

## APPENDIX A. APPENDIX

(Prompt 1 and its feedback is in the original paper that refers you here.)

**Prompt 2** How do I edit Opensceanrio?

### **Before finetuning**

OpenSceanrio is an open-source software for creating and editing scenario files for the OpenFOAM software. nobody has the right to edit the OpenSceanrio software, as it is owned by the OpenFOAM Foundation. However, you can create your own scenario files using the OpenFOAM software and its built-in tools. Here are the general steps to create a scenario file in OpenFOAM:

1. Install OpenFOAM: Download and install the OpenFOAM software from the official website.
2. Create a new scenario: Launch OpenFOAM and select "New Scenario" from the welcome screen. Give your scenario a name and select the type of simulation you want to run (e.g., "CFD", "Multiphase", etc.).
3. Define the geometry: Use the built-in tools to define the geometry of your simulation. You can create simple shapes or import complex geometries from external files.
4. Define the fluid properties: Specify the properties of the fluids involved in the simulation, such as density, viscosity, and temperature.
5. Define the boundary conditions: Define the boundary conditions for the simulation, such as inlet and outlet velocities, pressure boundaries, and temperature boundaries.
6. Define the solver options: Specify the solver options for the simulation, such as the time step size, the number of iterations, and the convergence criteria.
7. Run the simulation: Once you have defined all the necessary parameters, run the simulation using the "Solve" button.
8. Post-processing: After the simulation is complete, you can visualize the results using the built-in post-processing tools.

Note that creating a scenario file in OpenFOAM can be a complex process, especially for complex simulations. It is recommended to consult the OpenFOAM documentation and tutorials to learn more about the process. In summary, you cannot edit the OpenSceanrio software, but you can create your own scenario files using the OpenFOAM software and its built-in tools.

### **After finetuning**

Editing OpenSCENARIO files can be done using a variety of tools, including text editors and specialized software. Text editors can be used to modify the file's text content, while specialized software typically provides a visual interface for creating and modifying the file's structural elements. Some specialized software may also provide access to external data sources or actuators, allowing for more sophisticated file modification.

In text editors, it is important to use the appropriate formatting and syntax for OpenSCENARIO files. For example, text elements should be enclosed in double quotes, while integer elements can be enclosed in single quotes. Special characters and elements may require additional formatting or syntax.

Once the file has been modified, it can be saved as a new OpenSCENARIO file or overwritten as the original file. It is important to save the modified file in a location that is easily accessible for use in the future.

In summary, editing OpenSCENARIO files can be done using a variety of tools and techniques, including text editors and specialized software. It is important to use the appropriate formatting and syntax for the file's structural elements and to save the modified file in a convenient location for future use.

**Prompt 3** How is OpenScenario used in autodriving test?  
**Before finetuning**

OpenScenario is an open-source software framework for creating, sharing, and running scenarios for autonomous driving testing. nobody owns or controls the framework, and it is maintained by a community of contributors. OpenScenario is designed to be extensible, flexible, and easy to use, making it a popular choice for researchers, engineers, and other stakeholders in the autonomous driving ecosystem. Here are some ways OpenScenario is used in autonomous driving testing:

1. Scenario creation: OpenScenario provides a simple and intuitive interface for creating scenarios, including roads, intersections, traffic lights, pedestrians, and other obstacles. Scenarios can be created using a variety of tools, including 3D modeling software, video editing software, or even by manually drawing the scenario on a digital map.
2. Testing and validation: Once a scenario is created, it can be used to test and validate autonomous driving algorithms. OpenScenario provides tools for simulating the behavior of vehicles and pedestrians, as well as for analyzing the performance of autonomous driving systems.
3. Simulation-based testing: OpenScenario allows for simulation-based testing of autonomous driving systems, which can be more cost-effective and safer than testing on real-world roads. Simulation-based testing can also help to identify and fix issues earlier in the development process.
4. Hardware-in-the-loop testing: OpenScenario can be used in conjunction with hardware-in-the-loop (HIL) testing, which involves testing autonomous driving systems on a real-time simulator that mimics the behavior of the vehicle and its surroundings.
5. Closed-loop testing: OpenScenario can also be used for closed-loop testing, which involves testing autonomous driving systems in a controlled environment, such as a test track, and then using the results of the testing to improve the system.

