overall, the track fuser had the lowest error metric values when compared to the radar and LiDAR track metrics for all the small-scale trials. In all but five instances, the fusion method increased the tracking accuracy, often substantially, across the MSE, RMSE, and MAE metrics. When compared to the radar track alone, half the data metrics improved by 30% or more (Table F1). With the LiDAR, 13 out of the 18 conditions improved by more than 30% (Table F1). Out of 18 track metric increase factor comparisons, only twice did the fusion technique deliver overall track metrics that were worse than using only the radar to track the vehicle at small scale (Table F2).

Discussion

Full-Scale Tests

The RTS did well when tracking the full-sized vehicle, particularly during the Straight and Diagonal trials. For both the Straight and Diagonal runs, the MSE and RMSE were well below a value of 1, meaning the average residual of the errors was less than 0.5 m in either the *x* or *y* dimensions.

In contrast with the Straight and Diagonal trials, the S-turn trial resulted in substantially larger error metrics: up to 6 times higher in the *x* dimension and 3 times in the *y* dimension. It is likely that the KF used could not adapt to the non-linear movements inherent in the S-turns to maintain MSE, RMSE, and MAE values that were similar in magnitude to those observed for the Straight and Diagonal trials. The error metrics increased directly after the initiation of each turn, as the vehicle began to maneuver in the opposite direction. Most of the tracking error observed in the S-turn trials was due to the KF lagging in its adjustment to non-linear movements in the S-turns or to clustering errors, and not primarily due to tracking point drift. Despite the increase in error metrics, the tracker still generated useful and reasonable tracks through the S-turns. For example, the MAE in the *x*-dimension for the S-turn trial reached a maximum of 1.5 m, which is well below the length of a normal vehicle. Other metrics, such as the EES and SE, remained low for all the full-scale trials, including the S-turn.

Other sources of error during the full-scale trials included (1) multi-path radar reflections and (2) reduced data accuracy at the edge of the FOV. Multi-path radar reflections, or ghost reflections, consisted of radar returns that were seen primarily in positions directly behind the vehicle, possibly due to multi-path behavior. Additionally, the end of each full-scale trial always featured the vehicle driving out of the FOV of the radar (see Appendix D). Because the radar is less accurate in position and range rate values at these high angles, the tracker was often less accurate during these times.

Although both problems—radar reflections and reduced sensor accuracy at the edge of the FOV—were present in the collected data, the radar tracker was highly successful during the full-scale tests. This was even the case despite use of a static radar in all trials, which tends to be a more difficult tracking scenario than a dynamic radar situation due to a lack of range rate attributes for many returns in the radar FOV. Range rate attributes can be helpful during the clustering process.

Small-Scale Tests

Radar and LiDAR Error Metric Comparisons

Compared to the radar track metrics of the full-scale tests, there was a large increase in the scale-adjusted errors associated with the radar and LiDAR tracks in the small-scale test, particularly in the longitudinal (*x*) direction, which contained most of the trial travel distance. Although the tracker was able to follow the small-scale vehicle reasonably well in the Straight and S-turn trials, there were substantial track switching and position errors in the Diagonal trial. A track switch is always accompanied by a large increase in error metrics, indicating the tracker is struggling to maintain its track of the object. The increased errors are likely partially a result of increased incidence of radar reflections during the small-scale trials.

Unlike the full-scale tests, the small-scale S-turn trial produced error metrics comparable to the small-scale Straight trial. Thus, a prediction of poor performance in the small-scale S-turn trial, given the full-scale results, was not accurate and may be related to the slower speeds in the small-scale trials. The slower speeds may have allowed the radar tracker to adjust to the accelerations more accurately. There was, however, an associated increase in the covariance values, showcased in the EES graph for the radar sensor in the S-turn trials. Video and radar cluster analysis showed that the overall number of points detected by the radar from the VUS decreased greatly for the small-scale vehicle. Fewer points mean the

density-based clustering was more susceptible to errors and radar reflections, increasing error metrics at times, especially during the Diagonal run.

Another factor that likely contributed to the poor performance of the sensor was the reduced clustering range of the sensor, especially for the small-scale vehicle. At distances greater than 10 m for the small-scale trials, the small-scale vehicle could slip through the channels of the LiDAR, resulting in less accurate clustering. Using multiple LiDAR sensors or a higher-channel LiDAR could alleviate this problem in future similar efforts. In addition, the fact that the LiDAR cannot report range rate values could have resulted in delayed response to vehicle maneuvers and, ultimately, higher error metrics, particularly for the S-turn and Diagonal trials. Although less accurate than the radar, the LiDAR was still able to track the small-scale vehicle in all three tests using only basic clustering techniques.

Fusion Track Metrics

The collected data showed that radar tracking was generally less accurate at small scale. Due to the limits in resolution of the radar and the increased extraneous radar reflections for nearby objects, this was expected. In turn, the fusion method implemented was expected to increase the accuracy of tracking at small scale using the LiDAR and radar tracks as inputs. This was indeed generally observed across trial types and sensors.

Among the observed improvements, the S-turn small-scale trial run benefitted the most from the track fuser. Although the radar sensor, by itself, had relatively low errors for that trial, the fuser was still able to improve performance. The track fuser did produce higher tracker errors in two tests than either the LiDAR or the radar sensors working independently. This was likely due to the large difference in track accuracies between the LiDAR and radar for those tests. The track fuser can improve upon the individual tracks, but only if the accuracies of the individual tracks are within a certain margin. Otherwise, the fusion technique fails to improve upon either track.

Despite this occasional pitfall, out of 18 track metric increase factor comparisons (Table F2), only twice did the fusion technique deliver overall track metrics that were worse than using only the radar to track the vehicle at small scale. These two instances represented a 31% and 18% error metric increase for the MSE and RMSE during the S-turn trials. For all other instances, the increase factor decreased when using the track fuser in the small-scale environment. As with the various implementations of track-to-track or hybrid-level fusion [29,30,31,32] that noted increased tracking accuracy, the current investigation showed the successful deployments of radar and LiDAR fusion at small scale, using only the clustered centroid of point clouds as low-level inputs to a LiDAR tracker.

However, while the increase factor did decrease for most track metrics, in many instances it remained quite high. The Diagonal trial provides one such example. The track fuser decreased the increase factor for the MSE in the x dimension by 36%, but the increase factor remained at 26.3 times the original MSE from the full-scale test. This suggests a relationship between the relative accuracy pairings of the sensor tracks being fused. If two sensor tracks have widely different accuracies or covariances sustained over time, the track fuser may fail to produce a track with lower errors.

In general, however, this work has shown that a track fuser can increase track accuracy substantially over time when used in a challenging tracking scenario such as a 1/5th scale environment with clutter and highly non-linear vehicle maneuvers.

Conclusions

The aim of this work was to establish a method of including radar sensor tracking capabilities in STB vehicles. The team researched and implemented a high-level sensor fusion approach between radar and LiDAR to address several questions regarding this topic.

The first step was to establish accurate RTS at full scale as a benchmark comparison. The trial runs of vehicle maneuvers showed that the RTS was capable of overall mean absolute errors of 1.5 m or less for each trial, even in the highly non-linear S-turn driving scenario. Similarly, the same tracking software was used to track a vehicle in a 1/5th scaled environment, albeit with larger errors when accounting for the change in scale. Error metrics commonly increased by factors of 3 to 10, with some increases reaching a factor of 49. Attempting to restore accuracy metrics of the RTS required sensor fusion between the radar and LiDAR. Tracking accuracy for the small-scale trials improved by over 30% on average for all radar trials at small scale. The fused tracks resulted in increases of up to 70% for the small-scale S-turn trial. However, while improved, the track errors remained 3 to 5 times higher than the full-scale radar metrics in many cases. This suggests that the track fuser can improve upon, but not totally restore, the tracking accuracy of the full-scale radar tracks.

