AVstack: An Open-Source, Reconfigurable Platform for Autonomous Vehicle Development

R. Spencer Hallyburton
Duke University
Durham, North Carolina
spencer.hallyburton@duke.edu

Shucheng Zhang
Duke University
Durham, North Carolina
shucheng.zhang@duke.edu

Miroslav Pajic
Duke University
Durham, North Carolina
miroslav.pajic@duke.edu

Abstract

Pioneers of autonomous vehicles (AVs) promised to revolutionize the driving experience and driving safety. However, milestones in AVs have materialized slower than forecast. Culprits include (1) the lack of verifiability of proposed state-of-the-art AV components, and (2) stagnation of pursuing next-level evaluations, e.g., vehicle-to-infrastructure (V2I) and multi-agent collaboration. In part, progress has been hampered by: the large volume of software in AVs, the multiple disparate conventions, the difficulty of testing across datasets and simulators, and the inflexibility of state-of-the-art AV components. To address these challenges, we present `AVstack`[1],[2], an open-source, reconfigurable software platform for AV design, implementation, test, and analysis. `AVstack` solves the validation problem by enabling first-of-a-kind trade studies on datasets and physics-based simulators. `AVstack` addresses the stagnation problem as a reconfigurable AV platform built on dozens of open-source AV components in a high-level programming language. We demonstrate the power of `AVstack` through longitudinal testing across multiple benchmark datasets and V2I-collaboration case studies that explore trade-offs of designing multi-sensor, multi-agent algorithms.

CCS Concepts

• **Computer systems organization → Robotic autonomy**; • **Software and its engineering → Software libraries and repos**.

Keywords

autonomous vehicles, perception, tracking, planning, control

ACM Reference Format:
R. Spencer Hallyburton, Shucheng Zhang, and Miroslav Pajic. 2023. `AVstack`: An Open-Source, Reconfigurable Platform for Autonomous Vehicle Development. In *ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2023) (ICCPS '23), May 9–12, 2023, San Antonio, TX, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3576841.3585930

[1]https://avstack.org/research [2].
[2]https://github.com/avstack-lab.

ICCPS '23, May 9–12, 2023, San Antonio, TX, USA

ACM ISBN 979-8-4007-0036-1/23/05…$15.00
https://doi.org/10.1145/3576841.3585930

1 Introduction

The AV industry has proliferated over the past two decades. Experts point to the DARPA Grand Challenge as the coming of age of AVs [30]. Soon after, expectations ballooned that we would see fully automated vehicles on the road within a decade [9]. In response, the autonomy community has exploded into industry and academic players both large and small. However, milestones in AV development have slowed in recent years. In fact, Tesla has promised to deliver fully self-driving cars "next year" for the last 8 years [34] and is as of yet still deploying Level-2 solutions.

A major challenge to AV development is that most AV solutions are proprietary and closed-source. This is a result of the immense cost of development that safety-critical AVs require. However, the rush to deploy autonomous vehicles and the proprietary nature of industry solutions are conflicting. In particular, industry progress is outpacing research and development. This disparity is negatively impacting progress and trust in AVs. While industry rushes to capture a new market, access to representative platforms is hampering fundamental safety and performance research [13].

We find two culprits for such a slowdown in AV research. First, proposed state-of-the-art AV algorithms and components perform insufficient transfer testing and longitudinal analysis. This leads to a lack of accountability and verifiability. Second, much AV research pursues (marginal) improvements on single-component benchmarks (e.g., LiDAR-based detection challenges [10, 15, 33]). Similarly, pursuing critical next-level evaluations such as multi-agent (e.g., vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I)) collaborative sensing or safety & security analysis has stagnated.

At the root of these problems are several barriers: (1) Designing and implementing an AV require large amounts of complex software. The jumps from testing components on static datasets to longitudinal datasets to full-stack simulations are large, and no existing platform can handle all scenarios. (2) Data sources use different conventions for coordinates, reference frames, metrics, calibrations, and more, which makes case-by-case conversions prone to error. (3) Mature AV platforms are designed with custom messaging protocols in low-level languages, which creates rigid AV architectures and implementations. Rigidity inhibits modular testing and puts a high-barrier on design changes. (4) Implementations of state of the art AV algorithms are highly tailored towards benchmark challenges. Adapting them to new contexts is time-consuming.

Several platforms have emerged to support open-source AV development. Baidu's Apollo [3] and Autoware [1] are established as preeminent platforms for deployable, real-time AVs. Recently, Pylot [16] has also allowed for more trade studies in AVs (i.e., evaluating the impact of parameter/configuration changes on output metrics). Each of these platforms is useful and needed. However, each has serious shortcomings when it comes to the design of novel AV architectures for next-level challenges and transfer testing of AV components between datasets/simulators.

To fill this void, we present AVstack, a new research platform for the design, implementation, test, and analysis (DITA) of AVs. AVstack has the following four key innovations designed to promote modular AV design, simple implementation, wide-reaching testing, and insightful analysis. The key innovations are:

(1) *Wide compatibility:* To our knowledge, AVstack is the first platform compatible with *both benchmark AV datasets and physics-based AV simulators*. It maintains compatibility with dozens of *open-source AV algorithms* across AV components and established metrics.

(2) *Unified conventions:* AVstack implements a flexible set of coordinate conventions attached to all vector-type objects to unify coordinates. It also unites component-wise metrics from multiple providers. This helps maintain forward and backward compatibility, reduces the user burden to keep track of case-by-case uniqueness, and enables complex, multi-sensor, multi-agent configurations.

(3) *Modular testing:* AVstack streamlines DITA in AVs. Reconfigurability breaks rigid constraints of prior platforms allowing for novel designs and reusable implementations. With AVstack, testing is performed seamlessly on static/longitudinal datasets and physics-based simulators with metrics at every AV component.

(4) *Low barrier adoption:* AVstack is written in a high-level programming language to allow for rapid prototyping and reusable implementations. A suite of AVs can be designed, implemented, tested, and analyzed with little effort in unique configurations such as multi-sensor, multi-agent settings.

AVstack is a framework for AV development. It provides a unique combination of performance and modularity – high-performing algorithms with a flexible and reconfigurable architecture to enable diverse and rapid prototyping. AVstack is not the "best" framework for all AV applications. However, its innovations fill important voids in validating state of the art results, transfer testing between datasets/simulators, standardizing AV evaluations, and pursuing next-level questions in multi-sensor, multi-agent configurations. AVstack is available open-source[3].

In summary, AVstack contributes the following innovations:

- Unifies testing and analysis within and between benchmark static/longitudinal datasets and physics-based simulators.
- Enables reconfigurable and reusable AV design through standardized interfaces and open-source support.
- Unifies disparate coordinate conventions to achieve forward *and* backward compatibility to data sources.
- Streamlines component-wise metrics in all cases ranging from single-component analysis on static datasets to full-stack AVs on longitudinal situations.
- Promotes easy transfer testing between datasets and simulators with low-barrier trade study configuration tables.
- Provides a standardized interface for training supervised learning models on datasets and the CARLA simulator [14].
- Facilitates testing multi-sensor, multi-agent scenarios.

Terminology. In this work, we use the following terminology:

- *Algorithm:* A specific implementation of an AV component; e.g., PointPillars [22] is an algorithm.
- *Component:* A generalization and grouping over algorithms; e.g., PointPillars [22] and 3DSSD [24] fall under the "3D object detection" component.
- *Module:* A grouping of similar components under a goal; e.g., 2D & 3D object detection → "perception" module.
- *Architecture:* A designed connection of *components* that will process sensor data and output control signals or state.
- *Implementation:* A connection of specific algorithms that defines one particular realization of an AV architecture.

Organization. Section 2 summarizes related efforts and their shortcomings for AV DITA. Section 3 expands on challenges to AV research and how key design decisions allow AVstack to overcome these obstacles. Section 4 provides use-cases in longitudinal and multi-agent sensing demonstrating that AVstack enables new AV DITA capability. We finish with concluding remarks in Section 5.

2 Related Work

Deployable AV Systems. Baidu Apollo [3] and Autoware [1] are highly adopted and stable AV repositories. Each have an architecture philosophy and have provided specific implementations. Both are designed for self-driving and have established relationships with industry to deploy in physical systems. Apollo is built on the CyberRT message passing framework while Autoware uses ROS [28]. While both achieve high levels of performance, both struggle to maintain accessible APIs for research-level development. Both have high learning curves, are difficult to modify, and require powerful computers. Thus, both are ill-suited to perform longitudinal testing and reconfigurable prototyping on important benchmarks.

