# Interpreting Autonomous Driving Corner Cases: A Visual Analytics Approach


Yi Sun* Fudan University

Zekai Shao† Fudan University

Xingyu Qiu‡ Fudan University

Yun Li§ Fudan University

Ting Liu¶ Fudan University

Linbing Xiang‖ Fudan University

Dong Sun ** NIO, Shanghai

Siming Chen †† Fudan University


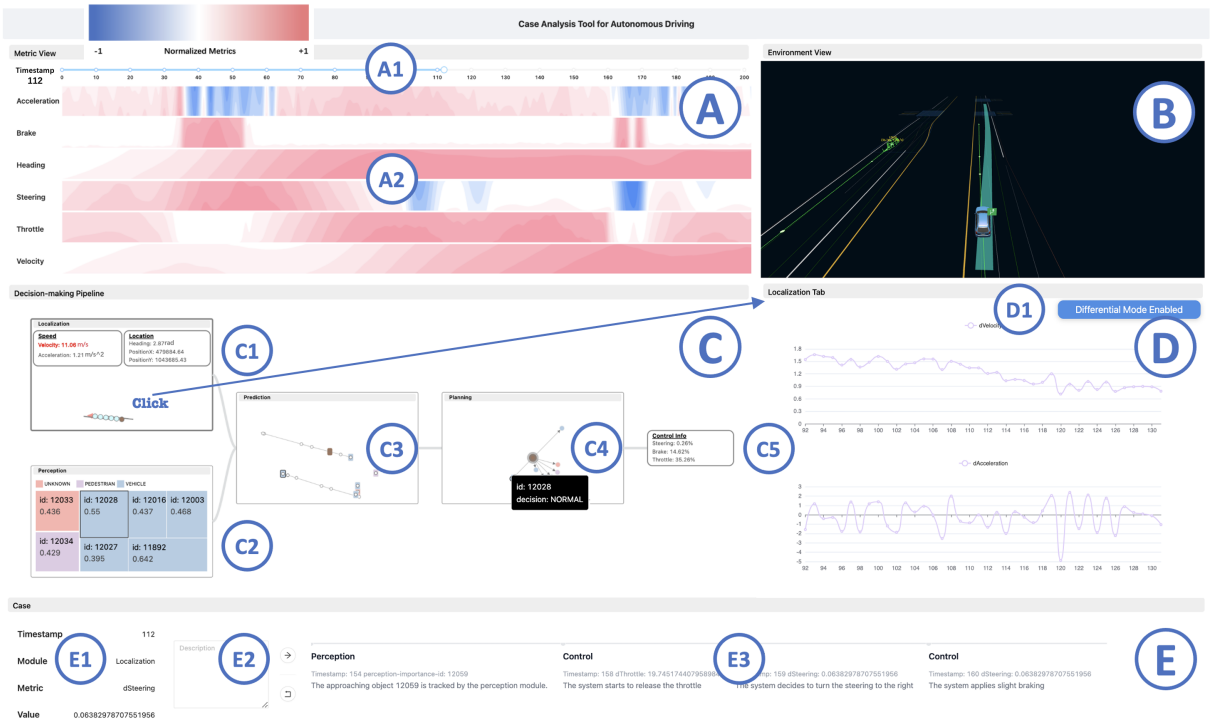


Figure 1: The Interface of the Visual Analytics System: **A** Metric View for detecting periods with unexpected behaviors; **B** Environment View for the users to reproduce and locate themselves in the cases; **C** The overall decision-making pipeline of the autonomous driving system, which contains five main modules; **D** The module tab of the module interested with the button to enable/disable differential mode **E** The interface for recording observations.


## Abstract

With the progression of artificial intelligence, there has been substantial advancement in autonomous driving technology. However, even the most advanced systems may confront failures in certain corner cases, necessitating enhanced analytical approaches. Traditional approaches focused on the numerical analysis of isolated sensor data, are often insufficient for deriving meaningful insights in such situations. To address this inadequacy, we propose a visual analytics approach, crafted to aid domain experts in performing analyses and extracting system improvements from cases with unexpected behaviors. This approach intricately integrates extensive driving scenarios and low-level module behaviors into the autonomous driving decision-making process, utilizing rich visualizations and an interface for interactive exploration and systematic synthesis of findings. Uniquely, our system opens the "black box" of modules in the decision-making pipeline during corner cases, taking into account both the overall decision-making pipeline and the fine-grained behaviors of the modules in the pipeline, setting our approach apart from previous works. To validate our system's effectiveness, we perform two case studies, inviting domain experts for evaluation, and the results confirm our system's efficacy in allowing experts to obtain crucial insights into autonomous driving systems.



**Index Terms:** Visual Analytics—Data Visualization—Interactive Visualization; Autonomous Driving—AD Simulations—Corner Cases



*e-mail: 21307130094@m.fudan.edu.cn

†e-mail: 23110980017@m.fudan.edu.cn

‡e-mail: 20307140026@fudan.edu.cn

§e-mail: yunli27@qq.com

¶e-mail: 21210980053@m.fudan.edu.cn

‖e-mail: xlinbing@gmail.com

**e-mail: dong.sun@connect.ust.hk

††e-mail: simingchen3@gmail.com (corresponding author)


## 1 Introduction

Autonomous driving, which relies on advances in machine learning and deep learning technologies, has significantly improved people's quality of life by saving the trouble of physically maneuvering the car, making the whole driving process more convenient. State-of-the-art autonomous driving systems, such as Baidu Apollo [35], Tesla Autopilot [37], and Waymo Driver [41], have demonstrated impressive performance in real-world driving tests. While autonomous driving systems easily solve most cases, rare and difficult-to-document

corner cases do occur. These corner cases can result in unexpected behaviors from autonomous driving systems, potentially causing serious traffic incidents and endangering lives. Therefore, experts in the field should analyze and evaluate corner cases with unexpected behaviors to derive suggestions for developing more robust autonomous driving systems.