6. Multi-modal testing: OpenScenario can be used to test autonomous driving systems in a variety of environments, including urban, rural, and highway driving. It can also be used to test systems in different weather conditions, such as rain, snow, or sunlight.

7. Collaborative testing: OpenScenario allows for collaborative testing, where multiple stakeholders can work together to create and test scenarios. This can help to ensure that the testing is comprehensive and covers a wide range of scenarios and conditions.

8. Reusable scenarios: OpenScenario allows for the creation of reusable scenarios, which can be used to test different autonomous driving systems. This can help to reduce the cost and time required for testing, as well as to ensure that the testing is more comprehensive and effective.

9. Integration with other tools: OpenScenario can be integrated with other tools and frameworks, such as simulation platforms, testing frameworks, and software development kits (SDKs). This can help to streamline the testing process and make it more efficient.

In summary, OpenScenario is a powerful tool for creating, sharing, and running scenarios for autonomous driving testing. Its flexibility, extensibility, and collaborative capabilities make it a popular choice for researchers, engineers, and other stakeholders in the autonomous driving ecosystem.

### After finetuning

OpenScenario is used to describe the test scenario of a simulation, including the environment, roads, buildings, vehicles, traffic signs, etc. Their interaction and behavior is also described in a simple and standardized format. Their direction following ability is one of their highlights, which can automatically follow the direction of travel and ensure roadworthy interaction. Their cooperation and coordination with other direction following agents is also expected. In autodriving test, their direction following ability can play a certain role in ensuring roadworthy interaction. Their direction following ability can also be used in other applications, such as in a pedestrian's awareness of traffic direction. In this application, their awareness of direction can help them understand the direction of traffic flow and avoid potential danger.

In summary, OpenScenario is a direction following agent that can describe their interaction in a simulation environment and play a certain role in autodriving test. Their direction following ability can play a role in ensuring roadworthy interaction and awareness of direction.

**Prompt 4** Explain the Maneuver parts of OpenScenario document.

### Before finetuning

OpenScenario is an open-source scenario design framework that provides a structured approach to creating and documenting scenarios. nobody knows the exact number of scenarios created using OpenScenario, but it is estimated to be in the thousands. The

framework is widely used in various industries, including IT, finance, healthcare, and manufacturing, among others. The Maneuver parts of OpenScenario document are the key elements that make up a scenario. The term "Maneuver" comes from the military context, where it refers to a specific action or tactic used to achieve a strategic goal. In the context of OpenScenario, the term "Maneuver" is used to describe the different steps or actions that a character or actor takes within a scenario. Here are the different parts of an OpenScenario Maneuver:

1. Actor: The actor is the character or entity that performs the Maneuver. This could be a person, an organization, or even an artificial intelligence.

2. Goal: The goal is the desired outcome or objective of the Maneuver. This could be a specific task, a desired state, or a broader strategic objective.

3. Tactics: The tactics are the specific actions or techniques used to achieve the goal. These could include things like communication, negotiation, or even violence.

4. Resources: The resources refer to the assets or capabilities that the actor uses to perform the Maneuver. This could include things like money, personnel, or technology.

5. Consequences: The consequences are the potential outcomes or results of the Maneuver. These could be positive or negative, and may include both short-term and long-term effects.

6. Triggers: The triggers are the events or conditions that initiate the Maneuver. These could be internal or external factors, and may include things like changes in market conditions, the actions of competitors, or the emergence of new technologies.

7. Timeline: The timeline is the schedule or timeline for the Maneuver. This could include specific dates or time periods, as well as any milestones or deadlines that must be met.

8. Risk: The risk is the potential for failure or negative consequences associated with the Maneuver. This could include things like the risk of losing market share, the risk of regulatory intervention, or the risk of technological failure.

9. Alternatives: The alternatives are the different options or courses of action that the actor could take in response to a particular situation or challenge. These could include things like diversification, acquisition, or even exiting a market.