AV Research Platforms. Pylot [16] is an AV architecture in the Python language. Sensor data is passed using the low-latency, low-copy ERDOS [16] framework. Pylot provides an accessible interface where developers can compare algorithms within established components on the CARLA simulator [14]. Pylot demonstrated near-real-time capability on a real system at low speeds. While Pylot maintains an accessible API, it is limited to the CARLA simulator and real-world self-driving; it cannot be tested on benchmark datasets. While it supports different algorithms within each component, the component architecture is fixed. It is not suitable for next-level questions such as multi-agent, collaborative sensing nor does it support end-to-end learning-based implementations.

OpenAI maintains its gym for evaluating reinforcement learning in episodic tasks. Researchers have used gym to develop control algorithms in self-driving using AirSim [31] and TORCS [37]. gym is not well-suited for component-wise evaluations in AVs and is designed only as a tool for training reinforcement learning algorithms.

General Frameworks. ROS [28] provides a communication infrastructure above operating systems for multi-component robotics applications. ROS handles message passing between peer nodes in full-stack robotics case studies. In this way, ROS has greatly streamlined the development process for deployable robotics and has been recognized as a major research platform.

ROS and AVstack serve different purposes and can be used in a complementary way. AVstack handles development environments

[3] https://github.com/avstack-lab.

for DITA in both single-component and multi-component settings. AVstack is more suitable for rapid prototyping of AV algorithms and components while ROS is designed to handle communication between components for cyber physical systems (CPS) over potentially heterogeneous networks. For a full stack-simulation or a physical implementation, the two can be complementary: ROS can managed message passing and computation resources while AVstack can provide components and analysis.

3 AVstack Key Design Decisions

A recent slowdown in AV development is a consequence of at least two factors. First, proposed state-of-the-art AV algorithms and components perform insufficient trade studies, transfer testing, and longitudinal analysis. Second, a platform is needed that allows multi-component evaluations and lowers the barrier of pursuing next-level evaluations such as V2V and V2I collaboration.

AVstack was designed to address the above shortcomings and more. In this section, we present the high-level innovations of AVstack that have allowed it to uniquely fill this large void in AV DITA. As illustrated in Fig. 1, AVstack's key innovations have allowed for great strides in design modularity and robust testing & analysis. The critical design decisions of AVstack fall under:

- **Wide Compatibility:** AVstack is widely compatible with benchmark AV datasets and physics-based AV simulators. AVstack leverages many open-source AV components.
- **Unified Conventions:** AVstack standardizes coordinate conventions and maintains backward compatibility with legacy conventions. AVstack metrics and evaluations are expanded over many preceding benchmarks.
- **Modular Testing:** AVs can be designed quickly and flexibly in AVstack drawing from a bank of reconfigurable components. AVstack tests implementations seamlessly on static datasets, longitudinal datasets, and AV simulators with minimal software changes.
- **Low Barrier:** AVstack is written in a high-level programming language. A user can design Level 2-5 AVs and test in just dozens of lines of code. Trade studies can be initiated with simple configuration tables.

In the following, we present the motivations and high-level design details for AVstack. Each section begins by identifying specific barriers in AV development and how AVstack was designed as a solution to those challenges. We identify some intentional omissions from the design in Appendix A.

3.1 Design Goal 1: Wide Compatibility

A community-supported foundation is of the utmost concern in AVstack. To ensure utility and staying power, AVstack was designed to be widely compatible with gold-standard benchmark datasets and simulators. To support representative AV design and implementation, AVstack maintains compatibility with many open source components. Below, we provide details on this compatibility.

3.1.1 Design Goal 1.1: Inferfaces & APIs

Motivation. The challenge to designing a widely-compatible platform is in supporting backward-compatibility and preparing for forward-compatibility without requiring constant overhauls that precipitate uncontrolled software bloat. This task is difficult enough that, until now, we had yet to see a platform bridge the dataset-simulator gap and deliver inter-source compatibility. In the following, we investigate several challenges.

Datasets. The KITTI dataset [15] changed the world of autonomous driving. Many foundational works in computer vision benchmarked on KITTI. However, despite KITTI's success, it has fundamental limitations; namely, its small size and lack of full 360° camera coverage. Further, algorithms trained on KITTI have shown lower performance when transferred to other datasets [5], suggesting that training on KITTI may suffer from overfitting.

In recent years, major players including Waymo and Motional have released datasets more extensive than KITTI with multiple sensing modalities [10, 33]. Despite this, many works still benchmark primarily on KITTI with only marginal improvements over prior results. To investigate, we scraped the KITTI leaderboard and selected all works from the top 50 places with a validated journal or conference publication. Of the 18 validated entries, many are recent: 13 were published in 2022; all have been released since 2020. They are all within 3% on the leaderboard. Disappointingly, only a single entry ran experiments on KITTI, nuScenes [10], and Waymo [33] datasets, while one entry ran on KITTI and nuScenes, and eleven entries ran on KITTI and Waymo; see Appendix B for the full table.

Simulators. Simulators such as CARLA [14] allow for important AV testing in a dynamic environment, provide closed-loop feedback (i.e., planning, control), and enable rare-event simulations that are difficult or dangerous to capture in the real world. However, CARLA comes with minimal resources to bootstrap AV DITA. In fact, CARLA provides no algorithms that do not use ground truth data nor do they provide an architecture for standing up one's own AV. As a result, there are few relevant AVs designs out of CARLA and few benchmark submissions to the CARLA challenge [4].

Recent works using CARLA have established success using end-to-end, vision-based reinforcement and imitation learning (RL, IL) in AVs (e.g., [11, 14, 27, 35]). The lean towards RL/IL is in part because CARLA has *no built-in support for training or testing supervised learning algorithms*. There is no "CARLA dataset" and no clear way to generate one. There are several barriers to this, including that, to our knowledge, there is no way to obtain the list of objects in the field of view (even as a ground-truth oracle) without using a depth-sensor to determine if e.g., a building is blocking the view to the object. On the other hand, continuously running simulator trials with a reward function for RL/IL is easy.

Design Goal. We designed AVstack to both bridge the dataset-simulator gap and to expand upon critically-absent features in existing APIs. We identified the core features *essential* for wide dataset and simulator compatibility. The overarching theme of these innovations is simple: attach details to objects, not just documentation.

In particular, some features that enable AVstack's wide compatibility include: assigning attributes directly to sensor measurements so that each natively possesses all identifying information; expanding all object labels with 3D bounding box, object type, object ID, velocity, acceleration, orientation, and angular velocity fields; defining flexible data structures to route sensor data to multiple end-points; standardizing reading, writing, and transforming sensor data and ground truth labels, and much more. Each feature is made possible

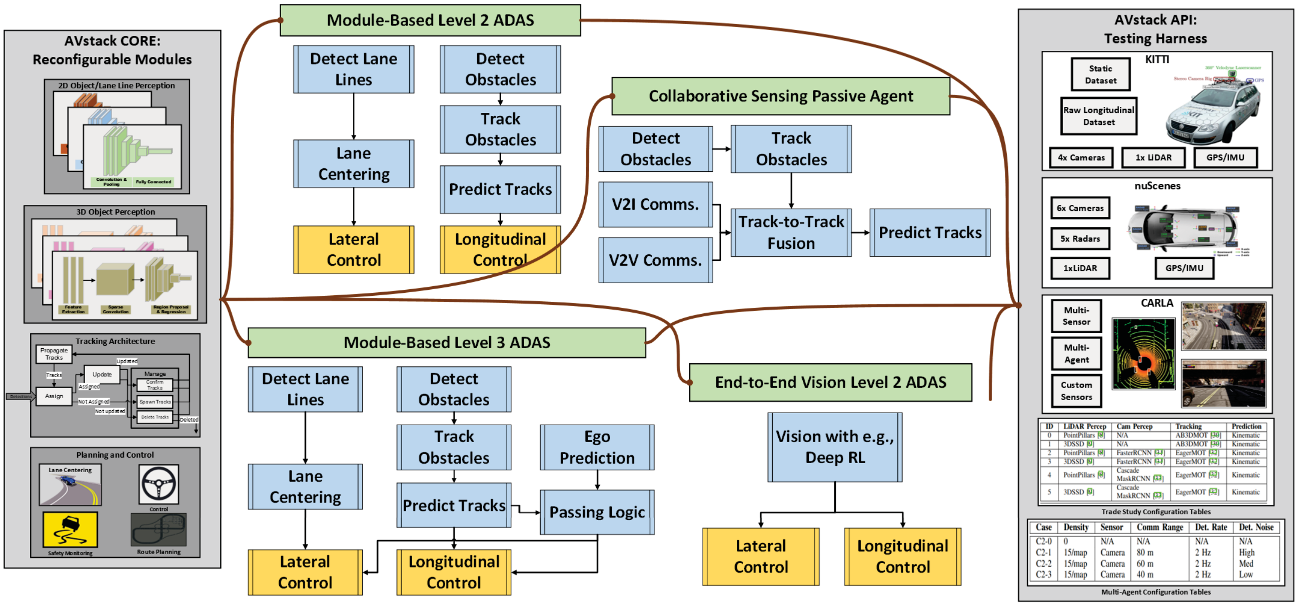


Figure 1: (left) AVstack's core library provides source-agnostic modular components within a reconfigurable architecture. AVstack unifies diverse conventions from open-source providers to deliver community-vetted AV design. (center) With the core library, developer can specify many AV architectures, including "active" (with control) and "passive" (without control). Shown are a sample of architectures used in this work. (right) AVstack's API library provides data source interfaces to the most popular of AV benchmarks and simulators. API also provides low-barrier trade-study capability with configuration tables.

by many precise decisions. For example, 3D bounding boxes need clear reference frames (e.g., camera-frame, lidar-frame, ego-frame), orientation angle definitions (e.g., yaw=0 aligned with x-axis in camera frame, yaw-pitch-roll vs. roll-pitch-yaw ordering), bounding box height-offset (i.e., $0 :=$ bottom of box, $0 :=$ center of box).

