WHITE PAPER

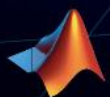
Building Key Competencies for Autonomous Vehicle Development

Simulate virtual worlds, build multidisciplinary skills, and deliver software for complex autonomous systems

Automated driving spans a wide range of automation levels, from advanced driver assistance systems (ADAS) to fully autonomous driving (AD). As the level of automation increases, the use cases become less restricted and testing requirements increase, making the need for simulating scenarios in virtual worlds more critical. Developing these automated driving applications requires multidisciplinary skills—from planning and controls to perception disciplines such as detection, localization, tracking, and fusion—in an environment that supports the design, validation, and deployment of increasingly complex software.

In this white paper, you will learn how automotive engineers can:

- Manage validation complexity by building virtual worlds and leveraging simulation
- Develop multidisciplinary skills to navigate fundamental changes to automotive engineering
- Develop software applications to meet ISO 26262



MathWorks®

ADAS/AD Development

ADAS/AD engineers ask several common questions: How can I analyze and synthesize scenarios? How can I design and deploy algorithms? And at a system level, how can I integrate and test the entire AD system?

Many times conversation on ADAS/AD development quickly gets into perception, which in turn gets into AI and AI modeling. However, ADAS/AD development is more than perception. It spans virtual worlds and requires multidisciplinary skills for both developing algorithms using multiple tools and deploying these algorithms as software applications, as shown in Figure 1.

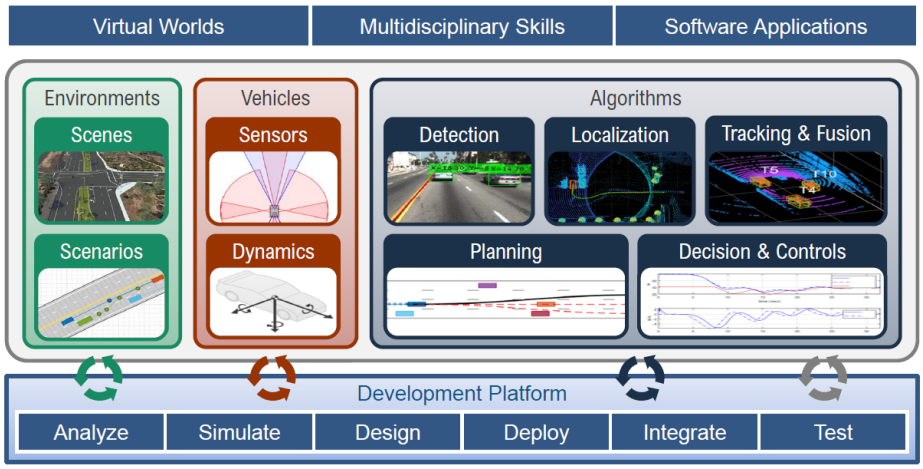


Figure 1. The three-pronged approach engineers should consider for developing AD applications.

In addition, these engineers often expect to spend a large percentage of their time developing and fine-tuning models of the environment, vehicle, and algorithms. Yes, modeling is an important step in the workflow, but the model is not the end of the journey. The key element for success in practical development of ADAS/AD applications is uncovering any issues early and knowing on which aspects of the workflow to focus time and resources for the best results.

Two important asides should be considered before diving into the typical workflow:

- ADAS and AD are multidisciplinary domains with many development tools and vendors. This, in turn, emphasizes the need for good connectors to enable setup of an integrated simulation platform. Integration permits putting together all algorithms (developed in many platforms) to perform system simulation to gain insights.
- In addition to integration, another key requirement is a tool or platform that enables easy visualization to assess performance of algorithms across the workflow.