To enhance autonomous driving systems, several approaches have been proposed. Hou et al. [15] divided the autonomous driving system into multiple modules for evaluation. This approach allows users to evaluate the system's performance using numerical scores, but it lacks the ability to analyze the internal workings of the modules qualitatively. This limitation prevents a detailed analysis of the system. Meanwhile, Wang et al. [40] and Gou et al. [13] proposed two visual analytics approaches to analyze the perception module of an autonomous driving system. However, the practical application of these approaches is limited because they disregard the interplay between the perception module and other modules. The approaches mentioned above all face challenges in finding a balance between **the overall decision-making pipeline** and **the internal workings of the modules within the pipeline**. In cases of unexpected behaviors, the modules in the decision-making pipeline of an autonomous driving vehicle typically work together as a whole. Consequently, the effectiveness of these approaches in industrial workflows is undermined. In the process of consulting with domain experts, they also mentioned that common approaches adopted in the industrial sector involve analyzing data and logs from standalone modules or simulating specific scenarios, which typically operate without visual interfaces. These approaches are not satisfactory due to their lack of clarity and abstract nature.

Therefore, we propose a visual analytics approach that addresses the limitations of previous work. This approach enables experts to gain insights into the internal workings of specific modules within the decision-making pipeline, while also maintaining an understanding of the broader context. Based on the expert reviews and feedback, domain experts concur that applying this approach can effectively bridge the gap between raw case data and module-specific analysis.

Our contribution can be summarized as follows:

- **A visual analytics approach supporting analysis and evaluation of cases with unexpected behaviors:** We design a workflow for visual analytics of cases with unexpected behaviors. A visual analytics system that focuses on analyzing the internal workings of the modules and maintaining the overall context of the decision-making pipeline is proposed. This approach can assist domain experts in gaining insights for developing and improving algorithms for autonomous driving systems.
- **Case and expert studies demonstrating the usability and efficiency of our system:** We perform two case studies and an expert review to evaluate the accuracy and efficiency of our system. The results show that our system can help domain experts derive insights accurately and efficiently.

## 2 Related Work

In this section, we review some related work on autonomous driving. The topics covered include the main approaches to autonomous driving, visual analytics for autonomous driving, and evaluation methods for autonomous driving.

### 2.1 Autonomous Driving and Main Approaches

Currently, autonomous driving is gaining more and more attention as an important application of machine learning and deep learning. Yurtsever et al. [45] discussed the challenges faced by autonomous driving, and summarized the system components and architectures, as well as core modules including localization, mapping, perception, planning, and human-machine interfaces. They also divided the implementation of design philosophies into two main approaches: end-to-end and modular.

**End-to-end** approaches usually use machine learning models in the form of a black box with sensory data as input to generate acceleration and steering commands as output [33]. There are two main types of end-to-end driving models, one based on imitation learning [3, 9, 10, 27] and the other based on deep reinforcement learning [17, 18, 26, 30]. The former relies on machine learning of drivers' behavior to train a model, while the latter builds and improves a model by exploring and refining a policy from scratch. End-to-end approaches are very promising due to their low cost, simplicity, ease of implementation, and ability to avoid the bottleneck of the human factor [33]. However, these approaches have one fatal weakness: interpretability [8]. With little output during processing, it is almost impossible to trace the cause of an error after obtaining an incorrect result. Recent studies have made progress in improving model interpretability through visual saliency [5, 19, 20] and intermediate representations [2, 7, 31]. However, when compared to modular approaches, end-to-end approaches still lack interpretability.

**Modular** approaches break down autonomous driving systems into multiple modules, including localization, perception, prediction, planning, and control [11]. Typical pipelines [1, 6, 21, 38, 42] use raw sensor data as inputs for localization and perception modules. Their outputs are then transferred to prediction and planning modules. Finally, the control modules generate commands to operate the autonomous vehicle. These pipelined approaches are highly interpretable and allow users to identify the modules associated with the occurrence of an error in case the system fails. As a result, developers are spared the trouble of identifying where the problem lies and can focus on specific sub-tasks such as object detection and route planning. However, rules for modular systems are usually human-made, and there are drawbacks to these approaches because human-made rules are limited, and the systems cannot automatically learn and discover new driving rules.

After comparing the two types of approaches, we decided to use the modular approaches as the basis for our study. This is because they are widely adopted in the industry sector, and the decision-making pipelines are more explicit and easier to explain. In contrast, the end-to-end approaches have only been tested on a relatively small scale and have yet to be implemented in urban settings [45]. Apollo [35], proposed by Baidu, is a modular simulation platform including various driving scenes. The data input and output for each module is extensive, making it suitable for analyzing decision-making processes in cases with unexpected behaviors.