Until now, each user would write software tailored towards the minimum required information for a single dataset – the heterogeneity of options was too high a barrier to support multiple data sources. With AVstack, users can make software that is reusable and transferable between sources. Unlike Apollo, Autoware, and Pylot, AVstack is not designed towards a low-level, low-latency message-passing scheme. Rather, it exists in a high-level programming language quickly adaptable to new simulators and complex configurations. The reusability and adaptability lower the amount of effort required to test algorithms on different platforms. Now, we can start to expect more validation and verification of AV components on representative testing scenarios.

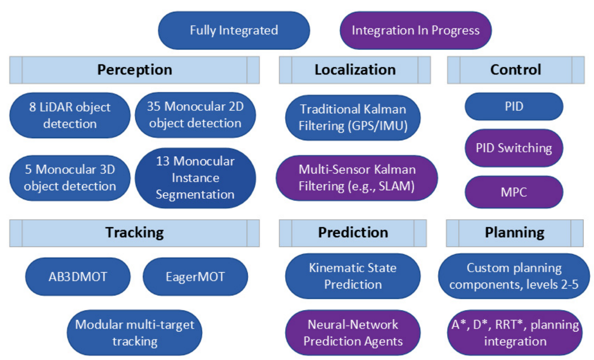


Figure 2: AVstack supports a broad set of open-source algorithms. Both custom implementations and open-source libraries bootstrap the diverse set of integrated capabilities. The set of compatible software is continuously growing.

3.1.2 *Design Goal 1.2: Component Compatibility*

Motivation. A frustrating challenge to researchers is that groundbreaking AV components are difficult to use beyond their original benchmarks. To enable insightful evaluation of new components within a longitudinal environment, the developer must be able to quickly stand up and rearrange an AV using off-the-shelf components. While an individual benchmark may have many perception algorithms that can all be tested uniformly, minimal support exists to stitch that perception algorithm together in a longitudinal evaluation with tracking, motion prediction, and path planning.

Platforms that provide some degree of component compatibility, such as Pylot, do not support both datasets and simulators. Pylot also uses a custom message passing framework designed to minimize latency which is not useful for algorithm trade studies. Rather, it is suited for applications with real-time consideration.

Design Goal. AVstack supports many prominent open-source components for both module-based design of AVs and end-to-end learning-based approaches. AVstack leans on existing open-source libraries to complement custom components. Fig. 2 provides a sample of the capabilities at the time of publication. In particular, AVstack currently supports 4 modes of perception with over 50 different perception algorithms. This wide compatibility is obtained under a common interface that allows for AV reconfiguration and reuse of algorithms and components.

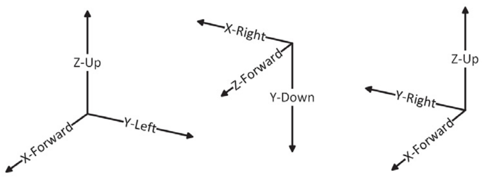


Figure 3: Sensors, datasets, and simulators often use different coordinate systems which may require up to a 4x4 transformation matrix to convert vectors between frames: (a) Standard frame employed by nuScenes [10], Waymo [33]; (b) Camera frame employed by KITTI [15]; (c) Left-handed frame employed by CARLA [14].

3.2 Design Goal 2: Unified Conventions

There is no "official" set of conventions for AV datasets and simulators. This is for good reason: each dataset satisfies different needs of the end user. Datasets without positioning data may specify all objects in ego-relative coordinates (e.g., [15]). Some datasets may introduce sensors not present in other datasets (e.g., radar in [10]) while others defer all sensor specification to the user (e.g., [14]). Further, end users may have different state-vector requirements with degrees of freedom ranging from three (x, y, yaw; e.g., [37]) to nine (x, y, z, height, width, length, roll, pitch, yaw; e.g., [10, 14]).

In a review of state-of-the-art AV datasets and simulators, we find that no two sources share the same coordinate axes, frame origin, and orientation angle conventions. In fact, we find that even within single providers (e.g., [15]) there can be discrepancies in the conventions. Similarities and differences are highlighted in Table 1.

3.2.1 Design Goal 2.1: Reference Frames

Motivation. The level of complexity and lack of standard of reference frames hinders dataset-agnostic component design and introduces error into complex multi-sensor, multi-agent configurations. To mitigate this, we standardize reference frame definitions with a wrapper around each dataset and simulator. We also introduce the reference frame chain of command (RefChoC) that represents the dependency on secondary reference frames (see Fig. 4).

Coordinate Axes. Coordinate frames cause headaches in even the most proficient of developers. This is particularly important when data sources use different conventions. One small difference is illustrated in Fig. 3. While KITTI always labels objects in a right-down-forward ("camera") coordinate frame, nuScenes and Waymo use frames dependent on the sensor which includes many possible orientations (see Appendix C). CARLA uses a non-traditional left-handed forward, right, up frame.

Rotation Conventions. Many datasets represent orientation with Euler or Tait-Bryan (grouped under the name "Euler" in this work) angles (see Table 1). This allows for a compact representation of the orientation that is (sometimes) human-interpretable. However, Euler angles are problematic for several reasons. The most obvious drawback is the lack of specificity: there are at least 12 accepted methods of specifying orientation using Euler conventions [20]. This introduces error into the development process. Second, Euler angles suffer from gimbal lock and discontinuities in certain special cases. While this is seldom a problem in real-world driving, it is very relevant for AV simulators due to local nature of map coordinates.

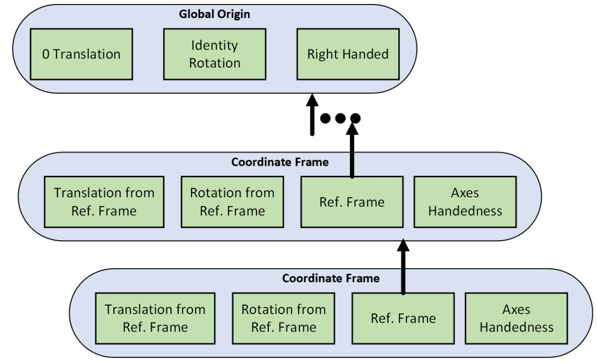


Figure 4: Reference frame defined by $(Tr, R, Ref\text{-}P, H)$, with Tr a translation, R a rotation, $Ref\text{-}P$ a parent reference, and H the axes handedness. Reference frame chain of command (RefChoc) followed to a common ancestor when comparing objects or fusing data from complementary sensors/agents.

Sensor Calibrations. Each sensor should be accompanied by a calibration that describes both where the sensor is positioned relative to the ego (often called: "extrinsics") and sensor-specific properties (often called: "intrinsics"). Unfortunately, many datasets have ambiguous calibrations. KITTI provides calibration data but minimal instructions on how to use it or which data requires transformation. KITTI also only allows for ego-relative coordinates, which can impair target tracking models. Meanwhile, CARLA describes the unique conventions of its coordinate system but no supporting functions in the software.

There is additional complexity beneath the surface across the board: calibrations must define *whether the translation is in the pre-rotated or post-rotated reference frame.* 4 x 4 transformation matrices use post-rotation while it is most interpretable to use pre-rotation. Different providers take different approaches, and details are seldom documented. Furthermore, specifying a calibration is ambiguous, even under a clear reference frame and pre/post order if it does not specify which *direction the transformation should be applied* (i.e., does it represent "$A \rightarrow B$" or "$B \rightarrow A$"?).