Typical ADAS/AD Workflow

You start with creating a scene. This is followed by creating a scenario that includes the scene, actors (vehicles, pedestrians), weather, and light sources. Next, the ego vehicle needs to be modeled to include sensors that are part of the AV sensor suite along with the vehicle dynamics (for lateral control, longitudinal control, or both). With this preparation, you are now ready to begin simulating the scenario, which in turn permits iterative refinement of algorithms for perception, planning, and controls. After you gain confidence in these algorithms, you create the software. That software code is either generated automatically from tools or handwritten. Then, integrate code to perform system-level simulation to gain confidence that the code is functionally correct at the system level. Finally, run simulations as a part of testing, either interactively or automatically (on your desktop, on a cluster, or on the cloud).

Simulating Virtual Worlds

You have probably heard about the notion of running a million scenarios, enabled by simulation. And simulation needs to reflect the real world. Before you test scenarios, you need a scene to simulate in a virtual world. A scene needs to reflect the real world, which can be complicated, with a road intersection representing a challenging road scene. Consider a roundabout that can range from a reasonably simple three-entry/exit layout to a complex 12-entry/exit layout, as in the Arc de Triomphe in Paris, France.

With RoadRunner, a real-world road scene can be re-created in a fast and functional way even when this reality is quite complex (see Figure 2). For AD, roads are a critical part of the scene.

Figure 2. Re-creating a real-world scene with RoadRunner.

Re-created scenes need to be in a format such that they can be exported for use with popular simulators in the market, such as CARLA, CarMaker®, and NVIDIA® DRIVE Sim®. If you need to create long stretches of road scenes, this manual approach can be cumbersome. At this point you will benefit from having an approach that is automatic. It is now possible to import longer road sections in 3D from HERE HD Live Map.

You can author driving scenarios based on these scenes. One source for scenarios could be from recorded data. Ford developed its Active Park Assist feature through event identification

and scenario generation from recorded data. Along similar lines, GM generated scenarios from recorded vehicle data for validating lane-centering systems.

You can identify new scenarios from recorded data. In this approach, you extract information from your CAN logs or directly from a camera or a lidar. You can visualize data and then label it. The labeling can be automated using either public or custom algorithms. You then identify scenarios of interest from your recorded and labeled data to re-create simulation test cases. This process is typically an open-loop workflow.

You can also identify new scenarios from scenario variations. In this approach, you create a scenario. You then create variations and use simulations to help identify new scenarios of interest and add to your regression tests. This process enables a closed-loop workflow.

Through the above two approaches, you can identify and add new test cases into your design and simulation workflows.

Scenes and scenarios can be created either interactively or programmatically. In addition, you can:

- Import/export a scene to OpenDRIVE, among other formats
- Import OpenStreetMap® data into a scenario
- Export a scenario to OpenDRIVE
- Export a scenario to OpenSCENARIO

The fidelity of the virtual world can be chosen depending on the need for simulating specific use cases. For example, tracked detections from a radar can be used to develop planning and controls algorithms, whereas camera detections can be used to develop perception algorithms. MathWorks provides two environments for virtual worlds:

- **Cuboid:** You can use cuboid world representation to simulate driving scenarios, use sensor models, and generate synthetic data to test automated driving algorithms in simulated environments, including controls, sensor fusion, and path planning. For example, you can use this approach to identify the best location of sensors and number of sensors.
- **Unreal Engine®:** You can develop, test, and visualize the performance of driving algorithms in a 3D simulated environment rendered using the Unreal Engine from Epic Games. In addition to the algorithms noted in the cuboid world, you can develop and test perception algorithms driven by camera data from different camera models.

Figure 3 shows the sensors that are part of the typical AV sensor suite.