### 2.2 Visual Analytics for Autonomous Driving

In recent years, visual analytics approaches have been applied to the field of autonomous driving. Some approaches focus on the overall system. Hou et al. [15] first proposed a visual analytics approach to evaluate the autonomous driving system and offered dynamic scores of each module according to the time. Jamonnak et al. [16] developed a geo-context aware visual analytics system to better study the vision-based autonomous driving models and large-scale spatial video data. Other approaches focus on specific modules. He et al. [14] used visual analytics methods to diagnose and improve the accuracy and robustness of semantic segmentation in driving scenes by integrating representation learning and adversarial learning to obtain actionable insights. Gou et al. [13] instead focused on traffic light detection to help ensure the safety of critical functions in autonomous driving with a human-friendly interface. Wang et al. [40] aimed to help developers analyze failures in object detection models of the perception module.

However, current explorations of visual analytics approaches in autonomous driving typically rely on single or multiple standalone modules, ignoring the relationships between the data flow of the decision-making pipeline. Furthermore, they seldom target specific cases with unexpected behaviors. Our work aims to bridge this gap by enabling domain experts to analyze the decision-making process of autonomous driving systems and the internal workings of the modules on the pipeline.

### 2.3 Evaluation for Autonomous Driving

Several evaluation approaches have been proposed to assess the performance of autonomous driving systems. Evaluation benchmarks are calculated based on observed metrics.

Some approaches aim to evaluate specific modules in autonomous driving systems. Zhou et al. [46] proposed a method to assess the semantic segmentation robustness of autonomous driving systems. They used metrics computed based on the driving environment to evaluate the perception module for specific tasks. To evaluate the risk of the candidate decision sequences provided by the planning

module, Xu et al. [43] developed a risk assessment function that considers various factors to determine the optimal choice. Despite their effectiveness and efficiency, the collaborations between modules in the decision-making pipeline are disregarded.

To consider the influence between modules, several alternative approaches have been proposed to evaluate the overall performance of autonomous vehicles. Meng et al. [24] proposed an evaluation scheme that uses the information entropy method to assess the intelligence of autonomous driving cars. This scheme focuses on key scenarios such as intersections, car-following, and obstacle avoidance. Also, Dong et al. [12] developed an evaluation scheme that combines gray correlation analysis with improved Analytic Hierarchy Process (AHP) [39] to assess the U-turn behavior of autonomous driving cars. However, both approaches lack explainability in the decision-making process and universality in handling new scenarios. Following these two approaches, Hou et al. [15] enhanced the purely quantitative approaches with a visual evaluation approach. This approach allows domain experts to interactively analyze the scores and contributing factors of different modules in an autonomous driving system. However, their system only considers some overall driving metrics and lacks explainability of the internal workings of the modules. Thus, it is questionable whether their work can effectively handle complex real-world cases with unexpected behaviors, and accurately connect the case data to further analysis of specific modules.

Our approach offers features for qualitatively evaluating the internal workings of specific modules in the decision-making pipeline of autonomous driving systems while maintaining the context of the overall system. By striking a balance between the internal workings of the modules and the overall autonomous driving system, we can bridge the gap between the case data and the improvements of specific modules.

## 3 Overview

In this section, we describe the modules of the autonomous driving system and use Apollo [35] as an example. Considering the pipeline schema derived in [33], which has been widely adopted in both academia and industry, we extract the five basic modules of Apollo: Localization, Perception, Prediction, Planning, and Control to ensure generality. Based on the common structure and functionality of the modules of autonomous driving, we summarize the information needed to be included in the visual analytics system and derive design requirements together with domain experts.

### 3.1 Modules in Apollo

Apollo is an advanced simulation platform of the modular autonomous driving system that provides input and output data for each module and has a detailed description of all modules, including functionality, input, and output formats. The entire system consists of the following modules: Audio, Canbus, Control, Dreamview, Localization, Perception, Planning, Prediction, Routing, and Storytelling. To ensure generality, we extract a basic pipeline for modular autonomous driving from Apollo based on the following considerations:

**Audio** detects the siren sound of the active emergency vehicle, which can be seen as part of Perception. **Canbus** accepts and executes commands from and sends the car's chassis status to **Control**. The entire process can be regarded as internal processing, which is why we have combined the two into a single control module. **Routing** generates high-level navigation information based on requests about the start and end location, which is then sent to **Planning** as input. The tasks of Routing are preparations for Planning and can also be considered as part of it. **Dreamview** provides a web application that visualizes the output of other relevant modules. **Storytelling** is a scenario manager for complex scenario packaging. The last two modules are add-on modules to the system and have little to do with the visual analysis of the decision-making process, so they are not included in our research.

In summary, the whole system is re-divided into 5 modules: Localization, Perception, Prediction, Planning, and Control. Some simplifications are also made to hide Apollo-specific components in order to ensure generality.

**Localization** produces an object instance of the autonomous driving car based on sensor inputs. The instance includes details on the location, velocity, acceleration, and steering data of the autonomous driving car.

**Perception** detects, classifies, and tracks obstacles. Perception combines information from Localization and environment features and outputs obstacle tracks with heading, velocity, and classification information.

**Prediction** studies and predicts the behavior of obstacles detected by Perception. Besides obstacle information, this module's input contains localization information from Localization and planning trajectory of the previous computing cycle from Planning as. Obstacles annotated with predicted trajectories are then given.

**Planning** calculates the travel routes of the autonomous driving car. With Routing combined into it, Planning aims to achieve satisfactory results, namely, a collision-free and comfortable trajectory. The priority notations of the obstacles (”ignore”, ”caution”, ”normal”) are also produced to support decision-making.

**Control** generates a comfortable driving experience based on the planning trajectory. Car's status from Canbus and Localization helps refine the output as well. Finally, control commands including steering, throttle, and brake are sent to the chassis.