This investigation demonstrates the ability to use radar in small-scale environments with an associated track fuser, particularly during offline analysis of sensor recordings. The simplified LiDAR clustering and KF implementations make the approach accessible to many STB architectures that already implement a LiDAR sensor. To maintain radar functionality, the fused tracks can be reliably used as an input to any ADAS that needs radar track input.

Additional Products

The Education and Workforce Development (EWD) and Technology Transfer (T2) products created as part of this project can be downloaded from the project page on the Safe-D website. The final project dataset is located on the Safe-D Dataverse.

Education and Workforce Development Products

This project supported the work of one graduate student (Mr. Beale) and two undergraduate students (Mr. Alex Broz and Mr. Sean Copenhaver). Throughout the project, the students developed firsthand experience with agile project management and task completion, as well as experience with collecting, manipulating, and viewing data from automotive-grade radar, LiDAR, and DGPS sensors within integrated systems. These students were also part of the VTTI InternHUB program, which allowed them to work directly with and participate in several summer internships with the industry sponsor (i.e., Continental Automotive). The project also created two exhibits centered on radar and LiDAR systems in autonomous vehicles, one for over 6,000 attendees at the Virginia Tech Science Festival and one for several dozen attendees at a local school STEM night. The exhibits focused on the mechanics of radar wave propagation and laser scanners, as well as an introduction to the software needed to identify objects and create path trajectories for autonomous vehicles using the two sensors. A set of lecture notes peripherally related to the project was also developed for a graduate-level class centered on ADAS.

Technology Transfer Products

The initial project idea was pitched to the SAFE-D Industry Advisory Board and received board support. Continental Automotive was engaged with the research team throughout the project, as these efforts complemented their interest in sensor fusion for small-scale vehicles. For a long portion of the project, the research team held weekly update calls with Continental Automotive personnel to discuss progress on related tasks, providing a platform for informal T2. As part of these meetings, the research team also interacted with the extended community at Continental Automotive, making them aware of this effort. In addition, a publication, either in the form of a journal article or a conference proceeding that summarizes the results of the project, is currently being developed.

Data Products

The .bag files with the full-scale and small-scale vehicle trajectories are available for download. These files can found at this link.

References

1. National Center for Statistics and Analysis. Summary of motor vehicle crashes: 2016 data. (Traffic Safety Facts. Report No. DOT HS 812 580). Washington, DC: National Highway Traffic Safety Administration, 2018).

2. R. Ellenberg, R. Sherbert, P. Y. Oh, A. Alspach, R. J. Gross, J. Oh, "A common interface for humanoid simulation and hardware", *Humanoid Robots (Humanoids) 2010 10th IEEE-RAS International Conference on*, pp. 587-592, 2010.

3. Y. Paul. "Radio Controlled Car Model as a Vehicle Dynamics Test Bed." 2000.

4. S. Brennan and A. Alleyne, "Using a scale testbed: Controller design and evaluation," in *IEEE Control Systems Magazine*, vol. 21, no. 3, pp. 15-26, June 2001.

5. R. G. Longoria, A. Al-Sharif and C. Patil. "Scaled vehicle system dynamics and control: a case study in anti-lock braking." *Int. J. Vehicle Autonomous Systems, Vol. 2, Nos. 1/2,* 2004.

6. S. Brennan & A. Alleyne. "Robust Scalable Vehicle Control via Non-Dimensional Vehicle Dynamics", *Vehicle System Dynamics*, 36:4-5, 255-27. 2001.

7. D. Lodato, R. K. Kamalanathsharma and M. Farber. "Scaled Testbeds for Automated Robotic Systems." *2018 NVIDIA Ground Vehicle Systems Engineering and Technology Symposium.* 2018.

8. K. Sevcik, P. Oh. "Testing Unmanned Aerial Vehicle Missions in a Scaled Environment." *Unmanned Aircraft Systems.* Springer, Dordrecht. 2008.

9. Z. Xu, M. Wang, F. Zhang, S. Jin, J. Zhang, X. Zhao. "PaTAVTT: A Hardware-in-the-Loop Scaled Platform for Test- Testing Autonomous Vehicle Trajectory Tracking." *Journal of Advanced Transportation.* 2017.

10. O. Gietelink, J. Ploeg, B. De Schutter, and M. Verhaegen, "Development of advanced driver assistance systems with vehicle hardware-in-the-loop simulations," *Vehicle System Dynamics*, vol. 44, no. 7, pp. 569–590, July 2006.

11. M. O'Kelly, V. Sukhil, H. Abbas, J. Harkins, C. Kao, Y. P. Vardhan, R. Mangharam, D. Agarwal, M. Behl, P. Burgio, M. Bertogna. "F1/10: An Open-Source Autonomous Cyber-Physical Platform." 2019.

12. Continental AG, "Technical Product Specification: ARS430 RDI," Continental AG – Advanced Driver Assistance Systems Business Unit, October 2017.

13. R. E. Kalman. A new approach to linear filtering and prediction problems," *Journal of basic Engineering,* Vol.82. 35-45, 1960.

14. G. Welch, and G. Bishop. "An Introduction to the Kalman Filter," UNC-Chapel Hill, TR 95-04, 2006.

15. R. R. Labbe, "Kalman and Bayesian Filters in Python," 2015, GitHub repository https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python.

16. Velodyne Acoustics, Inc. "Velodyne LiDAR Puck," 63-9229 Rev-A datasheet. 2015.

17. "Documentation: Downsampling a PointCloud using a VoxelGrid filter." Pcl.org. Point Cloud Library. 2018.

18. "Documentation: Point Cloud Compression – Octree Compression." Pcl.org. Point Cloud Library. 2018.

19. R. Fischler and M. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM*. Vol24. 619-638. 1981.

20. Y. Roth-Tabak, and R. Jain. "Building an Environment Model Using Depth Information," *Computer*. Vol22. 85 – 90. 1989.

21. D. Xu and Y. Tian. "A Comprehensive Survey of Clustering Algorithms," *Annals of Data Science.* Vol2. 10. 2015.

22. H. Wang, X. Lou, Y. Cai, and L. Chen. "A 64-Line Lidar-Based Road Obstacle Sensing Algorithm for Intelligent Vehicles," *Scientific Programming.* 1-7. 2018.

23. N. Kaempchen, K. Fuerstenberg, A. Skibicki, and K. Dietmayer. "Sensor Fusion for Multiple Automotive Active Safety and Comfort Applications," *Advanced Microsystems for Automotive Applications*, 137-163. 2004.

24. D. Goehring, M. Wang, M. Schnurmacher, and T. Ganjineh. "Radar/Lidar sensor fusion for car-following on highways," *5th International Conference on Automation, Robotics and Applications*. 407-412. 2011.

25. S. Brennan and A. G. Alleyne. "Using a scale testbed: Controller design and evaluation," *Control Systems*. Vol21. 15 – 26. 2001.

26. R. Longoria, A. Al-Sharif, and C. Patil. "Scaled vehicle system dynamics and control: A case study in anti-lock braking," *International Journal of Vehicle Autonomous Systems*. Vol2. 2004.

27. D. Katzourakis and A. Katzourakis. "Scaled Test Bed for Automotive Experiments: Evaluation of Single Accelerometer Electronic Stability Control," *Advanced Microsystems for Automotive Applications,* pp239-257. 2008.

28. Z. Xu, M. Wang, F. Zhang, S. Jin, J. Zhang, and X. Zhao. "PaTAVTT: A Hardware-in-the-Loop Scaled Platform for Testing Autonomous Vehicle Trajectory Tracking," *Journal of Advanced Transportation*. 2017.

29. M. O'Kelly, V. Sukhil, H. Abbas, J. Harkins, C. Kao, Y. Pant, R. Mangharam, D. Agarwal, M. Behl, P. Burgio, and M. Bertogna. "F1/10: An Open-Source Autonomous Cyber-Physical Platform." 2019.

30. H. Hajri and M. Rahal, Mohamed. (2018). "Real Time Lidar and Radar High-Level Fusion for Obstacle Detection and Tracking with evaluation on a ground truth," *International Journal of Mechanical and Mechatronics Engineering*. Vol12. No8. 2018.