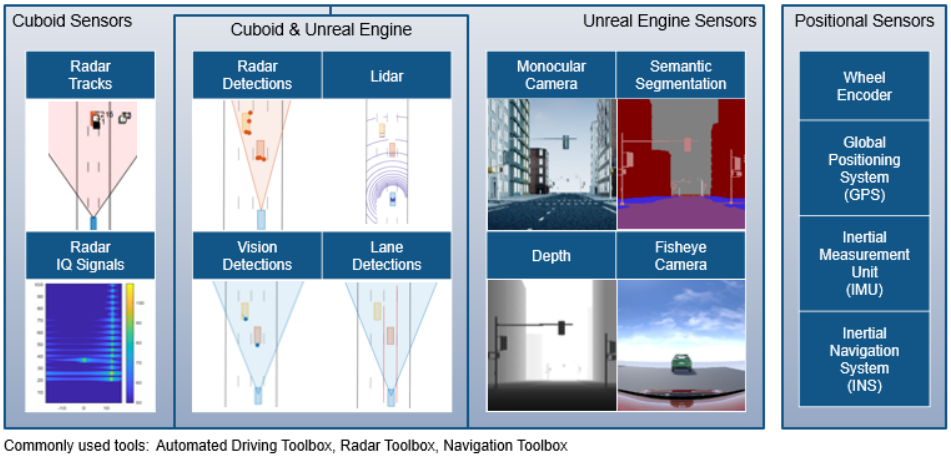


Figure 3. Simulating sensors for AD applications.

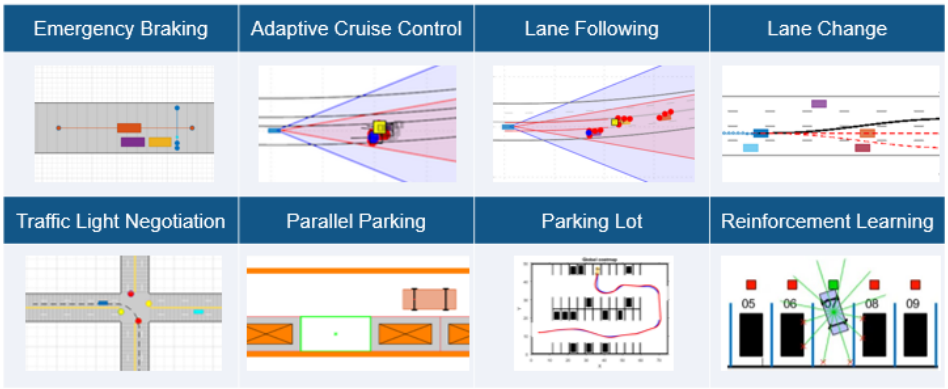
Radar, lidar, and camera sensors are used for detecting objects with sensors and detections corresponding to different simulation environments. Positional sensors can be used in both simulation environments. To simulate vehicle dynamics, you need models of multi-axle vehicles, trucks and trailers, the powertrain, steering, suspension, wheels, and tires.

To reiterate, developing virtual worlds involves creating scenes, creating scenarios, modeling sensors, and modeling vehicle dynamics. This process is scalable and gives users the flexibility to apply their domain expertise without having to become experts in other domains.

Building Multidisciplinary Skills

The multidisciplinary nature of AV development requires ADAS/AD algorithms to exist within a larger system and be interoperable with other constituents of the vehicle system. In an ADAS/AD application, not only do you have a perception system for detecting objects (pedestrians, cars, stop signs), but this system must integrate with other systems for localization, path planning, controls, and more.

Developing this complex system requires multidisciplinary skills to develop algorithms for ADAS/AD features such as adaptive cruise control, automatic emergency braking, and higher-level features such as highway lane change and automated parking/parking valet. Figure 4 shows a few examples of AD features.



Commonly used tools: Automated Driving Toolbox, Model Predictive Control Toolbox, Stateflow, Navigation Toolbox, Reinforcement Learning, Robotics System Toolbox

Figure 4. Typical planning and control algorithms for AD.

These algorithms cover planning, controls, and perception disciplines:

- **Planning and controls** includes motion planning, decision logic, and longitudinal and lateral controls.
- **Perception** includes detection, object tracking and sensor fusion, and localization.