Design Goal. We performed many iterations designing reference frames for the unified API of AVstack. To achieve standardized reference frames for the first time and provide a clear and elegant reference management solution, each physical object, bounding-box, sensor, and sensor measurement in AVstack is accompanied by a `calibration` and/or `origin` field. These are handled automatically by AVstack for the supported datasets (KITTI, nuScenes, CARLA).

Translations, vectors, rotations, and transformations are always relative to a reference coordinate frame. AVstack innovatively defines the reference frame as the tuple $Ref := (Tr, R, Ref\text{-}P, H)$, with Tr a translation, R a rotation, $Ref\text{-}P$ a parent reference frame (for chained reference-frames, e.g., detection-to-sensor-to-ego-to-world), and H the handedness of the axes. (Tr, R) form the `origin` field. We illustrate AVstack's approach for chained reference frames using a pass-by-reference approach in Fig. 4. We call this approach the *Reference Frame Chain of Command* (RefChoc). The RefChoc is the most reliable way to-date to support both simple cases of chaining (e.g., detection-to-sensor-to-ego-to-world) and complex cases

Source	Vehicle Frame	Ego Origin	Object Origin	Rotation	Sensors (#, Rate)	KeyFrame Rate
KITTI Object	RDF	N/A	Box Bottom	Euler (1D)	Camera (4, 10Hz), LiDAR (1, 10Hz)	10Hz
KITTI Raw	FLU	N/A	Box Bottom	Euler (1D)	Camera (4, 10Hz), LiDAR (1, 10Hz)	10Hz
KITTI Odometry	FLU	Camera 0	N/A	DCM	Camera (4, 10Hz), LiDAR (1, 10Hz)	10Hz
nuScenes	FLU	GP Rear Axle	Box Center	Quaternion	Camera (6, 12Hz), LiDAR (1, 20Hz), Radar (5, 13Hz), GPS/IMU (1, 1000Hz)	2Hz
Waymo	FLU	Ego Center	Box Center	Euler	Camera (5, 10Hz), Main LiDAR (1, 10Hz), Peripheral LiDAR (4, 10Hz)	10Hz
CARLA	FRU	GP Ego Center	Box Center	Euler (3D)	Many (user-specific)	N/A
TORCS	FL(U)	BEV Ego Center	BEV Center	Euler (1D)	Many (user-specific)	N/A
AVstack	Any	Any	Any	Any	Any	Any

Table 1: Minor differences in data design become major headaches for the developer. The ego reference and other objects can be specified with different coordinates, sensor origin, and rotation conventions. Each data source uses different sensors of varying rates and different attachments. AVstack handles transformations automatically. GP - ground projected; RDF - right, down, forward; FLU - forward, left, up; FRU - forward, right, up; DCM - direction cosine matrix; BEV - bird's eye view.

(e.g. multi-sensor, multi-agent) equally while implicitly handling coordinate transformations for the user to mitigate error-prone manual calculations.

3.2.2 Design Goal 2.2: Relevant Metrics & Evaluations

Motivation. Metrics facilitate quantitative assessment of an autonomy stack's performance. Many popular self-driving and computer vision benchmarks (e.g., [10, 15]) provide metrics at the component-level such as camera perception mean-average-precision (mAP), LiDAR perception mAP, tracking performance, prediction accuracy. These follow the hypothesis that improving individual components will lead to improved AVs in the aggregate.

The sum-of-its-parts argument neglects cross-cutting interactions and trade-offs that exist at the intersection between components. For example, many perception metrics neglect model runtime and the impact of latency on path planning and control. Similarly, improving individual components ignores inter-component error propagation; e.g., mAP takes the mean AP over all classes while not all classes impact motion prediction or path planning equally.

Design Goal. In response to the shortcomings of single-component metrics, we quantify performance at multiple components simultaneously, similar to [16]. `AVstack` provides a large selection of metrics at each level of the pipeline including the Responsibility Sensitive Safety (RSS) metric [32]. A select list of the supported metrics can be found in Table 2. Maintaining a broad set of metrics for longitudinal scenarios helps pursue:

1. **Cross-Cutting Interactions:** AV designers cannot ignore the interactions between components and the error propagation that exist when designing a longitudinal agent.
2. **Longitudinal Analysis:** Single-frame examples from datasets are incapable of validating the full performance of AVs due to their complex temporal behavior.
3. **Safety Evaluation:** Paradoxically, safety is both a primary method of regulating autonomy [13] and woefully under-utilized in quantitatively evaluating AVs.

3.3 Design Goal 3: Modular Testing

`AVstack` enables expanded AV lifecycle analysis. We describe how `AVstack`'s design enables for the first time reconfigurable architectures, expanded evaluations, streamlined model training, and multi-sensor, multi-agent configurations.

Module	**Metric**
Perception	False Positive Rate (FPR), Precision, mAP False Negative Rate (FNR), Recall, IoU
Tracking	IoU, False Track Rate (FTR) Missed Track Rate (MTR) [7] Higher Order Tracking Accuracy (HOTA) [23] CLEAR [8], VACE [25], IDEucl [23]
Prediction	Average Displacement Error (ADE) [26] Final Displacement Error (FDE) [6]
Planning	Responsibility Sensitive Safety (RSS) [32] Path KL Divergence [10]
Control	Responsibility Sensitive Safety (RSS) [32] CARLA Leaderboard Benchmark [14]

Table 2: `AVstack` unifies metrics for longitudinal testing while previous works only tested isolated components. `AVstack` uniquely incorporates the RSS safety metric.

3.3.1 Design Goal 3.1: Reconfigurable Architectures

Motivation. Many open platforms constrain users to purely module-based [16] or purely end-to-end [27], which limits software reusability and next-level evaluations. Pylot, Apollo, and Autoware have rigid architectures (green lines in Fig. 5) due to their low-level message passing. Changing architecture is difficult in all cases, and changing implementation in Apollo and Autoware is very challenging. It is more difficult to perform trade studies comparing sensors, to incorporate new sensors, and to consider novel AV architectures. These factors contribute to stagnation in AV development.

Design Goal. Components are the backbone of computation in AVs. In contrast to other platforms, `AVstack` enables *any* connection between components with its reconfigurable design. The reconfigurable architecture cuts software complexity at the expense of real-time guarantees. Fig. 5 illustrates that `AVstack` opens up "non-traditional" connections between modules.

Imperatively, `AVstack`'s design philosophy disassociates implementation from platform. Thus, components are reusable between and among datasets and simulators. We illustrate in Fig. 6 the flow of data. Simulator and dataset interfaces are standardized around base classes with common methods to get sensor data and object labels. The API is flexible enough to serve as the interface for all

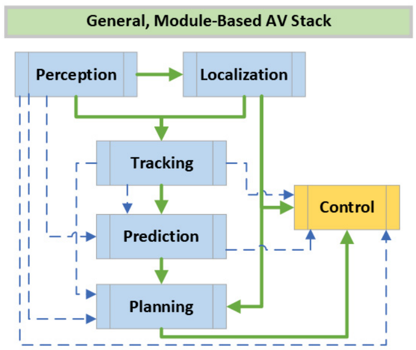


Figure 5: `AVstack` configuration is modular: any connection between modules is feasible. Breaking traditional constraints, any connection between *components* is also feasible. Module names illustrate classic module-based AV design. Traditional connections in bold green; new connections in dashed blue.

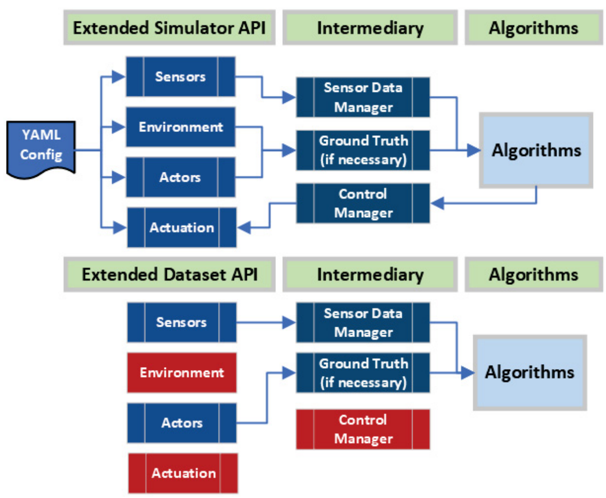


Figure 6: `AVstack` partitioned into API, intermediary, and algorithm modules to support reusability and portability of components. APIs built on common framework to allow first-of-a-kind dataset↔simulator transfer testing.

data sources. This supports early-stage development on captured datasets with longitudinal testing on end-to-end simulators.