31. K. Na, J. Byun, M. Roh, and B. Seo. "Fusion of multiple 2D LiDAR and RADAR for object detection and tracking in all directions," *International Conference on Connected Vehicles and Expo*. 2014.

32. H. Cho, Y. Seo, B. Kumar, and R. Rajkumar. "A multi-sensor fusion system for moving object detection and tracking in urban driving environments," *IEEE International Conference on Robotics and Automation*. 2014.

33. S. Challa, M. R. Morelande, D. Mušicki, and R. J. Evans. "*Fundamentals of object tracking,"* Cambridge University Press. 2011.

34. D. F. Bizup and D. E. Brown. "The over-extended Kalman filter – don't use it!" *Proceedings of the Sixth International Conference of Information Fusion.* 1. 40- 46.

35. T. Fortmann, Y. Bar-Shalom and M. Scheffe, "Sonar tracking of multiple targets using joint probabilistic data association," in IEEE Journal of Oceanic Engineering, vol. 8, no. 3, pp. 173-184, July 1983.

36. J. Fisher and D. Casasent, "Fast JPDA multitarget tracking algorithm," Applied Optics, Vol.28. 371-376. 1989.

37. "Fitted Line.svg" by Msm is licensed under CC 3.0 Attribution Share. 2007.

38. U.S. Department of Transportation (2018) "Low-Speed Automated Shuttles: State of the Practice." Report no. FHWA-JPO-18-692. John A. Volpe National Transportation Systems Center. Cambridge, MA.

39. B. Schuhmacher, B. T. Vo, B. amd N. Vo. "A Consistent Metric for Performance Evaluation of Multi-Object Filters." IEEE Transactions on Signal Processing, Vol, 56, No, 8, pp. 3447-3457, 2008.

Appendix A: Kalman Filter Equations

There are two major sets of linear control equations included in a KF. One pertains to the prediction update (a priori estimate) and the other to the measurement update (a posteriori estimate) [14]. The prediction step requires three basic elements: (1) the magnitude of the time step used to predict the future state, (2) the process noise expressed in variances and covariances (i.e., uncertainty values in the model predictions), and (3) the current covariances of the system. Only two calculations are needed. One calculation predicts the state at the future time step (Eqn. 1), and one updates the covariances of the system given the process noise (Eqn. 2).

Three equations are then computed to complete this step of the KF. First, the Kalman gain is computed with knowledge of the sensor noise and the previously updated covariance matrix from the predict step (Eqn. 3). The Kalman gain is used as a weighting factor in whether to trust the a priori estimate of the current state more or less than the incoming measurement values and, thus, introduces Kalman's novel approach to the filtering problem [13]. Second, the a posteriori state estimate is calculated using the Kalman gain factor along with the difference between the predicted and measured (from the sensors) states (Eqn. 4). Third, the covariance matrix of the system is updated to express the filter probabilistic certainties (Eqn. 5). The filter can then recursively proceed to the next time step.

Predict:

$$\mathbf{x}_k^- = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_{k-1} \quad (1)$$

$$\mathbf{P}_k^- = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{Q}_k \quad (2)$$

Update:

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}^T(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R})^{-1} \quad (3)$$

$$\mathbf{x}_k = \mathbf{x}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\mathbf{x}_k^-) \quad (4)$$

$$\mathbf{P}_k = \mathbf{P}_k^-(\mathbf{I} - \mathbf{K}_k\mathbf{H}) \quad (5)$$

Equations 1 through 5 employ the naming convention in Challa et al. [33]. Here, **x** is a column vector that denotes the system state estimates, **F** is the projection of the state into the next time step, the product **Bu** incorporates control commands (which is not applicable in the case under investigation, i.e., tracking an unknown moving target with a static radar), **P** is the covariance matrix of the system, **Q** is the process covariance matrix, **H** is the mapping function from measurement space to state space, **R** is the

sensor covariance matrix, **K** is the Kalman gain, **z** is a matrix of measurement values, **I** is the $n \times n$ identity matrix (where n is the state space size—trackers that only estimate position with two-dimensional Cartesian coordinates, x and y, will have n equal to 2, while those that estimate position and velocity with respect to x and y will have n equal to 4), and the difference $\boldsymbol{z} - \boldsymbol{Hx}$ is commonly referred to as the residual or innovation. The subscripts $k-1$ and k refer to the previous and current time step, respectively. In a similar fashion, the superscript "−" denotes that the value has been obtained during the a priori estimate. Kalman proved these equations result in minimizing the mean of the squared error in the state estimation [13, 14].

The EKF variation arises from the origin of the KF; while Kalman originally designed the KF to track *linear* processes, purely linear applications are rarely observed in nature. Vehicles, airplanes, pedestrians, and other objects of interest can easily accelerate in two or three dimensions, and linear models are often inadequate to effectively track their movement. This required a solution for non-linear tracking with the KF, which became the EKF.

Based on the popular technique to linearize any non-linearities in the model by using the first few terms in a Taylor Series expansion, the EKF uses partial derivatives of the process and measurement relationships with respect to the state vector to predict the system state [14]. Systems governed by non-linear equations are accompanied by continuously changing values for the sensor and process covariance matrices (**R** and **Q**, respectively), and the mapping matrix (**H**). To combat this, at each time step, the Jacobian of those matrices is computed and applied to the covariance matrices. Additionally, the non-linear functions for the time projection (f) and mapping (h) can be defined for each time step to consider any non-linear movement in the state or non-linear mappings of the state space to the measurement space. Welch and Bishop [14] provide a good overview of this process and the variation of the five KF equations required, which are presented in Equations 6 to 10 [14, 34].

Predict:

$$\mathbf{x}_k^- = \boldsymbol{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) \quad (6)$$

$$\mathbf{P}_k^- = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{W}_k\mathbf{Q}_{k-1}\mathbf{W}_k^T \quad (7)$$

Update:

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T\left(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{V}_k\mathbf{R}_k\mathbf{V}_k^T\right)^{-1} \quad (8)$$

$$\mathbf{x}_k = \mathbf{x}_k^- + \mathbf{K}_k(\mathbf{z}_k - h(\mathbf{x}_k^-)) \quad (9)$$

$$\mathbf{P}_k = \mathbf{P}_k^-(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k) \quad (10)$$

The terms of $\mathbf{W}_k$, $\mathbf{V}_k$, and $\mathbf{H}_k$ have been added to represent the Jacobian at time step *k* of the process covariance matrix, the sensor noise covariance matrix, and the mapping function, respectively. The most important step in the EKF process relies on linearizing the mapping function, **H**, as it relates non-linear system measurements to linear estimates, *h*(), that fit within the linear stochastic model of the KF. Each of the three measurement update equations (Eqns. 8, 9, 10) rely on the linearization of the mapping function [15]. Systems that attempt to model the changing velocity of tracked objects need to expand this step from the regular KF to the EKF.

Appendix B: Data Association Techniques

A basic association technique is the nearest neighbor filter (NNF). In simplified tracking environments where an object only returns a single data return to the sensor, the NNF first filters all the data returns to a statistically relevant area around the tracked object (a process called *gating*) and then selects the closest measurement to the predicted state of the tracked object. All other data returns are considered extraneous and the result of sensor noise and unwanted clutter inherent in the environment [33]. Because the state covariance matrix includes standard deviation information, the point-by-point statistical comparison to find first the gating thresholds and then the closest data point is relatively straightforward, and the closest point can be easily selected. As described, the NNF limits its state update equations to a single data point, which in actual tracking applications may not be as useful considering multiple data points can originate from the same object of interest. The simplified NNF therefore discards many of these useful data points.

The probabilistic data association (PDA) filter and its variations, on the other hand, use all the gated data returns to update the state prediction and are generally accepted as more accurate than the NNF in complex tracking environments. The PDA filter, however, requires two assumptions: (1) in the simplified tracking environment, the object of interest only results in one true data return to the sensor, and (2) the extraneous data returns are uniformly distributed in the perception space [33]. Following these assumptions for the PDA, the true measurement of the object lies somewhere within the gated threshold values but cannot be exactly determined by selecting only one of the gated measurements. Instead, the PDA filter proposes to use all gated measurements after they are statistically weighted and their probabilities normalized [10]. In this manner, data points that more closely resemble the predicted state account for more in the measurement update step, but other returns within statistically relevant positions in the environment still influence the a posteriori state estimate.