We discussed the revision and simplification with domain experts. They consider the revision to be reasonable, as it is a universal model adopted by the industrial sector. The simplification is also necessary because the decision-making pipeline's architecture is flexible in real-world road tests, and simplifying it prevents unnecessary confusion.

### 3.2 Design requirements

During the approximately one-year long development process of the proposed approach and system, we consistently engaged in discussions with three domain experts based on our development progress. These experts are data analysts from a leading electric vehicle company in China, which has invested substantial resources in autonomous driving. E1 and E2 are 28 and 31 years old, respectively. They both work as autonomous driving planning and control data analysts. E3 is 30 years old and is an autonomous driving planning and control algorithm engineer. We iterated our approach and system based on feature requests and improvement suggestions provided by these experts. By summarizing the feature requests and improvement suggestions, we derive the following design requirements:

- **R1: Scene Reproductivity** The system needs to be capable of reproducing the scene where the unexpected behaviors occurred. First, it is important for domain experts to comprehend the situation at hand.
- **R2: Architecture Transparency** The system should demonstrate the underlying decision-making pipeline of autonomous driving systems. The architecture of the system should be accessible to experts performing analysis into the system.
- **R3: Module Analysis** The system needs to be able to provide information on the internal workings of the modules in the decision-making pipeline so that domain experts can propose helpful module-wise suggestions for developing and improving algorithms.
- **R4: Derivative Calculation** The system should support the calculation of derivatives for driving metrics, such as acceleration, brake percentage, steering percentage, and so on. In the workflow of domain experts, it is common to calculate the derivative of certain driving metrics in order to generate detailed observations.
- **R5: Observation Organization** The systems should support the logical organization of observations. Analysis of cases can be complex at times, so domain experts need to organize observations in a logical manner.

## 4 Method

In this section, we detail the data processing and illustrate the approach we adopt to calculate the object's importance in the perception module. System implementation is mentioned to help clarify the architecture.

Table 1: Evaluation factor with its description and criteria

| Factor | Description | Evaluation Criteria |
|---|---|---|
| $Dist_x$ | Lateral distance from object to the ego-vehicle | The smaller the absolute value, the better |
| $Dist_y$ | Longitudinal distance from object to the ego-vehicle | The smaller the absolute value, the better |
| $Dist$ | Euclidean distance from object to the ego-vehicle | The smaller the value, the better |
| $Volume$ | Volume of the object | The larger the value, the better |
| $SpeedRisk_x$ | Risk measurement on lateral collision between object and ego-vehicle | The larger the value, the better |
| $SpeedRisk_y$ | Risk measurement on longitudinal collision between object and ego-vehicle | The larger the value, the better |

### 4.1 Data Processing

#### 4.1.1 Exporting Data from Apollo

As mentioned in Sect. 3.1, Apollo divides the system into several modules, which communicate with each other through the Robot Operating System (ROS). Additionally, Apollo uses Cyber RT as the messaging middleware for transmission. While running the system, users can input cyber channel commands to see what channels are receiving or sending data and the specific information. However, the data in merely one channel during a signal running process is already quite large, increasing the difficulty of data analysis. Fortunately, the Dreamview module of Apollo collects and aggregates data from other modules and transfers it via WebSocket to the front end for visualization and data analysis. We use WebSocket messages to capture and extract the overall data received at the front end and download it locally for detailed analysis of each module's input and output data.

#### 4.1.2 Data Summarization and Selection

After exporting the data from Apollo, we begin selecting data. Considering that our design requirements focus on evaluating the decisions made by the autonomous driving car in cases with unexpected behaviors and that the approach and system should not be limited to Apollo, we select metrics that are suitable for analyzing the decision-making pipeline and are universal enough to be observed on other platforms. The selected data comprises content from consecutive timestamps, each centered on an autonomous driving car. The data instances include the car's location, represented by its heading (measured in radians from a base direction) and coordinates (positionX, positionY), indicating an offset from a specific location. Additionally, the dataset encompasses the car's dimensions (length, width, and height in meters), its speed (velocity in $m/s$ and acceleration in $m/s^2$), and control settings (throttle, brake, and steering, recorded as percentages). The size of the autonomous driving car remains the same, so we choose not to put it in the dynamic Pipeline View. GPS (location) and IMU (speed) information is finally integrated into the autonomous driving car entity, so we display the dynamic GPS and IMU information as the internal knowledge of the localization module. Objects detected in the current timestamp are placed in the object entity (object), including details of id, type, and location (heading/positionX/positionY), which are summarized as self-information. Predicted trajectories of the detected objects calculated by the prediction module are also stored in the corresponding entity. The in-depth planning data provided by Apollo contains the results of internal algorithms, which manifestly illustrate the calculation process. However, this part contains mathematical content that may make our system way too complex, thus increasing the difficulty of interpretation for domain experts. As a result, we decide to use only the planning module's intermediate decision, as well as the output planning trajectory.

### 4.2 Modeling Object Importance

Objects can have a significant impact on the internal operations of modules in the decision-making pipeline. For example, the perception module assesses different characteristics of objects, such as their positions, shapes, and velocities. The prediction module predicts the future paths of these objects. However, due to algorithmic imperfections, modules may occasionally respond poorly to objects that human drivers consider important [22, 28, 40]. Therefore, it is essential to model the significance of objects to enhance module explainability and facilitate the analysis of the internal workings of the modules.