3.3.2 Design Goal 3.2: Expanded Evaluations

Motivation. Our meta-analysis from Table 7 (Appendix B) suggests that transfer testing of algorithms is too difficult with existing tools. Too few works perform testing on multiple large, complex datasets. Moreover, an even smaller set of works perform longitudinal analysis of inter-component error propagation. At the same time, simulators including CARLA do not provide sufficient resources to bootstrap AV implementations for longitudinal testing.

Design Goal. `AVstack` greatly expands evaluations for AVs. It enables dataset-to-dataset, dataset-to-simulator, and simulator-to-simulator transfer testing. AVs can be designed for static dataset, passive longitudinal, or active longitudinal (i.e., with control) self-driving scenarios. To show the deep level of insight made possible by `AVstack`, we present metrics from a large trade study across 5 AV configurations in Section 4.1.

3.3.3 Design Goal 3.3: Streamlined Learning

Motivation. Supervised learning is a critical piece of AVs. Many modules including perception and path planning rely on learned components to perform fast and accurate inference on sensor data. A major challenge of learning-based techniques is that retraining is fraught with errors when trying to adapt datasets. Moreover, even mature simulations have limited ways to generate labeled training data from the simulator, even as a ground-truth oracle. There is no way to natively capture ground truth object labels in view of a sensor and unoccluded by buildings.

Design Goal. To aid the supervised learning process for AVs, we leverage mature modular infrastructures for supervised and reinforcement learning. `AVstack` uses MMLab's [5] open-source training infrastructure and provides a custom `AVstack` dataset interface to train and test dozens of perception models. We also provide a methodology for generating training data from the CARLA simulator. `AVstack` implements much-needed automated methods for cleaning CARLA data such as field-of-view estimation, occlusion categorization, and bounding box projection to address critical barriers in the adoption of CARLA for realistic self-driving. In Section 4.2, we illustrate how this data generation process can be configured to generate complex multi-agent scenarios and collaborative V2V, V2I sensing data for model training and testing. This allows for creation of large volumes of collaborative perception data with consistent ground truth labels between multiple viewpoints.

3.3.4 Design Goal 3.4: Multi-Sensor, Multi-Agent Systems

Motivation. Multi-sensor and multi-agent testing are part of a critical wave of next-level challenges for AVs [13]. As investments in smart infrastructure are considered, it is critical to evaluate the trade-offs in collaborative configurations. However, there are several barriers to testing both cases. Multi-sensor testing is difficult because sensor data always requires transformations between reference frames and may be configured with partially overlapping fields of view. Unfortunately, it is error-prone to leave multi-sensor configuration up to the developer; yet few public platforms provide effective multi-sensor support. Multi-agent testing has also yet to be sufficiently realized. The majority of evaluations in self-driving have focused on static datasets that lack multi-agent information. Similarly, even in simulator environments, mature AV research platforms have constrained architectures and components. This limited modularity means that adding new sensor data, integrating new components, and designing new algorithms is burdensome.

Design Goal. To solve the sensor data and reference-frame challenges in multi-sensor/multi-agent configurations, `AVstack` has several important innovations. First, reference frame transformations can be performed automatically by specifying a start and end-point reference. This removes error-prone coordinate transformations (e.g., *object-to-sensor1-to-ego-to-sensor2* for multi-sensor; *sensor1-to-agent1-to-world-to-agent2-to-sensor2* for multi-agent). Second, `AVstack` has a growing list of sensors to which it offers compatibility. In the simulator context, `AVstack` bootstraps ego and sensor classes with clearer and developer-friendly configurations to support existing simulator features. Third, architecture design is

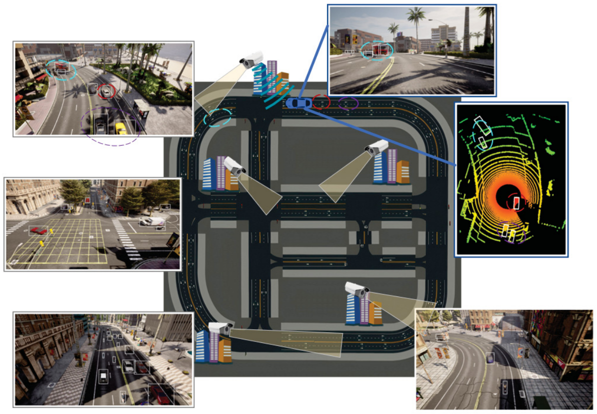


Figure 7: AVstack bootstraps CARLA for multi-sensor, multi-agent configurations, opening possibility for next-generation AV DITA. Here, camera sensors are placed at infrastructure locations to provide collaborative view. (top right) Ego vehicle with local camera and LiDAR: object detections in white boxes. (top left) Infrastructure sensor detects objects in camera corresponding to objects from ego: cyan, red, purple ovals.

modular in AVstack. Components from a single-agent AV can be reused in multi-agent contexts. Single-agent AVs can be evaluated against multi-agent AVs in a unified simulation framework (Fig. 7).

3.4 Design Goal 4: Low Barrier

Motivation. Apollo, Autoware, and Pylot are mature AV platforms but all have a high barrier to entry. All rely on high-performance message passing frameworks to deliver low-latency sensor data at the cost of architecture flexibility. Source code for Apollo and Autoware is complex and rigid. They are targeted to full-stack AVs that ingest sensor data and output control decisions. This makes debugging individual algorithms and components incredibly difficult; changing AV architecture is exceptionally challenging.

Design Goal. AVstack provides a low-barrier and flexible AV testing framework. For the first time, there is compatibility between datasets and simulators. At the intermediary between data and algorithms are thread-safe data structures that handle flexible routing of data from source to destination in a high-level programming language. Our no-copy philosophy allows data to be transferred efficiently to support near-real-time execution; however, data are handled with the utmost flexibility for the user. An object-oriented approach allows sensor data to be efficiently routed with multiple end-points. In Section 4, we provide case studies using just dozens of lines of code on top of AVstack to create unique AVs and diverse testing environments.

4 Use Case Experiments

In this section, we show how AVstack enables important exploration, trade studies, and analysis at low development cost.

4.1 Portability and Transfer Testing

Two major causes of a slowdown in AV development are poor infrastructures for transfer testing between datasets & simulators,

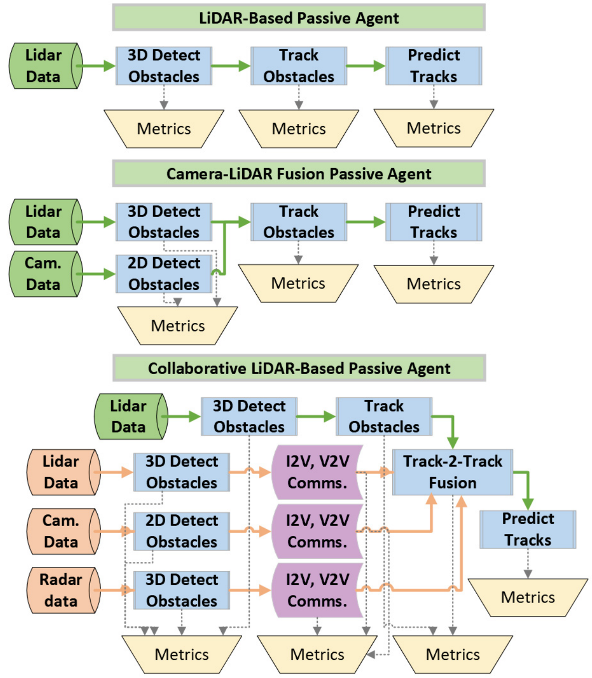


Figure 8: LiDAR-only, camera-LiDAR, and collaborative LiDAR agents require only 15, 20, and 30 lines of code at the top level to instantiate. Implementations can run on datasets & simulators allowing for insightful and rapid trade studies.

and limited longitudinal evaluations. The ability to perform algorithm testing across data sources is vital for validation of complex components. Running longitudinal evaluations helps understand cross-component error propagation, which is lacking in single-component analysis.

To demonstrate that AVstack enables transferability between data sources, we design passive agents using LiDAR-based and camera-LiDAR fusion component architectures (e.g., as in [17, 18]) shown in Fig. 8. We can use AVstack to create these dataset-agnostic agents using just 15 and 20 line of code. We call these "passive" because we leave out planning and control – a capability made possible by AVstack's reconfigurable design. Within the two architectures, we test different combinations of algorithms to form five different implementations. The complete case study specification is represented with a "configuration table" in AVstack, as illustrated in Table 3. With this configuration table, AVstack evaluates the different AV implementations over KITTI, nuScenes, and CARLA on 10 randomly sampled longitudinal sequences. During each run, AVstack captures per-frame and per-sequence metrics that were summarized in Table 2.