One adaption of the PDA is the joint probabilistic data association (JPDA) filter. This extends the underlying principles of the PDA filters to work well when multiple tracks cross within proximity—a scenario where the accuracy of the NNF and basic PDA filters often suffers due to their core assumptions, specifically the PDA assumption of uniform distribution of extraneous data returns outlined above. When track validation gates overlap, measurements from sensors can fall within both gates (Figure B1). For example, it is not immediately clear which track the measurement M_3 belongs to. The NNF will choose the closest points to the center of the tracks, assuming all others to be possible new tracks; the PDA filters will weigh all measurements in the validation gate based on statistical proximity and expected noise densities in the viewing area. For PDA filters, this can cause errors in measurement probability weightings when those measurements originate from another track and are not sensor noise, and, possibly, the tracker could diverge [35]. Fortmann et al. (including Bar-Shalom) recognized this limitation of PDA filters and proposed their solution: the JPDA [35].

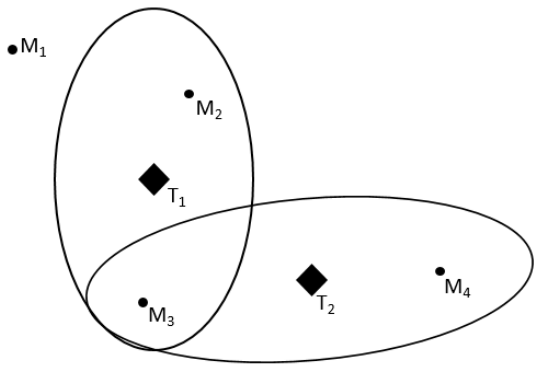

Figure 3. Diagram. Typical data association problem with two tracks and four measurements and associated validation gates.

To overcome this issue, Fortmann et al. [35] modified the way probabilistic weightings were calculated using what they called "feasible joint events." This time, the JPDA can probabilistically weigh the shared measurements of validation gates as possibly coming from both tracks and not as strictly noise like the PDA does. For every instance that track validation gates overlap, a cluster is extracted and a binary validation matrix is created (Figure B2). This matrix outlines the measurement and track possible pairings: each row is a measurement, and each column is a track assignment with the first column representing clutter. Values of "1" signal a possible pairing and values of "0" indicate non-possible pairings based on the validation gates. Of course, measurements outside the gates are not included. From here, "feasible joint events"—possible measurement and track pairing combinations—are created and commonly labeled Ω_i [35]. Feasible events are created by looping through the validation matrix and selecting one measurement per row and one track per column. This ensures that the measurement could only come from one source, and each track is associated with only one measurement (the noise column can have any number of measurements because the JPDA needs to be able to model the situation where all measurements were the result of clutter) [35]. The validation matrix and collection of feasible events for the instance in Figure B1 is shown in Figure B2.

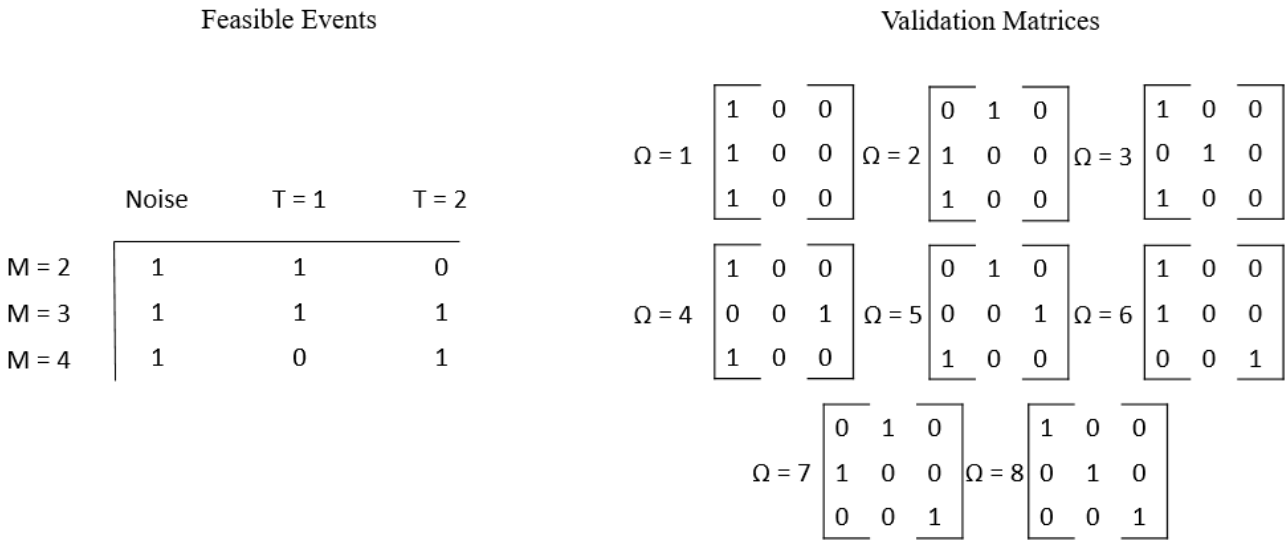

Figure 4. Image. Validation and feasible event matrices for the configuration in Fig. B1 [36].

The feasible joint event matrices allow the JPDA to calculate the correct association probabilities denoted β_m^t , where t is the track number and m is the measurement number. Calculating β_m^t is analogous to calculating the probability that the measurement, m, belongs to the track, t [35]. Using the expected probability of detection for the track, whether the measurement was assigned to a track or clutter, the probability density for each measurement and track pair, and a normal distribution scaled by the innovation, probabilities for each element of the feasible event matrices are calculated. Then, the probabilities for each measurement and track pair in the validation matrix can be summed by the corresponding entries in Ω_i and normalized to arrive at β_m^t, the correct weighting factors for the JPDA [36]. The probabilities β_m^t influence the overall innovation matrix for the measurements in this time stamp and, in the immediate steps, the Kalman gain factor.

Fortmann et al. [35] successfully demonstrated the JPDA's superiority over the regular PDA and the NNF when tracking crossing targets in the presence of clutter using their technique. However, the JPDA can suffer from long run times based on the sheer number of feasible event matrices needed and high volume of probability calculations for each of those matrices. The example in Figure B1 only contained three measurement points within the gates and two overlapping tracks, but those instances where more tracks overlap or there are more measurements (possibly from multiple sensors) can result in millions of feasible event matrices [36].

Appendix C: RANSAC

An increasingly complex option to identify the ground plane is to perform a least squares regression on a relevant subset of data points to estimate the angle of the road. However, there will be an unknown quantity of data points that belong to the road in any given point cloud frame, and a least squares regression may erroneously include data points from other nearby objects, skewing the plane estimate. The RANSAC approach offers a solution to this problem. Widely used and efficient, the RANSAC approach is to:

1. Randomly sample the minimum number of points needed to estimate the model of interest. For example, to estimate a line, two points are needed; for a plane, three points are needed, and so on.
2. Based on the resultant model estimate, determine the errors associated with fitting the current model to the rest of the data points.
3. Count how many of those data points lie within a given error threshold value.
4. Repeat the process *N* times for other randomly sampled starting points.
5. Choose the configuration with the best number of fitting data points [19].

Each random sample generates inliers and outliers for the model based on the error threshold value. The goal is to generate a model that identifies the noisy data as outliers and the "true" values as inliers, thus ignoring any skewed data returns in the parameter calculation. Figure C1 shows the different solutions RANSAC and regression could produce, given the same data set [37]. Usually, the RANSAC algorithm will terminate before performing *N* random samples if a certain proportion threshold of inliers is detected for the current model. RANSAC can also be used to detect multiple planes, a useful tool when a vehicle encounters a hill or speed bump where the road plane changes at a certain distance.