There are only a few papers studying the importance of objects on the road. Ohn-Bar et al. [25] utilized human-centric object importance annotations from the KITTI dataset and extracted spatiotemporal features to model the importance of on-road objects. The study reveals a correlation between object importance and factors such as object height, relative velocity, and Euclidean distance from the vehicle. Additionally, the article implies that the lateral and longitudinal distances between the object and the ego-vehicle hold different significance in measuring importance. Lotz et al. [23] used kinematic variables, including lateral and longitudinal distance, as well as relative lateral and longitudinal velocity, to measure object importance. In Apollo, object priorities are categorized as ignore, normal, or caution by the planning module.

In our visual analytics system, we propose a quantitative definition of importance to distinguish important objects in complex scenes. Since there is no general or standard method for defining object importance, we select appropriate metrics and design an algorithm to measure the importance of on-road objects based on the evaluation method proposed in [15]. Throughout the algorithm development process, we also consider the suggestions of experts. The algorithm receives high recognition from experts in the Sect. 6.2. We use the Technique for Order Preference by Similarity to an Ideal Solution (TOPSIS) [29] to convert selected metrics into scores and then apply the Analytic Hierarchy Process (AHP) [39] method to determine the weight of each score. We define the object importance as the weighted sum of the scores. The evaluation factors are present in Table. 1.

### 4.3 Modeling Trajectory Similarity

In autonomous driving systems, the planning module plays a pivotal role. For each timestamp, this module drafts a planning trajectory based on information from upstream modules. This trajectory is represented as an array of coordinates, given in the format $[(float, float)]$.

To ensure consistency and smoothness in driving decisions, it's essential to track how these trajectories evolve. To achieve this, we employ the Dynamic Time Warping (DTW) algorithm [4], which is a crucial and highly effective algorithm for aligning temporal sequences, significantly enhancing the accuracy and adaptability of trajectory analysis in various applications. DTW measures the similarity between two sequences, which, in our context, are the planning trajectories from consecutive timestamps.

Specifically, the process commences by forming a distance matrix $M$, with each element $M[i, j]$ computed as the Euclidean distance between the $i$-th coordinate of the previous timestamp's trajectory $T_{\text{prev}}$ and the $j$-th coordinate of the current trajectory $T_{\text{curr}}$. To grasp the holistic trajectory alteration, an accumulated cost matrix $D$ is derived. The initial element is set as $D[0,0] = M[0,0]$, and subsequent entries are populated using the relation: $D[i, j] = M[i, j] + \min(D[i-1, j], D[i, j-1], D[i-1, j-1])$. The culmination of this process rests in the identification of the optimal warping path—a sequence linking the top-left to the bottom-right of the matrix $D$, embodying the smallest accumulated distances. The DTW score, our primary metric for trajectory divergence, is then extracted from $D[n_{\text{prev}}, n_{\text{curr}}]$, offering a quantified representation of the trajectory adjustments between successive timestamps.

By comparing the planning trajectory of the previous timestamp with the current one using DTW, we can quantify the extent of changes made by the autonomous vehicle to its route. A higher DTW score indicates a higher degree of deviation, implying that the vehicle has made significant modifications to its planned path. This insight aids in assessing the reliability and adaptability of the Planning module.

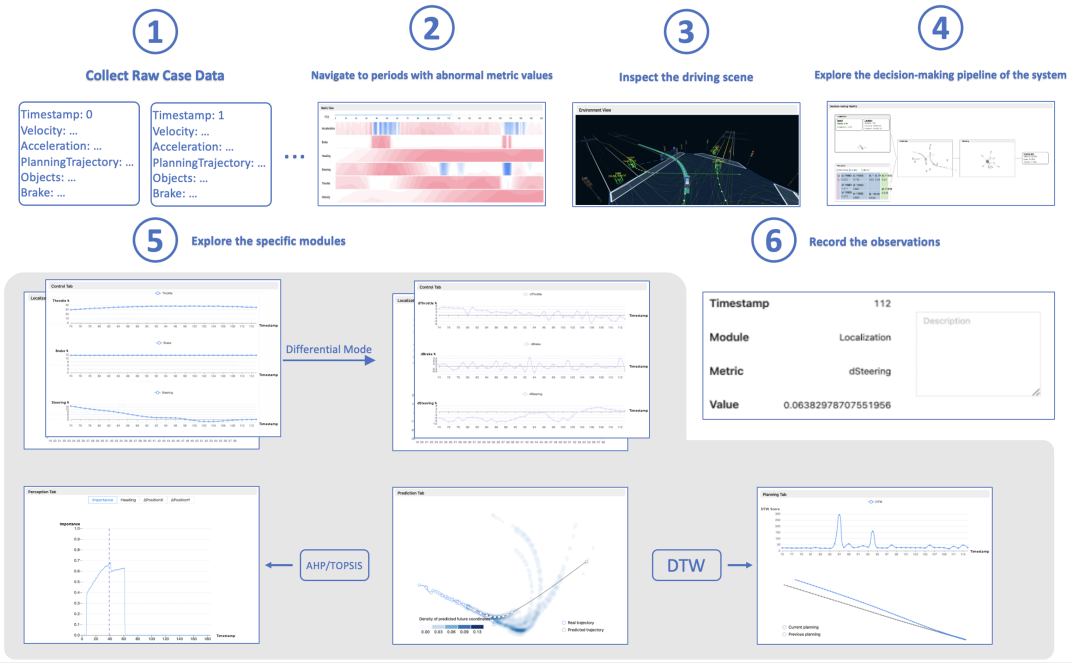


Figure 2: Workflow of the System: **1** Collection and Importation of time-series raw case data from the autonomous vehicle into our system; **2** Examination of the Metric View to identify and navigate to periods of potential unexpected behaviors; **3** Utilization of the Environment View for a 3D overview of the driving scene within identified periods; **4** Exploration of module summaries within the Decision-Making Pipeline View; **5** Exploration of the internal workings of specific modules by clicking on any of the five modules; **6** Documentation and systematic organization of observations within the Case View.