AVstack's output of the trade study is a set of detailed per-frame and per-case results (not shown) and an aggregated benchmark table; see Table 4. Videos of select sequences can be found online [2]. AVstack's breadth and depth of measurements make it useful for component-wise analysis of AVs. In this case study, we find 3D object precision is high across all algorithms and all datasets; however, recall on nuScenes is low. Similarly, nuScenes tracking performance (HOTA) is lower compared to KITTI and CARLA.

ID	LiDAR Percep	Cam Percep	Tracking	Prediction
0	PointPillars [22]	N/A	AB3DMOT [36]	Kinematic
1	3DSSD [24]	N/A	AB3DMOT [36]	Kinematic
2	PointPillars [22]	FasterRCNN [29]	EagerMOT [21]	Kinematic
3	3DSSD [24]	FasterRCNN [29]	EagerMOT [21]	Kinematic
4	PointPillars [22]	Cascade MaskRCNN [19]	EagerMOT [21]	Kinematic

Table 3: AVstack enables transferability between data sources. uses configuration tables to run trade studies. Modules are widely compatible with community implementations. Components are dataset-agnostic. The 5 case studies here are used for the transfer test in Section 4.1.

4.2 Multi-Sensor, Multi-Agent Collaboration

While multi-sensor, multi-agent configurations are imperative for next-generation AV evaluations, they are difficult to design and test using today's available platforms. Some recent works have begun to analyze cooperative settings using ad-hoc development environments [12, 38]. Previous evaluation platforms have leveraged existing datasets to run experiments. Usefully, AVstack is not tied to an individual dataset. Rather, the AVstack API provides a flexible and easy to use approach to leverage existing datasets and to generate *any* scenario, including multi-sensor, multi-agent configurations, in the CARLA simulator.

We use AVstack to design a collaborative agent with an architecture similar to Fig. 8(c). The agent possesses a LiDAR sensor with a limited range of 25 m. To obtain sufficient situational awareness, the agent must use information from nearby infrastructure sensors to complement its own limited sensing information. We do not consider planning or control components and instead investigate the agent just using perception, tracking, and prediction performance.

We use the AVstack API to test our multi-agent design. We place 40 64-line LiDARs with a field-of-view of 180° at random locations in CARLA's Town-10. These serve as the infrastructure sensors. Each collaborative sensor is placed at a 30° pitch angle and a height of 15 m to obtain an appropriate viewing angle. We chose to use LiDAR sensors to simplify 3D positioning, but any and all sensors in CARLA can be used, including cameras and radars.

With this configuration, we design two trade study experiments to evaluate the trade-offs between (1) sensor communication range and detection accuracy, and (2) sensor rate and detection accuracy. Table 5 highlights the different configurations in this experiment.

Using the trade study capability of AVstack, we run the ego agent over the 9 collaborative cases from Table 5 on 5 randomly-generated CARLA scenes with 150 "other" vehicles. Collaborative detections are transmitted from sensor to agent at the specified data rate. Upon receiving messages, the agent first performs preprocessing to ignore any detections outside of a 100 m radius, for computational efficiency. The agent then integrates detections with data association, assigns measurements to existing tracked objects, and spawns new tracks with unassigned detections. Additional configuration details can be found at [2] as well as in Appendix D.

At the culmination of the study, AVstack generates aggregated results tables, shown in Table 6. Videos of select sequences can be found online [2]. We find that collaborative sensing can aid an agent, particularly in this case where the ego's sensor range was limited. In Table 6-A, we find the HOTA metric is highest (best) for C1-Ideal and C1-1. Also, prediction error, ADE and FDE, are lower with collaboration compared to C1-base. In Table 6-B, we find that tracking performance does not deteriorate when trading sensor rate from 10 Hz to 5 Hz for a decrease in detection noise - HOTA remains constant among all cases. While differences in prediction errors, ADE and FDE, are not significantly different between cases C2-1 and C2-2, it is worth investigating in more detail the impact of sensor rate on prediction performance.

5 Conclusion

We have introduced AVstack as an open-source, reconfigurable software platform for AV design, implementation, test, and analysis. We have illustrated in several case studies that AVstack supports rapid prototyping of reusable AV components, longitudinal evaluations with component-wise metrics, and diverse multi-sensor, multi-agent configurations. AVstack delivers solutions to the most common challenges faced by AV users with its bank of community-support components, by bridging convention conflicts among datasets and simulators, by supporting algorithm reuse with dataset-agnostic and flexible components, and by delivering much-needed support for next-level analysis. Its key design principles will help accelerate the push toward important AV milestones. In several case studies focusing on portability and transfer testing, as well as testing of multi-sensor, multi-agent collaboration, we have illustrated these benefits of the use of AVStack.

Acknowledgments

This work is sponsored in part by the ONR under the agreements N00014-20-1-2745 and N00014-23-1-2206, AFOSR award number FA9550-19-1-0169, NSF CNS-1652544 award as well as the National AI Institute for Edge Computing Leveraging Next Generation Wireless Networks, Grant CNS-2112562.

References

[1] [n. d.]. Autoware. https://www.autoware.org/.
[2] [n. d.]. AVstack. https://www.avstack.org/research.
[3] [n. d.]. Baidu Apollo. apollo.auto.
[4] [n. d.]. CARLA Leaderboard. https://app.alphadrive.ai/benchmarks/3/overview.
[5] 2020. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d.
[6] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. 2016. Social lstm: Human trajectory prediction in crowded spaces. In *IEEE CVPR*. 961–971.
[7] Keni Bernardin and Rainer Stiefelhagen. 2008. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing* 2008 (2008), 1–10.
[8] Keni Bernardin and Rainer Stiefelhagen. 2008. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing* 2008 (2008), 1–10.
[9] Barry Brown. 2017. The social life of AV cars. *Computer* 50, 2 (2017), 92–96.
[10] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, et al. 2020. nuscenes: A multimodal dataset for autonomous driving. In *IEEE/CVF CVPR*. 11621–11631.
[11] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. 2020. Learning by cheating. In *CoRL*. PMLR, 66–75.
[12] Hanlin Chen, Brian Liu, Xumiao Zhang, Feng Qian, Z Morley Mao, and Yiheng Feng. 2022. A Cooperative Perception Environment for Traffic Operations and Control. *arXiv preprint arXiv:2208.02792* (2022).
[13] Missy Cummings. 2017. The Brave new world of Driverless cars. *TR News* 308 (2017), 34–7.
[14] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. CARLA: An open urban driving simulator. In *CoRL*. PMLR, 1–16.
[15] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. 2013. Vision meets robotics: The kitti dataset. *The Int. Journal of Robotics Research* 32, 11 (2013), 1231–1237.

Case	Data	Per: 3D Prec.	Per: 3D Rec.	Per: 2D Prec.	Per: 2D Rec.	Trk: HOTA	Trk: MOTA	Trk: MOTP	Pred: ADE	Pred: FDE
	K	0.37 +/- 0.24	0.90 +/- 0.21	N/A	N/A	0.52 +/- 0.19	-0.03 +/- 2.48	3.62 +/- 0.10	1.33 +/- 0.94	3.87 +/- 1.41
0	N	0.99 +/- 0.01	0.25 +/- 0.07	N/A	N/A	0.11 +/- 0.04	0.20 +/- 0.07	2.68 +/- 0.16	0.26 +/- 0.08	0.26 +/- 0.08
	C	0.99 +/- 0.00	0.77 +/- 0.02	N/A	N/A	0.51 +/- 0.07	0.48 +/- 0.05	2.85 +/- 0.14	4.99 +/- 3.65	11.55 +/- 5.37
	K	0.25 +/- 0.19	0.39 +/- 0.17	N/A	N/A	0.40 +/- 0.16	-0.15 +/- 0.32	3.07 +/- 0.73	1.20 +/- 0.78	1.86 +/- 1.77
1	N	1.00 +/- 0.02	0.19 +/- 0.06	N/A	N/A	0.09 +/- 0.04	0.12 +/- 0.05	2.75 +/- 0.13	0.31 +/- 0.06	0.31 +/- 0.06
	C	0.99 +/- 0.00	0.68 +/- 0.05	N/A	N/A	0.46 +/- 0.07	0.43 +/- 0.04	2.82 +/- 0.13	5.20 +/- 3.33	12.11 +/- 4.91
	K	0.37 +/- 0.24	0.90 +/- 0.21	0.31 +/- 0.18	0.73 +/- 0.20	0.71 +/- 0.13	0.60 +/- 0.22	3.77 +/- 0.17	0.77 +/- 0.58	1.78 +/- 1.67
2	N	0.69 +/- 0.18	0.32 +/- 0.02	0.90 +/- 0.04	0.52 +/- 0.11	0.11 +/- 0.04	0.11 +/- 0.07	2.89 +/- 0.30	1.05 +/- 0.43	1.05 +/- 0.43
	C	0.62 +/- 0.13	0.88 +/- 0.06	0.40 +/- 0.25	0.13 +/- 0.09	0.12 +/- 0.05	0.08 +/- 0.05	3.00 +/- 0.30	1.26 +/- 0.54	3.56 +/- 0.70
	K	0.25 +/- 0.19	0.39 +/- 0.17	0.31 +/- 0.18	0.73 +/- 0.20	0.46 +/- 0.18	0.34 +/- 0.14	2.98 +/- 0.55	0.61 +/- 0.69	1.21 +/- 1.95
3	N	0.67 +/- 0.19	0.24 +/- 0.03	0.90 +/- 0.04	0.52 +/- 0.11	0.09 +/- 0.03	0.07 +/- 0.03	2.93 +/- 0.29	0.63 +/- 0.59	0.63 +/- 0.59
	C	0.54 +/- 0.13	0.69 +/- 0.09	0.40 +/- 0.25	0.13 +/- 0.09	0.10 +/- 0.05	0.06 +/- 0.05	2.99 +/- 0.28	1.44 +/- 0.41	3.38 +/- 0.54
	K	0.37 +/- 0.24	0.90 +/- 0.21	0.29 +/- 0.18	0.95 +/- 0.23	0.70 +/- 0.08	0.59 +/- 0.08	3.77 +/- 0.12	0.86 +/- 0.23	1.51 +/- 1.12
4	N	0.69 +/- 0.18	0.32 +/- 0.02	0.78 +/- 0.05	0.72 +/- 0.08	0.12 +/- 0.03	0.10 +/- 0.08	2.89 +/- 0.10	1.06 +/- 0.33	1.06 +/- 0.33
	C	0.62 +/- 0.13	0.88 +/- 0.06	0.93 +/- 0.02	0.60 +/- 0.07	0.36 +/- 0.09	0.30 +/- 0.07	3.01 +/- 0.07	2.68 +/- 1.32	5.17 +/- 3.80