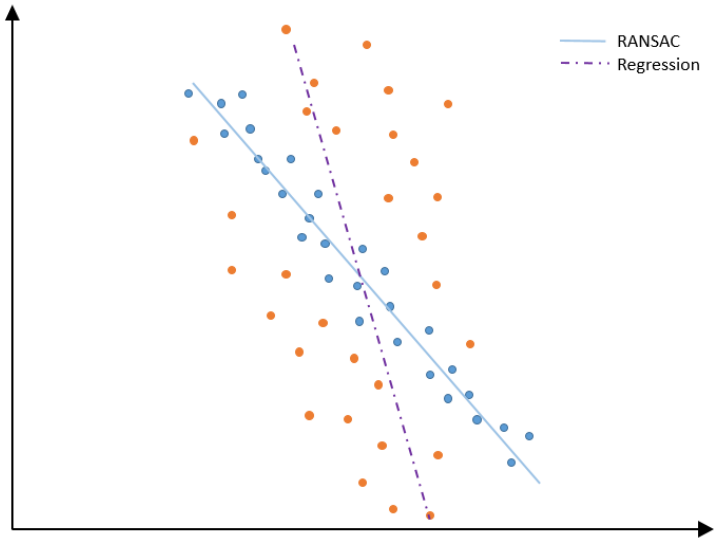

Figure 5. Graph. Line estimation with RANSAC showing inliers and outliers and possible regression line for the data set **[37].**

Appendix D: Sensor FOVs and Trial Configurations

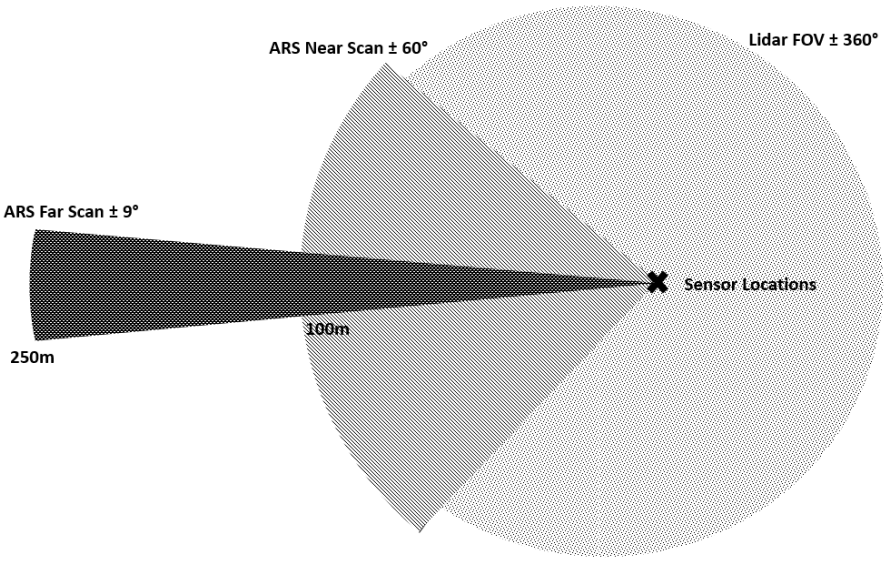

Figure 6. Diagram. Sensor FOV configurations for the vehicle tests.

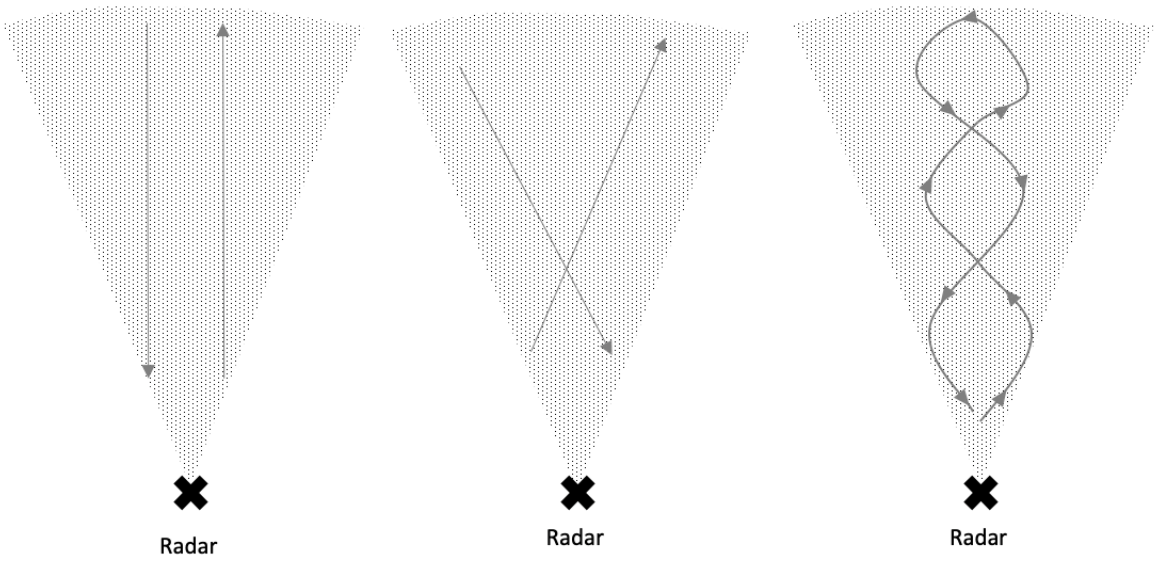

Figure 7. Diagram. The three selected vehicle maneuvers.

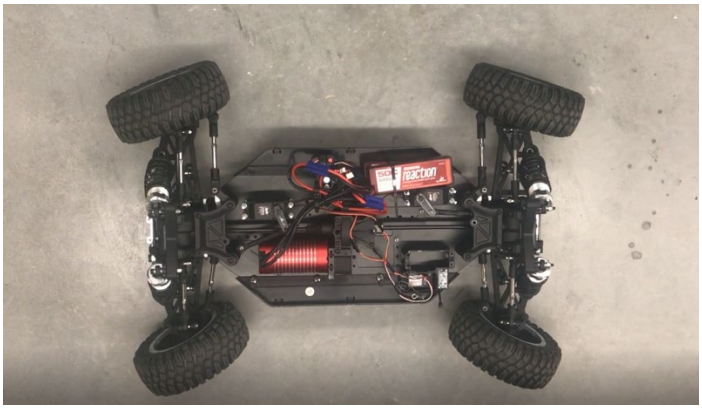

Figure 8. Photograph. The 1/5th scaled vehicle base.

Appendix E: Error Metrics

MEAN SQUARED ERROR

The MSE measured the average squared difference between the track values and the ground truth values at each time step the track was present in meters^2 (Eqn. 11). Given a vector of track positions, $x_{track,i}$, a vector of ground truth values, $p_{truth,i}$, and the number of time steps, N, the MSE was calculated as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\Delta x)^2, \quad \Delta x = x_{track,i} - x_{truth,i} \tag{11}$$

The MSE was calculated in both the *x* and *y* directions to independently analyze longitudinal and lateral track accuracy. For simplicity, only the calculation for the *x* direction is shown in the previous and subsequent equations. Calculations for the lateral (i.e., *y*) direction would simply have the *x* coordinates in each equation replaced by *y* coordinates.

ROOT MEAN SQUARED ERROR

Similarly, the RMSE was calculated through Eqn. 12 with the additional step of taking the square root. The resultant value was in meters. Given an MSE, the RMSE can be calculated as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\Delta x)^2}, \quad \Delta x = x_{track,i} - x_{truth,i} \tag{12}$$

As before, the RMSE was calculated independently for the *x* and *y* directions.

MEAN ABSOLUTE ERROR

The MAE was also calculated to compare track accuracy for *x* and *y* directions separately and had units in meters. Given a Δx and N as before, the MAE is:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\Delta x|, \quad \Delta x = x_{track,i} - x_{truth,i} \tag{13}$$

Lower MAE, MSE, or RMSE values corresponded to higher track accuracy over time.