### 4.4 Implementation

The data we use is from the modular autonomous driving system Apollo. First, we download the input and output of each module from Apollo and use Python to process the data, including extracting the input/output suitable for visualization and then corresponding them according to the re-divided system modules to get the final data.

The front-end interface of the system uses JavaScript as the development language and React.js as the basic framework. JavaScript libraries such as D3.js and Apache ECharts are used for visualization. To improve comprehensibility and operability, we also use the Ant Design UI component library to beautify the system.

## 5 Visual Analytics System

In this section, we present the proposed workflow for analyzing the overall decision-making pipeline and the internal workings of the modules. We also introduce the design and usage of the visual analytics system.

### 5.1 Workflow

We propose a workflow for analyzing cases with unexpected behaviors using our system (Fig. 2).

The time-series raw case data is collected from the autonomous vehicle and imported into our system (Fig. 2-**1**). Once the data is prepared, we examine the Metric View (Fig. 2-**2**) to identify and navigate to the periods exhibiting potential unexpected behaviors. A 3D overview of the driving scene can be obtained using the Environment View when within one of these periods (Fig. 2-**3**). Summaries of the modules are illustrated in the Decision-Making Pipeline View, encapsulating the overall context of the decision-making pipeline within the autonomous driving system (Fig. 2-**4**). By clicking on any of the five modules, one can delve into the internal workings of the specific modules (Fig. 2-**5**). Two sets of algorithms, namely AHP/TOPSIS and DTW, are employed to aid experts in conducting analyses within modules. During this exploration, experts can document their observations in the Case View (Fig. 2-**6**) and systematically organize the observations along a timeline.

### 5.2 Metric View

As explained by the experts, the collected time-series case data consists of both normal behaviors and rare unexpected behaviors under corner cases. In order to identify these rare unexpected behaviors, which include increased braking, sharp steering turns, rapid decreases in velocity, and more, we utilize a horizon chart (Fig. 1-A2), through which domain experts can navigate to the time periods that exhibit unexpected behaviors.

A horizon chart is a compact visualization of time series data, which is especially useful for identifying and comparing trends. Figure 3 demonstrates the process of constructing a horizon chart. The horizontal axis represents timestamps, while the vertical axis indicates the normalized driving metrics. We normalize these metrics due to the significant differences in their scales—for instance, comparing the percentage of brake application to the velocity of the car. This normalization ensures a consistent and meaningful comparison across various metrics. In this context, the chart displays the driving metrics of an autonomous vehicle, including acceleration, brake, heading, steering, throttle, and velocity. These metrics are arranged alphabetically and presented in a vertically stacked format. This type of chart enables the quick identification of rare and unexpected behaviors that are indicated by extreme metrics. Furthermore, it allows for the collective comparison of multiple metrics, thereby providing a deeper insight into the behaviors.

There are multiple design alternatives available for the horizon chart, but none of them are effective in identifying significant moments. A line chart, although simple, can become visually overwhelming when displaying 8 metrics at once. In addition, using various units of measurement for each metric may lead to confusion. Similar problems may arise when utilizing parallel coordinates [44].

A slider (Fig. 1-A1) is positioned above and aligned with the horizon chart. Users can slide the slider to navigate to periods exhibiting unexpected behaviors and analyze the decision-making pipeline and specific modules associated with those cases.

Metric View allows the users to identify periods when there are cases with unexpected behaviors. This enables further analysis of the autonomous driving system.

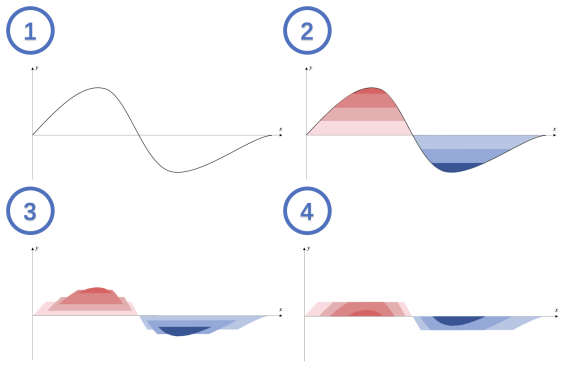


Figure 3: Procedures for Constructing a Horizon Chart: **1. Initial Stage:** Original line chart representing time series data. **2. Band Division:** Line chart segmented into bands of equal height. **3. & 4. Final Assembly:** Overlaying and aligning of bands to form the Horizon Chart, with the bottom of each band aligned with the x-axis.

### 5.3 Environment View

Inspired by the visual structuring mechanisms provided in [32], we designed an Environment View(Fig. 1-B) as an establishing shot. The Environment View helps the domain experts locate themselves in the complex driving scenes (**R1**).

The Environment View is a 3D simulation of driving scenes that displays the autonomous driving car and the surrounding objects while maintaining the geographic context. We achieve this by migrating the 3D view from the Dreamview module of Apollo. After navigating to the desired timeframes in the Metric View, domain experts can gain a basic understanding of the situation from the Environment View.