Table 4: AVstack enables first-of-a-kind trade studies simply by specifying a configuration table such as Table 3. Results are averaged over 10 longitudinal trials using the centrall mounted LiDAR and forward-facing camera. Each trial is over a 20 second scene for each dataset (K: KITTI, N: nuScenes, C: CARLA). Each AV configuration "Case" is described in Table 3. For the first time, metrics can be computed at each level of the pipeline (Per. 2D/3D: 2D or 3D Perception, Trk.: Tracking, Pred.: Prediction) at the same time to illuminate error propagation between modules. Best performance is highlighted per-cell.

Table 5: Collaborative Experiment Design

Case	LiDAR Percep	Cam Percep	Tracking	Prediction
All	PointPillars [22]	N/A	AB3DMOT [36]	Kinematic

Panel A: AV configuration constant for all collaborative studies.

Case	Density	Det. Type	Comm Range	Det. Rate	Det. Noise
C1-Ideal	40/map	3D Box	100 m	10 Hz	None
C1-1	40/map	3D Box	100 m	10 Hz	High
C1-2	40/map	3D Box	70 m	10 Hz	Med
C1-3	40/map	3D Box	50 m	10 Hz	Low
C1-Base	40/map	N/A	N/A	N/A	N/A

Panel B: Experiment (C1) trading comm range for noise.

Case	Density	Sensor	Comm Range	Det. Rate	Det. Noise
C2-Ideal	40/map	3D Box	80 m	10 Hz	None
C2-1	40/map	3D Box	80 m	10 Hz	High
C2-2	40/map	3D Box	80 m	5 Hz	Low
C2-Base	40/map	N/A	N/A	N/A	N/A

Panel C: Experiment (C2) tests communication rate vs. noise.

[16] Ionel Gog, Sukrit Kalra, Peter Schafhalter, Matthew A Wright, Joseph E Gonzalez, and Ion Stoica. 2021. Pylot: A modular platform for exploring latency-accuracy tradeoffs in autonomous vehicles. In *ICRA*. 8806–8813.

[17] R Spencer Hallyburton, Yupei Liu, Yulong Cao, Z Morley Mao, and Miroslav Pajic. 2022. Security analysis of camera-lidar fusion against black-box attacks on autonomous vehicles. In *31st USENIX Security Symposium (USENIX SECURITY)*.

[18] R Spencer Hallyburton and Miroslav Pajic. 2023. Securing Autonomous Vehicles Under Partial-Information Cyber Attacks on LiDAR Data. *arXiv preprint arXiv:2303.03470* (2023).

[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*. 2961–2969.

[20] David M Henderson. 1977. *Euler angles, quaternions, and transformation matrices for space shuttle analysis*. Technical Report.

[21] Aleksandr Kim, Aljoša Ošep, and Laura Leal-Taixé. 2021. Eagermot: 3d multi-object tracking via sensor fusion. In *ICRA*. 11315–11321.

[22] Alex Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. 2019. Pointpillars: Fast encoders for object detection from point clouds. In *IEEE/CVF CVPR*. 12697–12705.

[23] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. 2021. Hota: A higher order metric for evaluating multi-object tracking. *International journal of computer vision* 129, 2 (2021), 548–578.

[24] Qianhui Luo, Huifang Ma, Li Tang, Yue Wang, and Rong Xiong. 2020. 3d-ssd: Learning hierarchical features from rgb-d images for amodal 3d object detection. *Neurocomputing* 378 (2020), 364–374.

[25] Andrii Maksai, Xinchao Wang, Francois Fleuret, and Pascal Fua. 2017. Non-markovian globally consistent multi-object tracking. In *Proceedings of the IEEE international conference on computer vision*. 2544–2554.

[26] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. 2009. You'll never walk alone: Modeling social behavior for multi-target tracking. In *Proceedings of the IEEE international conference on computer vision*. 261–268.

[27] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. 2021. Multi-modal fusion transformer for end-to-end autonomous driving. In *IEEE/CVF CVPR*. 7077–7087.

[28] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. 2009. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, Vol. 3. 5.

[29] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Conference on Neural Information Processing* 28 (2015).

[30] Guna Seetharaman, Arun Lakhotia, and Erik Philip Blasch. 2006. Unmanned vehicles come of age: The DARPA grand challenge. *Computer* 39, 12 (2006), 26–29.

[31] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. 2018. Airsim: High-fidelity visual and physical sim. for AVs. In *Field and service robotics*. Springer, 621–635.

[32] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. 2017. On a formal model of safe and scalable self-driving cars. *arXiv preprint arXiv:1708.06374* (2017).

[33] Pei Sun, Henrik Kretzschmar, et al. 2020. Scalability in perception for autonomous driving: Waymo open dataset. In *IEEE/CVF CVPR*. 2446–2454.

[34] Victor Tangermann. 2022. Watch Elon Musk promise self-driving cars "next year" every year since 2014. https://futurism.com/video-elon-musk-promising-self-driving-cars

[35] Marin Toromanoff, Emilie Wirbel, and Fabien Moutarde. 2020. End-to-end model-free reinforcement learning for urban driving using implicit affordances. In *IEEE/CVF CVPR*. 7153–7162.

[36] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 2020. Ab3dmot: A baseline for 3d multi-object tracking and new evaluation metrics. *arXiv preprint arXiv:2008.08063* (2020).

[37] Bernhard Wymann, Eric Espié, Christophe Guionneau, Christos Dimitrakakis, Rémi Coulom, and Andrew Sumner. 2000. Torcs, the open racing car simulator. *Software available at http://torcs. sourceforge. net* 4, 6 (2000), 2.

[38] Runsheng Xu, Hao Xiang, Xin Xia, Xu Han, Jinlong Li, and Jiaqi Ma. 2022. Opv2v: An open benchmark dataset and fusion pipeline for perception with vehicle-to-vehicle communication. In *2022 ICRA*. 2583–2589.

Table 6: Collaborative Vehicle-to-Infrastructure Case Study Results.

Case	Data	Collab: Sensors-in-range/frame	Collab: Dets/frame	Trk: HOTA	Trk: MOTA	Trk: MOTP	Pred: ADE	Pred: FDE
C1-Ideal	C	13.00 +/- 3.30	122.00 +/- 63.69	0.55 +/- 0.16	-0.40 +/- 0.61	3.10 +/- 0.08	1.19 +/- 0.53	3.23 +/- 1.86
C1-1	C	13.00 +/- 3.30	122.00 +/- 63.69	0.52 +/- 0.14	-0.46 +/- 0.64	2.94 +/- 0.06	0.86 +/- 0.29	2.65 +/- 1.23
C1-2	C	5.00 +/- 2.45	55.00 +/- 35.72	0.32 +/- 0.14	-0.90 +/- 1.55	2.92 +/- 0.09	1.41 +/- 0.14	3.64 +/- 0.46
C1-3	C	2.00 +/- 1.89	20.00 +/- 41.96	0.54 +/- 0.12	-0.24 +/- 0.47	2.99 +/- 0.05	1.07 +/- 0.24	2.65 +/- 0.75
C1-Base	C	N/A	N/A	0.47 +/- 0.09	0.35 +/- 0.07	3.10 +/- 0.10	2.42 +/- 1.99	6.40 +/- 3.26

Panel A: Trading communication range for detection accuracy over 10 trials of 500 frames in CARLA.