Automated Driving Toolbox™ includes examples that serve as a framework to help you start designing your own ADAS/AD features. Engineers new to the automotive industry need an understanding of a typical automobile and its constituent subsystems including the control system. For example, they can get started quickly with control system design with Control System Toolbox™ and vehicle dynamics modeling with Vehicle Dynamics Blockset™. Given the complexity of ADAS/AD systems and fast-paced software development cycles, engineers moving into this domain from other domains can jumpstart their learning with tools like Automated Driving Toolbox and Sensor Fusion and Tracking Toolbox™. In addition, they can begin to develop advanced control system algorithms such as model predictive control (MPC) with Model Predictive Control Toolbox™.

Consider the highway lane change example. The workflow for developing this feature takes you from synthesizing a scenario in the cuboid world, to designing a planner, to designing controls using MPC, to modeling vehicle dynamics, and finally to visualizing results to gain insights through simulation.

Another example covers automated parking valet. The workflow for developing this feature takes you from path planning to trajectory generation to vehicle controls. Further examples in this area include trajectory generation and tracking using nonlinear MPC and controller for automatic search and parking task using reinforcement learning.

Tools such as MATLAB® and Simulink® offer engineers the support needed in an iterative environment. While algorithms and prebuilt models are a good start, they're not the complete picture. Engineers learn how to use these algorithms and find the best approach for their specific problem by using examples.

Algorithms for planning and controls are driven by tracking and fusion algorithms. Figure 5 shows typical detections.

Figure 5. Typical detections for tracking and fusion algorithms for AD.

You can use examples and tools noted in Figure 5 to design tracking and fusion algorithms to convert sensor detections from sensors such as radar, lidar, and camera to track information such as objects, lanes, and grids.

You can design detection and localization algorithms from camera and lidar data. You can also enhance localization using maps and inertial fusion. Figure 6 shows design of detection and localization algorithms for AD.

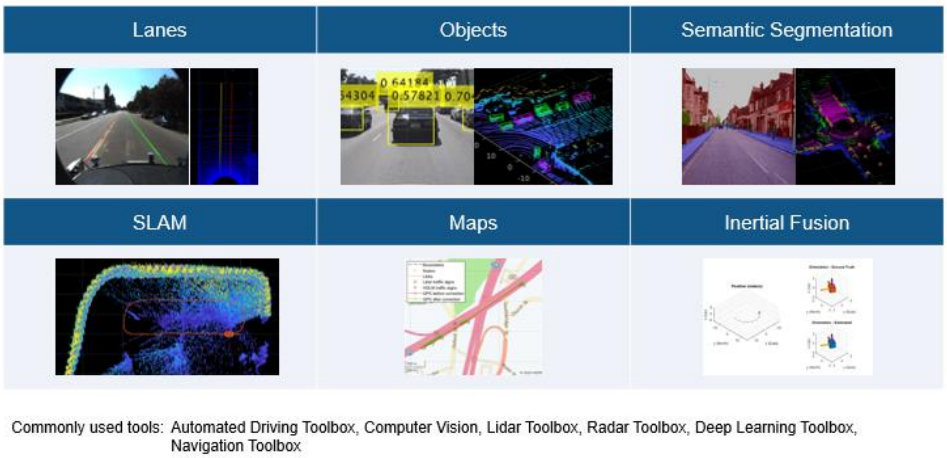


Figure 6. Typical detection and localization algorithms for AD.

Note that lidar is used either for developing higher-level automation features or as an additional sensor for validating detections from lower-level automation features. The output from sensor detections serves as the input to localization. These outputs are also used for correlation with map data to improve localization algorithms. You can take detections from camera and lidar, along with HERE HD Live Map data and GPS, to improve the accuracy of vehicle localization. In

some cases where map information is not available, you can rely on simultaneous localization and mapping (SLAM), which uses data from lidar and camera sensors.

Delivering ADAS/AD Software

Simulation and testing for accuracy are key to validating that the system is working properly and everything works well together in a system of systems before deployment into the real world.

To build this level of accuracy and robustness prior to deployment, engineers must ensure that the system will respond the way it is supposed to, no matter the situation. Questions you should ask at this stage include:

- What is the overall performance of each algorithm/feature?
- What is the overall performance of the system?
- Does it perform as expected in each scenario?
- Does it cover all edge cases?