LINEAR CORRELATIONS

The linear correlation between the track positions and the ground truth vectors was also used as an error metric. More specifically, the Pearson linear correlation coefficient was calculated for this comparison [38]. Given two column vectors, one containing the track values, x_{track}, the other containing ground truth values, x_{truth}, and their means, $\bar{x}_{track}$ and $\bar{x}_{truth}$, the linear correlation was found for both the x and y dimensions using:

$$rho(a,b) = \frac{\sum_{i=1}^{N}(x_{track} - \bar{x}_{track})(x_{truth} - \bar{x}_{truth})}{\sqrt{\sum_{i=1}^{N}(x_{track} - \bar{x}_{track})^2 (x_{truth} - \bar{x}_{truth})^2}} \quad (14)$$

Pearson coefficient values closer to 1 (or -1) indicated strong direct (or inverse) correlation between the track positions and ground truth values.

ESTIMATION ERROR SQUARED

The EES characterized track accuracy using covariance values with positional errors for each time step and has sometimes been used to compare multi-object trackers [39]. Given the position errors vector, Δp_i at a given time stamp, i, and the covariance matrix from the sensor track at that time stamp, C_i, the EES for a given track was calculated as:

$$EES = \Delta p_i{}^T C_i{}^{-1} \Delta p_i \quad (15)$$

Larger EES values at a given time step equated to larger covariances and less certainty for a track's position.

SQUARED ERROR

The SE was simply the square of the error value for a given time step in the x or y directions. Unlike the MAE, RMSE, and MSE, the SE was calculated at each time step to show the progression in positional error based on the sensor track.

Appendix F: Small-Scale Test Results

STRAIGHT

The first small-scale trial featured the 1/5th sized vehicle driving straight away and then back towards the sensors. Table F1, Table F2, and Table F3 show the error metrics for the radar, LiDAR, and fused tracks. The *x*-value and *y*-value track position overlayed onto the DGPS ground truth positions for the Straight trial are also provided (Figure F1, Figure F2, Figure F3).

Table 7. Error Metrics for Small-Scale Vehicle Radar Tracking, Straight Trial

Small Scale Vehicle Straight Trial Run, Radar	*X*	*Y*
MSE (m^2)	5.9467	0.5344
RMSE (m)	2.4386	0.7310
MAE (m)	2.2361	0.5692
Linear Correlation	0.9816	0.9426

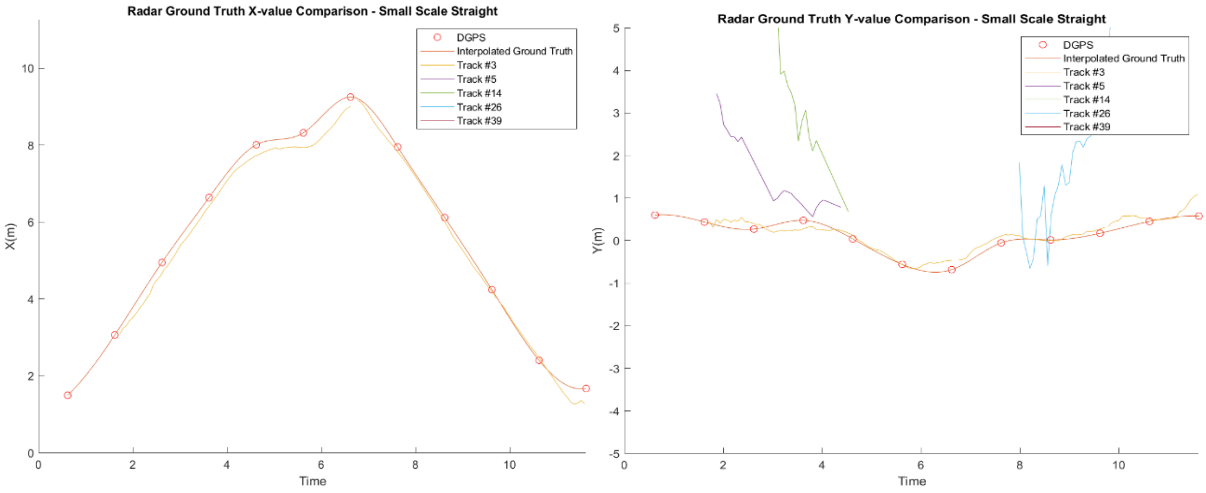

Figure 9. Graph. *x*-value (left) and *y*-value (right) radar track position and ground truth, small-scale Straight trial.

Table 8. Error Metrics for Small-Scale Vehicle LiDAR Tracking, Straight Trial

Small Scale Vehicle Straight Trial Run, LiDAR	*X*	*Y*
MSE (m^2)	9.2071	1.2424
RMSE (m)	3.0343	1.1146
MAE (m)	2.6984	1.0078
Linear Correlation	0.9740	0.9622

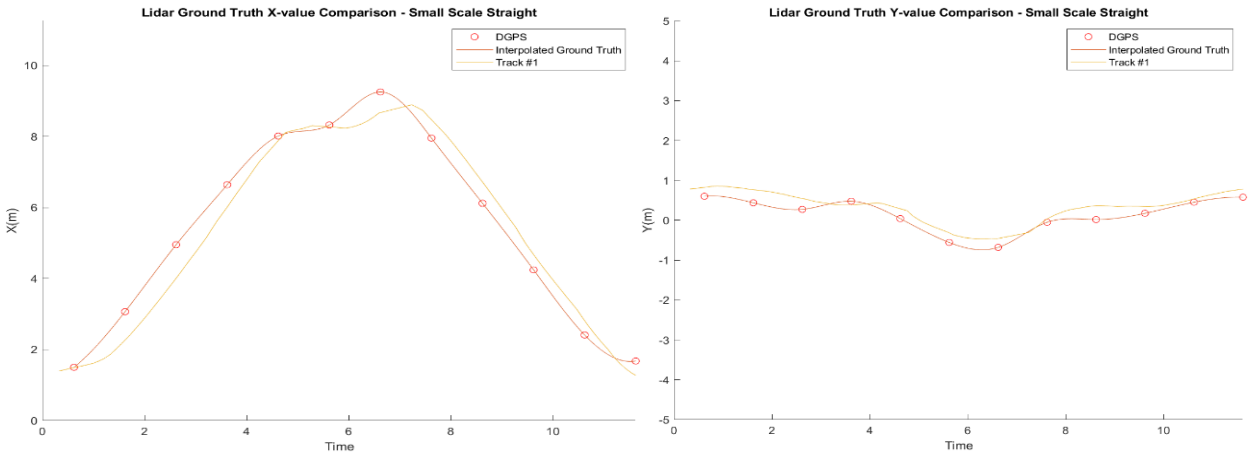

Figure 10. Graph. *x*-value (left) and *y*-value (right) LiDAR track position and ground truth overlay, small-scale Straight trial.