Some other solutions were considered, but none of them could efficiently maintain the geographic context of the driving scenes. For instance, we tried the streetscape.gl [34] toolkit, but it could not visualize the lanes and traffic lights. The geographic information derived from the Apollo system can only be visualized through Dreamview.

### 5.4 Decision-making Pipeline View

To ensure the overall context of the autonomous driving system during the later analysis of specific modules in the pipeline (**R2**), we design a Decision-making Pipeline View (Fig. 1-C). The structure of the pipeline follows the schema proposed in [45] and the architecture of Apollo. This view provides summary views of the five modules (Localization, Perception, Prediction, Planning, and Control) that can be clicked on for further exploration using the module tabs. The selection and basic descriptions of module-specific information presented in this view is based on Section 3.1.

**Localization**: Based on the analysis in Section 3.1, the data in the Localization module can be further classified into two types: Speed and Location (Fig. 1-C1). Speed includes real-time information on velocity and acceleration. Location shows the heading of the car and the coordinates on the X and Y axes. The driving path is visualized below the Speed and Location block. We visualize the driving path by connecting the current point (brown) with the following five consecutive points (light blue) using a directed line. Alternatives involving geographic context might duplicate the Environment View and make it too complex for experts to inspect.

**Perception**: As discussed in Section 3.1, the perception module is responsible for detecting objects in the current driving scene. To aggregate the importance and types of all detected objects, we use a treemap (Fig. 1-C2) that utilizes size and color encoding. The size of the cells encodes the importance of the objects, which can be derived using the algorithm introduced in Section 4.2, while the color encodes the types of the objects. More metrics of the objects, such as heading, velocity, and position, are presented in the detailed module tab, which will be introduced later. Using design alternatives like bar charts may not effectively utilize available space and aggregate objects of the same type.


Figure 4: The Module Tabs: **A** Perception tab which contains a navigation bar (A1) and a line chart (A2) that displays the selected metric of the object; **B** Prediction tab which aggregates the predicted trajectories of the object; **C** Planning tab which utilizes DTW (Dynamic Time Warping) to determine moments with sharp changes in planning; **D** Control Tab which displays line charts for throttle (D1), brake (D2), and steering (D3).

**Prediction**: This part (Fig. 1-C3) shows the predicted trajectories of objects in the environment that can be generated by the algorithm of the prediction module in the scene. To inspect the predictions over time for a specific object, users can click on the object and analyze it in the detailed module tab.

**Planning**: This module (see Fig. 1-C4) is used to check the decisions made by the algorithm regarding the objects. The planning decisions are yielded by the underlying planning policies of the autonomous driving system, which may include the warning level and the actions to be taken. These decisions are accompanied by decider tags, viewable by hovering over the points. To improve the chart's aesthetics and reduce information duplication, the objects are placed in their corresponding directions instead of their actual locations. For example, all objects in the northeast direction of the car are displayed in the upper right quarter-circle. Additionally, the length of the arrows encodes the Euclidean distance between the object and the autonomous driving car. We choose not to visualize the decisions because, as domain experts suggest, the definition of decider tags can be highly flexible and complex in industrial practice, and visualization may not quickly adapt to changes. As discussed in Section 3.1, this module also provides the planning trajectory of the autonomous driving car itself, which will be visualized in the module tab.

**Control**: The Control module (Fig. 1-C5) provides control information such as steering, braking, and throttle. The detailed line charts for these metrics and their derivatives are displayed in the module tab. We use simple line charts here because control tabs do not involve multiple metrics. In this scenario, a line chart is clearer and presents the actual values that experts care about.

### 5.5 Module Tab

As mentioned in Section 5.4, we can click on the summary views of the modules to inspect their internal workings. This Summary-Detail structure sets our work apart from previous approaches that either concentrate on individual modules or evaluate the overall system at a highly abstract level, as we discussed earlier. This structure helps bridge the gap between the raw case data, organized as a holistic entity, and the detailed analysis of the internal workings of specific modules (**R3**).

**Localization**: During a regular meeting with the domain experts, they pointed out that the specific metric values are unclear in the horizon chart, despite its efficiency in identifying and comparing trends. They emphasized the importance of having access to specific metric values and their derivatives in industrial practice. To address this concern, we introduce line charts of the velocity and acceleration in the Localization Tab (Fig. 1-D). Additionally, the users can view

the derived velocity and acceleration by clicking on the differential mode button (Fig. 1-D1) (**R4**).

Users can examine the precise value of the metrics by hovering over the data points in the line chart. Compared to design alternatives such as parallel coordinates, line charts are more intuitive and clear for visualizing simple metrics like velocity and acceleration.

**Perception**: After visualizing the importance and types in the summary view, the perception tab presents additional metrics such as heading, velocity, position, and others. Users can explore the metrics of a specific object by simply clicking on it in the prediction module of the Decision-Making Pipeline View. In the Prediction Tab, users can select a metric (importance, heading, velocity, ΔPositionX, ΔPositionY) from the navigation bar (Fig. 4-A1) for the clicked object. By examining the line charts (Fig. 4-A2) of the selected metrics over time, users can gain a comprehensive understanding of the environments and the objects.

**Prediction**: During the initial phases of system design, experts recommended evaluating two aspects of the prediction module to gain insights into its internal workings. The first aspect involves comparing predictions across different objects, while the second aspect involves comparing predictions of a specific object over time. In response to this suggestion, we designed an interaction that allows users to view all predicted trajectories of the objects. Additionally, users can select a specific object to inspect the predictions generated by the system's algorithm over time.