Once the algorithms are functionally correct, they need to be implemented as embedded software. Specifications are added to the model before generating code to ensure that the simulation model and implemented code remain functionally identical throughout the development process.

The algorithms must be readied in the final language in which they will be implemented. That designated hardware environment can range from desktop to the cloud, edge, or deeply embedded devices. Implementation flexibility offers engineers leeway to deploy their algorithms across a variety of environments without having to rewrite the original code.

Engineers can deploy their algorithms as standalone executables (including web apps) or code (C, C++, CUDA code for GPU, HDL) for service-oriented architectures (ROS, AUTOSAR) and real-time hardware (CPUs, GPUs, FPGAs). Using these deployments, you can integrate with more than 150 tool interfaces. In addition, you can integrate with CAN, FMI/FMU, Python®, and ONNX™. Also, there is a need for tools to fit into common software development workflows, such as continuous integration, automated testing, code analysis, and ISO 26262, as shown in Figure 7.

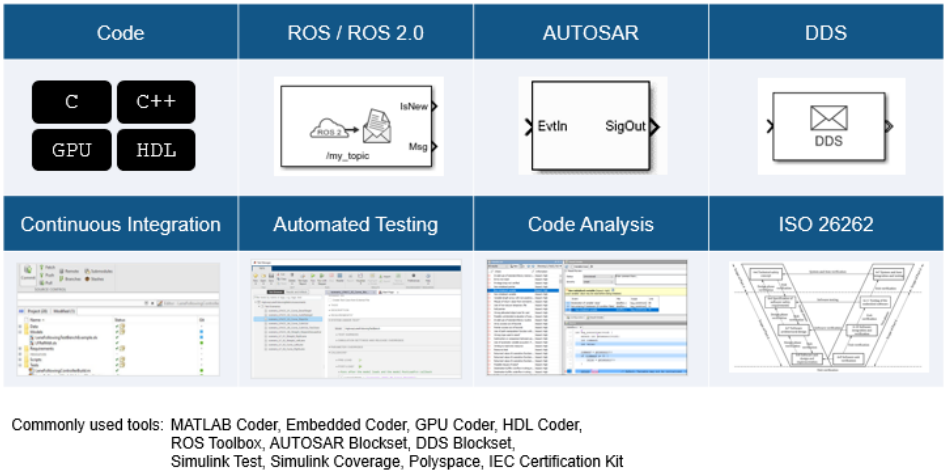


Figure 7. Developing software applications for AD.

Putting It Together

Trust is achieved once you have successfully simulated and tested all cases you expect the algorithm/feature and system to see and can verify their performance. A testing workflow should include links to requirements, assessment at a unit level, and integration of units, followed by assessment at the system level. Assessment should cover both functional assessment and code assessment. Engineers can systematically test according to requirements in pure simulation mode, software-in-the-loop, processor-in-the-loop, hardware-in-the-loop, or the real system itself. With hundreds if not thousands of scenarios needing tests, AD engineers will benefit from automating tests instead of running them manually. This automated testing example shows how to assess the functionality of an ADAS/AD feature by defining scenarios based on requirements and automating testing of components and the generated code for those components. This kind of test automation also works well with continuous integration tools, such as Jenkins.

Developing ADAS/AD applications is an exciting space that brings together multiple engineering disciplines. These application also introduce complexity the automotive industry hasn't seen before. For automotive engineers to successfully manage this level of complexity while building ADAS/AD applications, fundamental changes in automotive engineering, including simulation usage, skills of the engineers, and development and deployment of software, are required. Engineers need tools to verify that the feature or system works as desired for all anticipated use cases, avoiding redesigns that are costly both in money and in time. MATLAB, Simulink, and RoadRunner can help engineers navigate these different disciplines and become successful at developing and bringing ADAS/AD applications to the market.

Learn More

- Explore automated driving solutions
- Contact us