Table 9. Error Metrics for Small-Scale Vehicle Fusion Tracking, Straight Trial

Small Scale Vehicle Straight Trial Run, Fused Tracks	*X*	*Y*
MSE (m^2)	3.7869	0.3662
RMSE (m)	1.9460	0.6052
MAE (m)	1.3181	0.3845
Linear Correlation	0.9786	0.9406

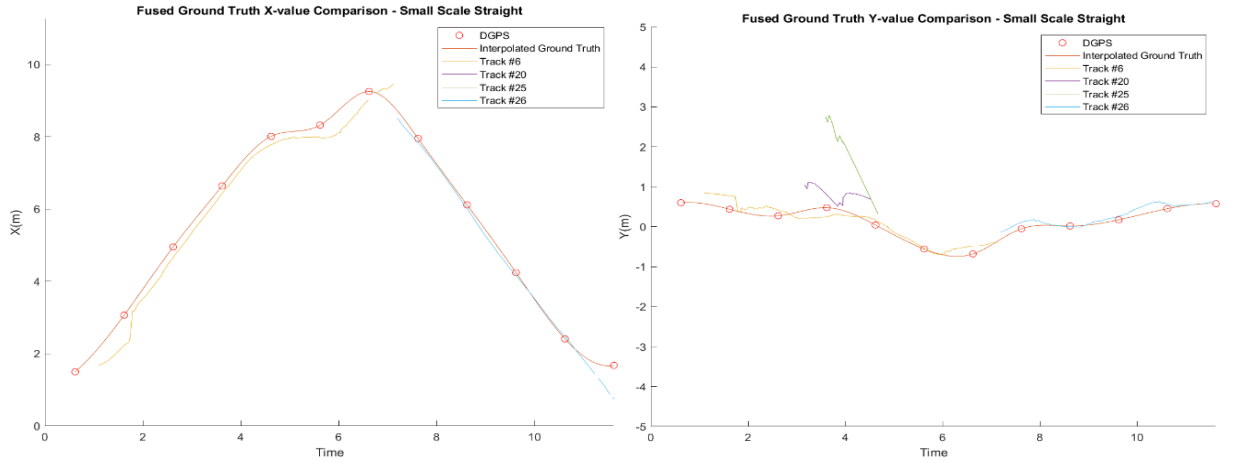

Figure 11. Graph. *x*-value (left) and *y*-value (right) fused track position and ground truth overlay, small-scale Straight trial.

DIAGONAL

The second small-scale trial featured the 1/5th sized vehicle driving diagonally away from and then across the sensor FOVs. Table F4, Table F5, and Table F6 show the error metrics for the radar, LiDAR, and fused tracks. The *x*-value and *y*-value track position overlayed onto the DGPS ground truth positions for the Diagonal trial are also shown (Figure F4, Figure F5, Figure F6).

Table 10. Error Metrics for Small-Scale Vehicle Radar Tracking, Diagonal Trial

Small Scale Vehicle Diagonal Trial Run, Radar	*X*	*Y*
MSE (m^2)	15.2316	11.5255
RMSE (m)	3.9028	3.3949
MAE (m)	3.2308	2.3558
Linear Correlation	0.9521	0.9704

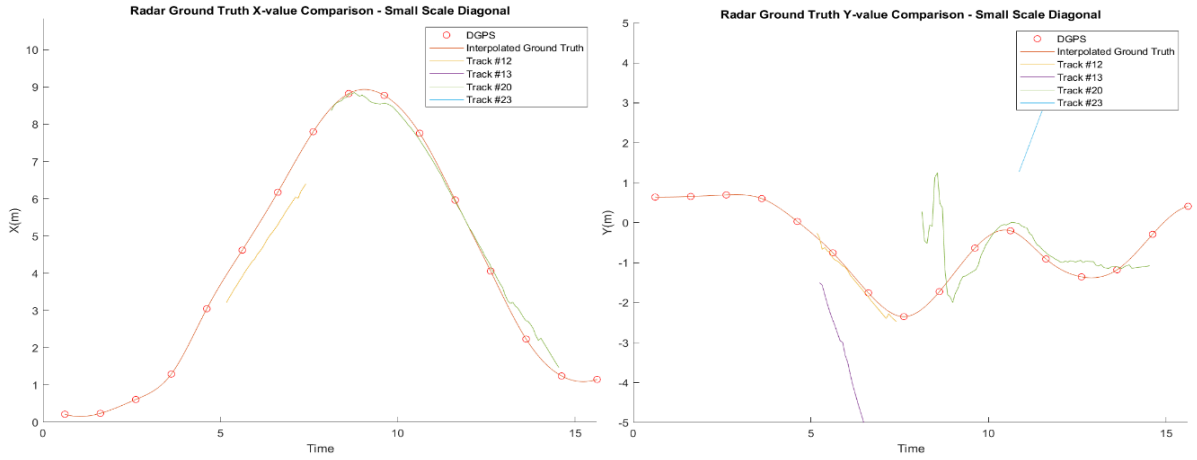

Figure 12. Graph. *x*-value (left) and *y*-value (right) radar track position and ground truth overlay, small-scale Diagonal trial.

Table 11. Error Metrics for Small-Scale Vehicle LiDAR Tracking, Diagonal Trial

Small Scale Vehicle Diagonal Trial Run, LiDAR	*X*	*Y*
MSE (m^2)	18.3231	3.8939
RMSE (m)	4.2805	1.9733
MAE (m)	3.6328	1.5070
Linear Correlation	0.9598	0.4704

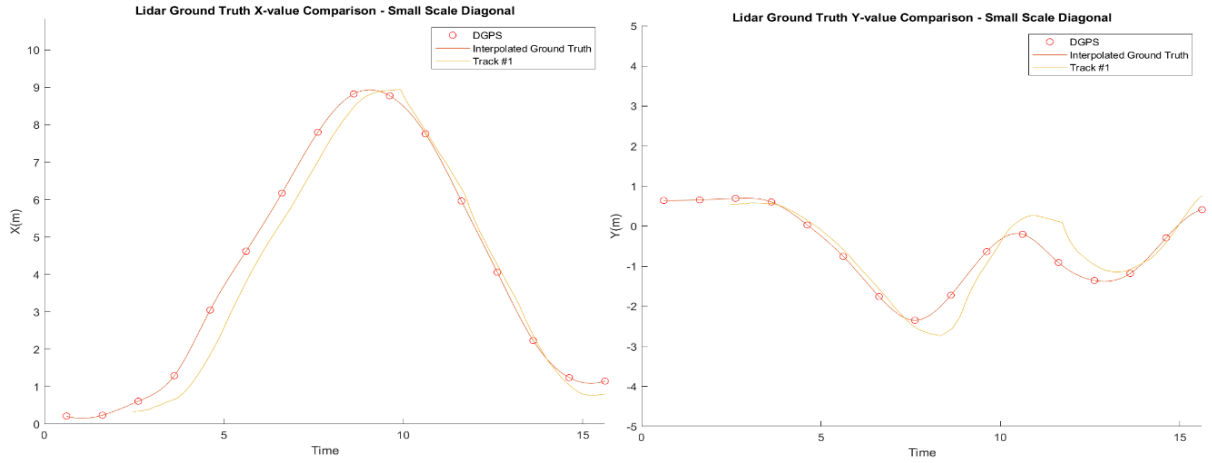

Figure 13. Graph. *x*-value (left) and *y*-value (right) LiDAR track position and ground truth overlay, small-scale Diagonal trial.

Table 12. Error Metrics for Small-Scale Vehicle Fusion Tracking, Diagonal Trial

Small Scale Vehicle Diagonal Trial Run, Fused Tracks	*X*	*Y*
MSE (m^2)	9.6805	6.7198
RMSE (m)	3.1113	2.5923
MAE (m)	1.9745	1.7689
Linear Correlation	0.9530	0.7040

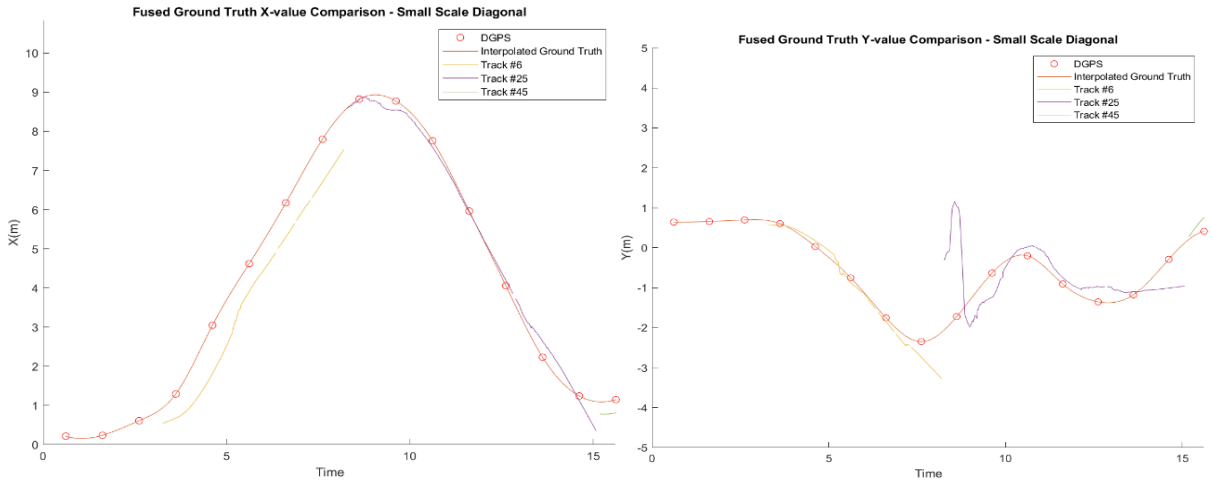

Figure 14. Graph. *x*-value (left) and *y*-value (right) fused track position and ground truth overlay, small-scale Diagonal trial.