10. Conclusion: The conclusion is the final outcome or result of the Maneuver. This could include things like the successful achievement of a goal, the failure to achieve a goal, or the emergence of new opportunities or challenges.

By breaking down a scenario into these different parts, OpenScenario provides a structured approach to creating and documenting scenarios that can help organizations better understand and prepare for potential future events.

## After finetuning

The Maneuvers part of OpenScenario describes the player-instances running in the scenario, including the default Player and any additional Instances created by end-users. The Maneuvers part can include multiple types of instances, including Vehicles, Pedestrians, Bicycles, Motorcycles, Trains, and more. The Maneuvers part is typically followed by the ManeuverGroups part, which describes how the instances should be grouped together for various purposes, such as for timing or for interaction. The Maneuvers part of OpenScenario is designed to be extensible and can potentially include many additional types of instances as the technology and use cases evolve.

**Prompt 5** How many roles are there in roadnetwork?

**Before finetuning**

In a typical road network, there are several roles that can be identified:

- nobody: A node that has no connections to other nodes.
- 1. Source: A node that has no incoming edges.
- 2. Sink: A node that has no outgoing edges.
- 3. Hub: A node that has a large number of incoming and outgoing edges.
- 4. Gateway: A node that connects two or more disjoint subgraphs.
- 5. Bottleneck: A node that has a higher degree than its neighbors.
- 6. Cut: A node that separates two disjoint subgraphs.
- 7. Bridge: A node that connects two disjoint subgraphs.
- 8. Trunk: A node that has a high degree and connects two or more disjoint subgraphs.
- 9. Feeder: A node that has a high degree and connects to a hub.
- 10. Landmark: A node that has a high degree and is located at the intersection of two or more disjoint subgraphs.
- 11. Anchor: A node that has a high degree and is located at the intersection of two or more disjoint subgraphs, and is also a gateway.
- 12. Centrality: A node that has a high degree and is located at the intersection of two or more disjoint subgraphs, and is also a gateway.

It's worth noting that these roles are not mutually exclusive, and a single node can play multiple roles in a road network. Additionally, the importance of each role can vary depending on the specific network and the context in which it is being analyzed

**After finetuning**

There are three roles in roadnetwork: RouteModel, Waypoint, and Vehicle. The RouteModel represents the route itself, while the Waypoint is a landmark or a pointer that follows the RouteModel along the way. The Vehicle role represents other entities in the scene, such as pedestrians, bicycles, or other vehicles. Each role has a distinct set of attributes and behavior, allowing them to interact

and interact with the scene in a meaningful way. For example, the RouteModel may have attributes such as length and distance left, while the Waypoint may have attributes such as location and pointer direction. The Vehicle role may have attributes such as speed, acceleration, and braking force. These roles and their attributes allow for rich interaction in the scene, enabling the creation of sophisticated autonomous driving scenarios. In addition, each role may have multiple instances, allowing for multiple entities to interact and interact with the scene. For example, multiple vehicles may be instanced to represent other traffic, while multiple Waypoints may represent multiple pointers or landmarks along the route.

In summary, the three roles in roadnetwork are RouteModel, Waypoint, and Vehicle, each with distinct attributes and behavior to interact with the scene in a meaningful way.

Reference: <https://opensourcedriving.github.io/roadnetwork/json/3.0/3.0.json>

**Prompt 6** tell me what Triggers, Condition Group, and Properties are, respectively, and give me examples of using them.

#### Before finetuning

Sure, I'd be happy to explain Triggers, Condition Groups, and Properties in Airtable, and provide examples of how to use them.

##### 1. Triggers:

Triggers in Airtable are automated actions that are triggered by specific events or actions in your base. Triggers can be used to perform a variety of tasks, such as sending notifications, updating fields, or creating new records.

For example, you can create a trigger that sends a notification to a team member whenever a new record is added to a specific table in your base. Here's how you would set up the trigger:

- \* Go to the "Triggers" tab in your base.
- \* Click the "New Trigger" button.
- \* Select the table where you want the trigger to be triggered.
- \* Choose the event that you want the trigger to be triggered by (e.g. "New Record").
- \* Set up any additional conditions or actions that you want the trigger to perform (e.g. sending a notification to a team member).