Case	Data	Collab: #S-in-range/frame	Collab: Dets/frame	Trk: HOTA	Trk: MOTA	Trk: MOTP	Pred: ADE	Pred: FDE
C2-Ideal	C	3.50 +/- 1.50	40.25 +/- 13.75	0.63 +/- 0.17	-0.06 +/- 0.68	3.04 +/- 0.07	1.36 +/- 0.45	3.36 +/- 1.31
C2-1	C	3.50 +/- 1.50	40.25 +/- 13.75	0.61 +/- 0.18	-0.08 +/- 0.69	2.90 +/- 0.07	0.98 +/- 0.14	2.23 +/- 0.60
C2-2	C	3.50 +/- 1.50	18.50 +/- 6.00	0.60 +/- 0.18	-0.09 +/- 0.67	2.88 +/- 0.06	0.94 +/- 0.01	2.05 +/- 0.13
C2-Base	C	N/A	N/A	0.66 +/- 0.08	0.54 +/- 0.07	3.02 +/- 0.10	1.60 +/- 0.56	5.74 +/- 0.74

Panel B: Trading communication rate for detection accuracy over 10 trials of 500 frames in CARLA.

A Intentional Design Omissions

No platform can satisfy the requirements of all AV use-cases because some are in conflict. For example, introducing architecture modularity can sacrifice real-time performance. To address some of the critical barriers to AV development, *a modular research platform is essential and lacking*.

We are faced with fundamental architecture questions for multi-sensor, multi-agent AVs where industry is dramatically outpacing research. For the next generation of smart vehicles, insightful DITA must be prioritized. To do so in an expeditious manner, there must be a low barrier to entry, even if this means sacrificing other qualities. In particular, `AVstack` intentionally places less emphasis on the following areas:

- **Real Time:** `AVstack` is not proposed as a real-time solution. We have not performed experiments evaluating latency. Attempting to package `AVstack` as a real-time AV may require a real-time operating system and low-latency data passing which would negatively affect modularity.
- **Low-Level Programming:** `AVstack` is based on Python to allow for rapid prototyping and easy interfacing to third-party simulation engines. It was not written with speed or memory as a primary goal, in contrast to higher-barrier autonomy stacks Apollo and Autoware.

B State of the Art Perception

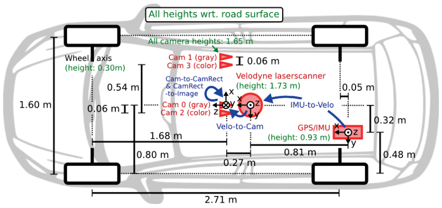
The KITTI dataset [15] was instrumental in the progress of AV perception development. Since, KITTI's original release, major players including Waymo and Motional have released datasets more extensive than KITTI with multiple sensing modalities [10, 33]. Unfortunately, we find that even recent state-of-the-art perception algorithms neglect to provide sufficient evaluation on these more challenging datasets. To investigate, we scraped perception benchmark leaderboards, as described in Section 3.1.1. The findings of this meta-analysis are in Table 7. Of 18 validated entries in the top 50 on KITTI, many are recent, and progress between them has been marginal at only 3% gained. Unfortunately, even these recent works neglect cross-dataset evaluations, leading to challenges with reproducibility and translational success in contexts such as simulators and real AVs.

Friendly Name	Year	KITTI	nuScenes	Waymo
Sparse Fuse Dense	2022	Y (84.8)	N	N
CasA	2022	Y (84.0)	N	Y (78.3/69.6)
GLENet	2022	Y (83.2)	N	Y (77.3/69.7)
VPFNet	2022	Y (83.2)	N	N
Graph R-CNN	2022	Y (83.2)	N	Y (72.6/72.1)
BtcDet	2022	Y (82.9)	N	Y (78.6/70.1)
SPG	2021	Y (82.7)	N	Y
SE-SSD	2021	Y (82.5)	N	N
DVF	2022	Y (82.5)	N	Y (67.6/62.7)
RDIoU	2022	Y (82.3)	N	Y (78.4/69.5)
FocalsConv	2022	Y (82.3)	Y (70.1)	Y (72.2/64.1)
CLOCs	2020	Y (82.3)	N	N
SASA	2022	Y (82.2)	Y (45)	N
VoTr	2021	Y (82.1)	N	Y (69.0/60.2)
Pyramid R-CNN	2021	Y (82.1)	N	(76.3/67.0)
VoxSet	2022	Y (82.1)	N	Y (77.9/70.2)
SRIF-RCNN	2022	Y (82.0)	N	N
Q-Net	2022	Y (82.0)	N	N

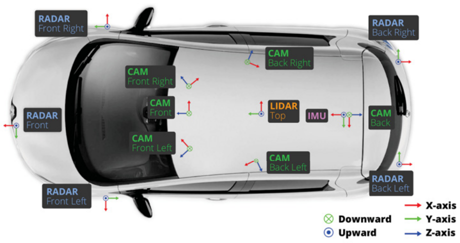
Table 7: Recent publications atop the KITTI leaderboard are not always cross-validated against other, larger datasets. The nuScenes dataset has limited adoption. Continued testing on KITTI has only achieved marginal improvements on already high performing marks.

C KITTI, nuScenes, Waymo Configurations

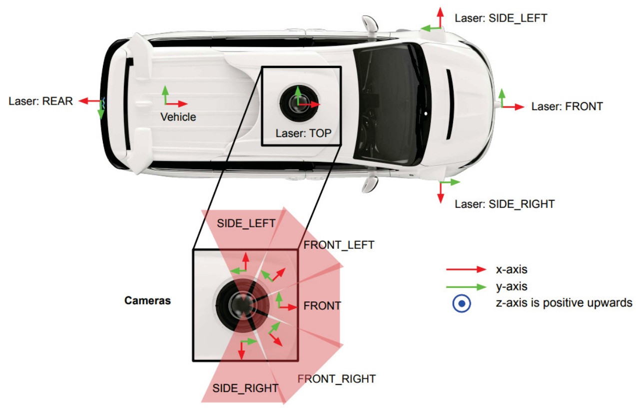
The release of high-fidelity benchmark datasets from major research institutions and prominent industry players has significantly contributed to a boom in AV algorithm development. Large datasets



(a) KITTI [15]



(b) nuScenes [10]



(c) Waymo [33]

Figure 9: Configurations from different AV data source providers are all unique; each dataset has its own sensor types, sensor orientations, reference frames, and data rates. Evaluating components across all sources leads to insightful results.

like nuScenes [10] and Waymo's Open Dataset [33] have garnered attention recently for their challenging mix of weather conditions and multiple complementary sensing modalities.

Despite their contributions to the field, no platform has managed to unify the datasets under an tractable umbrella. This is in part due to the intricacy and uniqueness of each platform itself. To help illuminate why unifying these datasets under a common interface is challenging, we provide the sensor configurations for KITTI [15], nuScenes, and Waymo's open dataset in Figure 9.

D Configuration of Collaborative Case Study

The vehicle-to-infrastructure (V2I) collaborative case study of Section 4.2 provides a framework for future efforts to develop multi-agent components and design smart cities. In this section, we provide additional details on the specific parameters used. These details can also be found in the source code online at [2]. We used LiDAR sensors as our infrastructure sensors. In pre-processing, we determined if objects were in the field of view of a sensor for ground-truth evaluation by using ray-tracing to filter out objects that were completely occluded (i.e., no LiDAR points in bounding box). We did so because CARLA has no alternative method, to our knowledge, of validating if an object is in view of a sensor. After pre-processing, we simulated detections from the LiDAR sensor rather than run a perception algorithm. This was solely so that we could apply our own noise model to the infrastructure detections as a trade study. Then, to simulate V2I communication, we performed range-based filtering to identify which infrastructure sensors were in-range of the ego vehicle. Detections were passed with no latency to the ego agent. The agent then fused the infrastructure detections with existing tracks in a Kalman filter with a standard assignment algorithm. We evaluated performance of the ego agent against objects in the field of view of the ego within a range of 100 m and a maximum occlusion score of "partial".