S-TURN

The third small-scale trial featured the 1/5th sized vehicle driving while performing a series of S-turns, within the sensors FOV, moving away and toward the sensors. Table F7, Table F8, and Table F9 show the error metrics for the radar, LiDAR, and fused tracks. The *x*-value and *y*-value track position overlayed onto the DGPS ground truth positions for the S-turn trial are also provided (Figure F7, Figure F8, Figure F9).

Table 13. Error Metrics for Small-Scale Vehicle Radar Tracking, S-turn Trial

Small Scale Vehicle S-turn Trial Run, Radar	*X*	*Y*
MSE (m^2)	2.0691	2.0808
RMSE (m)	1.4384	1.4425
MAE (m)	1.1773	1.2066
Linear Correlation	0.9943	0.9852

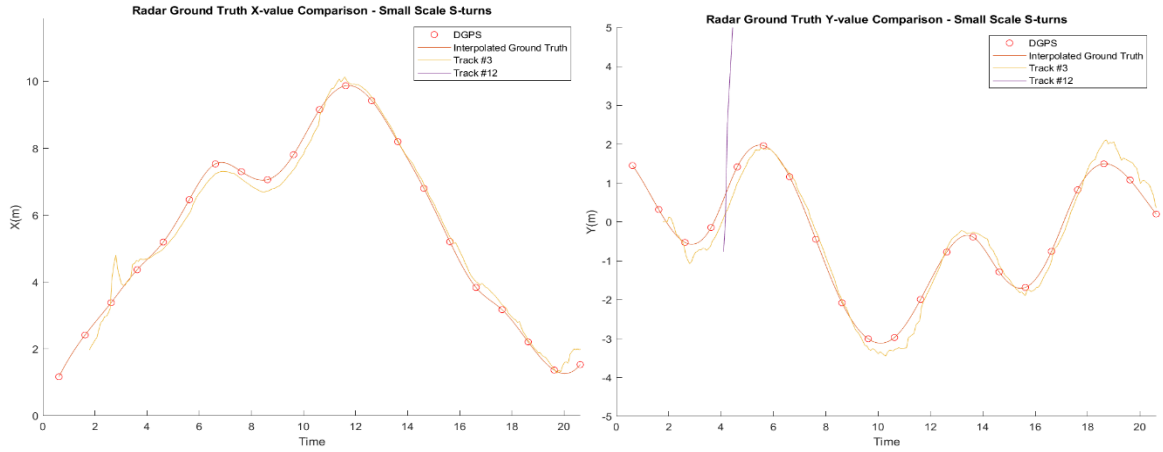

Figure 15. Graph. *x*-value (left) and *y*-value (right) radar track position and ground truth overlay, small-scale S-turn trial.

Table 14. Error Metrics for Small-Scale Vehicle LiDAR Tracking, S-turn Trial

Small Scale Vehicle S-turn Trial Run, LiDAR	*X*	*Y*
MSE (m^2)	5.7559	9.9737
RMSE (m)	2.3991	3.1581
MAE (m)	1.9804	2.6277
Linear Correlation	0.9872	0.9317

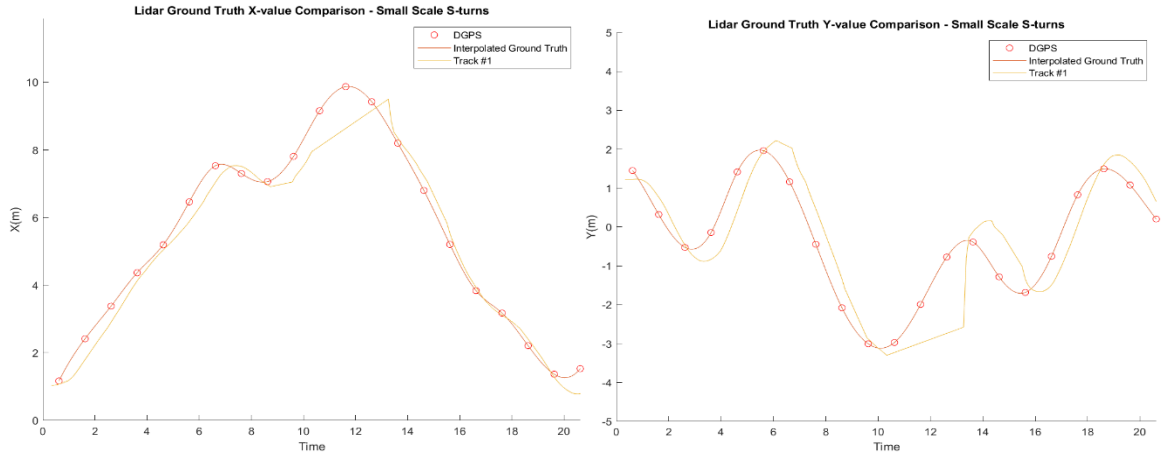

Figure 16. Graph. *x*-value (left) and *y*-value (right) LiDAR track position and ground truth overlay, small-scale S-turn trial.

Table 15. Error Metrics for Small-Scale Vehicle Fusion Tracking, S-turn Trial

Small Scale Vehicle S-turn Trial Run, Fused Tracks	*X*	*Y*
MSE (m^2)	1.2485	2.7226
RMSE (m)	1.1173	1.6500
MAE (m)	0.7049	1.1224
Linear Correlation	0.9924	0.9752

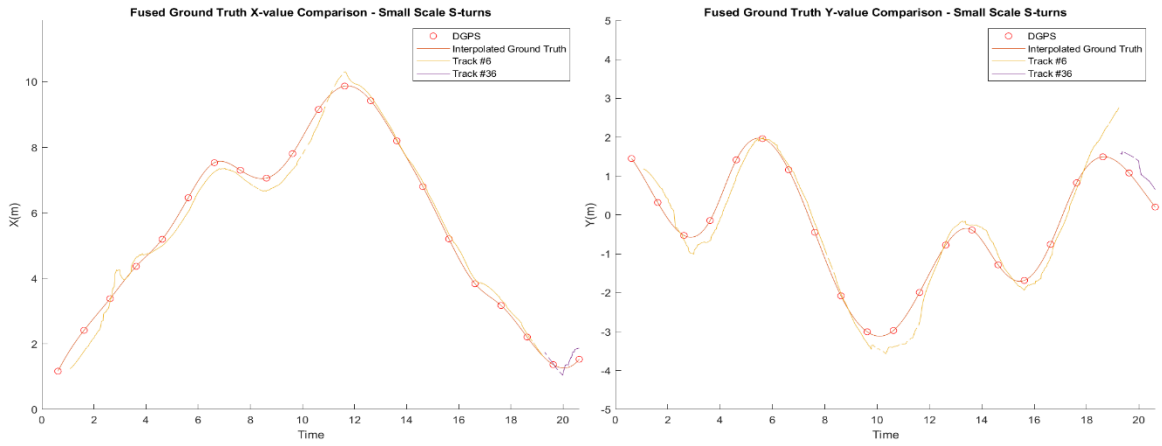

Figure 17. Graph. *x*-value (left) and *y*-value (right) fused track position and ground truth overlay, small-scale S-turn trial.