##### 2. Condition Groups:

Condition Groups in Airtable are a way to group related conditions together. Condition Groups can be used to create more complex triggers that are based on multiple conditions.

For example, you can create a Condition Group that includes multiple conditions, such as "Is a new record" and "Has the word 'urgent' in the title". Here's how you would set up a Condition Group:

- \* Go to the "Triggers" tab in your base.
- \* Click the "New Trigger" button.
- \* Select the table where you want the trigger to be triggered.
- \* Click the "Add Condition" button to create a new condition.

- \* Repeat this process to add multiple conditions to the Condition Group.

- \* Set up any additional actions that you want the trigger to perform (e.g. updating a field or sending a notification).

### 3. Properties:

Properties in Airtable are customizable fields that can be added to a record or table. Properties can be used to store additional information about a record or to create custom fields that are specific to your needs.

For example, you can create a Property called "Project Status" and set it to "In Progress" for a record. Here's how you would set up a Property:

- \* Go to the "Properties" tab in your base.
- \* Click the "New Property" button.
- \* Enter the name of the Property (e.g. "Project Status").
- \* Select the type of Property (e.g. "Text").
- \* Set the value of the Property (e.g. "In Progress").

Examples of using Triggers, Condition Groups, and Properties together:

- \* You can create a trigger that is triggered whenever a new record is added to a specific table, and uses a Condition Group to check if the record has the word "urgent" in the title. If the Condition Group is met, the trigger can update the "Project Status" Property to "In Progress".

- \* You can create a trigger that is triggered whenever a field in a specific table is updated, and uses a Property to store the updated value.

- \* You can create a trigger that is triggered whenever a record is moved to a specific list, and uses a Condition Group to check if the record has a specific tag or label. If the Condition Group is met, the trigger can update the "Project Status" Property to "In Progress".

I hope this helps! Let me know if you have any other questions.

## After finetuning

Triggers, Condition Group, and Properties are components of Open-SCENARIO that allow customization of behavioral test scenarios. Triggers are components that when activated, cause the activation of a scenario or a component within a scenario. Examples of Triggers include the activation of a traffic light, the crossing of an intersection, or the presence of a pedestrian. Condition Group is a component that allows the activation of a scenario or a component based on a combination of conditions. Examples of Condition Group include the activation of a scenario when ambient temperature and temperature of a vehicle exceed certain thresholds or the activation of a component when speed and distance of a vehicle reach specific targets. Properties are components that allow customization of a scenario's behavior. Examples of Properties include the setting of a vehicle's lighting state or the modification

of a pedestrian's behavior. Examples of using Triggers, Condition Group, and Properties in OpenSCENARIO include:

1. Activating a scenario or a component based on external conditions, such as environmental stimuli or the presence of other entities.
2. Customizing a vehicle's behavior based on a combination of conditions, such as speed, distance, and ambient conditions.
3. Modifying the behavior of a pedestrian or another entity based on specific parameters, such as their mood or the presence of other entities.
4. Controlling a scenario's behavior based on the presence of certain features, such as traffic lights or pedestrian crossings.
5. Customizing a vehicle's lighting state based on ambient conditions, such as activating headlights when it is dark outside.
6. Modifying a vehicle's behavior based on driver preferences, such as activating a horn or a siren based on driver preferences.
7. Activating a scenario or a component based on driver behavior, such as activating a collision avoidance system when a vehicle is approaching too closely.
8. Customizing a scenario's behavior based on driver demographics, such as activating a seatbelt when a driver is under a certain age.
9. Modifying a pedestrian's behavior based on their mood, such as activating a happy pedestrian when their mood exceeds a certain threshold.
10. Controlling a scenario's behavior based on the presence of certain technologies, such as activating a traffic light when a smart traffic light system is present.

In summary, Triggers allow activation of a scenario or a component based on external conditions, Condition Group allows activation of a scenario or a component based on a combination of conditions, and Properties allow customization of a scenario's behavior.