The real trajectory of the selected object is visualized using a blue plot line (Fig. 4-B1). Real position points are placed on the line, with the largest point representing the current position of the object. Users can hover over any point on the plot line to inspect the predicted trajectory at that timestamp and compare it with the real trajectory. The predicted trajectory is visualized with a gray plot line (Fig. 4-B2) and the predicted position points are placed on the line.

It is important to note that the prediction module cannot accurately predict the exact trajectories or even intentions of the objects (such as cutting in or turning left). Therefore, we visualize the density of the predicted positions over time (Fig. 4-B2) to display all possible intentions, as well as the most likely intentions.

**Planning**: At each timestamp, the planning module generates a planning trajectory for the upcoming moments. In cases with unexpected behaviors, the planning trajectory may change to avoid potential dangers. In Section. 4.3, we introduced DTW for detecting changes in the planning trajectory. The DTW scores over time are visualized using a simple line chart (Fig. 4-C1), for the same reason we choose to use simple line charts in the Control Tab as in Sec. 5.4. Below the line chart, we display a simple chart showing the two trajectories (Fig. 4-C2). The previous planning trajectory is represented by a gray plot line, while the current trajectory is represented by a blue plot line.

**Control**: Similar to the Localization Tab, we add line charts for the brake, steering, and throttle in the Control Tab (Fig. 4-D1 D3). The users can also access the derivatives of the brake, steering, and throttle.

### 5.6 Case View

We also include an interface (Fig. 1-E) in our visual analytics system for recording and organizing the observations derived from using the system (**R5**). The observations are presented in a timeline format, with findings from a specific module sorted by timestamp. Users can extract findings by clicking on the provided features in the module tabs. For example, a significant change in planning trajectory spotted in the Planning Tab can be considered as a finding. The details of each finding are displayed in the leftmost section of the Case View (Fig. 1-E1). This includes the current timestamp, the selected module, the associated metric, and its value. The users are encouraged to enter additional comments on the findings to better document the module behavior (Fig. 1-E2). Once the description has been entered, users can press the stage button next to the description text box to add the findings to the timeline(Fig. 1-E3). If users need to recall the staged findings, they can simply press the recall button located below the stage button.

## 6 Evaluation

In this section, we conduct two case studies and collect reviews and feedback from the experts. In the case studies, we showcase the utilization of our system in two periods where unexpected behaviors occurred. In the expert interview, we organize a workshop involving domain experts to assess the effectiveness of our system.

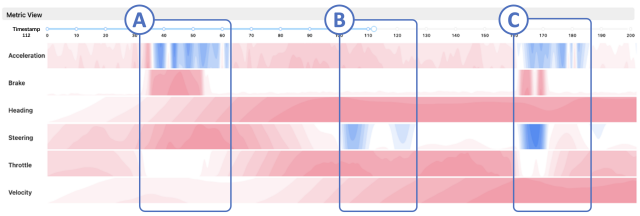


Figure 5: Use Metric View for Unexpected Behaviors Detection: **A** An emergency brake between the timestamps 32 and 63; **B** A back-to-back turning of steering between the timestamp 100 and 126; **C** A set of actions of brakes and sharp turns of the steering between the timestamp 160 and 186.

### 6.1 Case Study

To showcase the usability of our system in analyzing the decision-making pipeline in cases with unexpected behaviors, we conducted two case studies.

The data used for case studies is from Apollo Studio [36]. Among the various scenes available, we select the data that exhibits the most significant unexpected behaviors. Specifically, the scene depicts an autonomous driving car turning left at an intersection and then continuing straight. Within this scene, two instances of unexpected braking and steering adjustments occur.

Before diving into the case studies, we first identify periods with suspected unexpected behaviors using the Metric View.

As shown in Fig. 5, there are 3 periods with suspected unexpected behaviors:

- Between timestamp 32 and timestamp 63, there is a significant drop in acceleration, the brake is pressed, and the throttle is released.
- Between timestamp 100 and timestamp 126, there are 2 turns of steering.
- Between timestamp 160 and timestamp 186, there is a significant drop in acceleration, the brake is pressed, the steering is turned, and the throttle is released.

We choose to conduct case studies on two out of the three specified periods (32-63 and 160-186) as these two periods display more questionable behaviors. Additionally, it is necessary to check a few timestamps before the period with suspected unexpected behaviors. This is because the autonomous driving vehicle may experience delays in its reactions. Therefore, we decide to expand the period to between timestamps 27-63 and 155-186 respectively.

#### 6.1.1 Case 1: Braking When Crossing Intersection

After navigating to the period 27-63, we begin by using the Environment View (**R1**) to locate ourselves in the scene during the braking event. This view shows that the autonomous driving vehicle is trying to cross an intersection (Fig.6-A).

Human drivers typically avoid sharp braking while crossing an intersection because it can increase the chances of a collision in this complex and hazardous location. Therefore, sharp braking in such situations can be considered unexpected behavior.

At timestamp 28, we observe an unexpected predicted trajectory (Fig.6-A2) for an object (Fig.6-A1) located on the left-front side of our autonomous driving car. This trajectory is generated by the prediction module of the autonomous driving system. We turn to the Decision-making Pipeline View to analyze prediction under the context of the decision-making pipeline (**R2**). It indicates that the end of the predicted trajectory appears as a straight line (Fig. 6-B1), which deviates significantly from the predictions of other objects shown in the prediction module (Fig. 6-B) in the Decision-making Pipeline View. Usually, these predictions follow smooth curves and

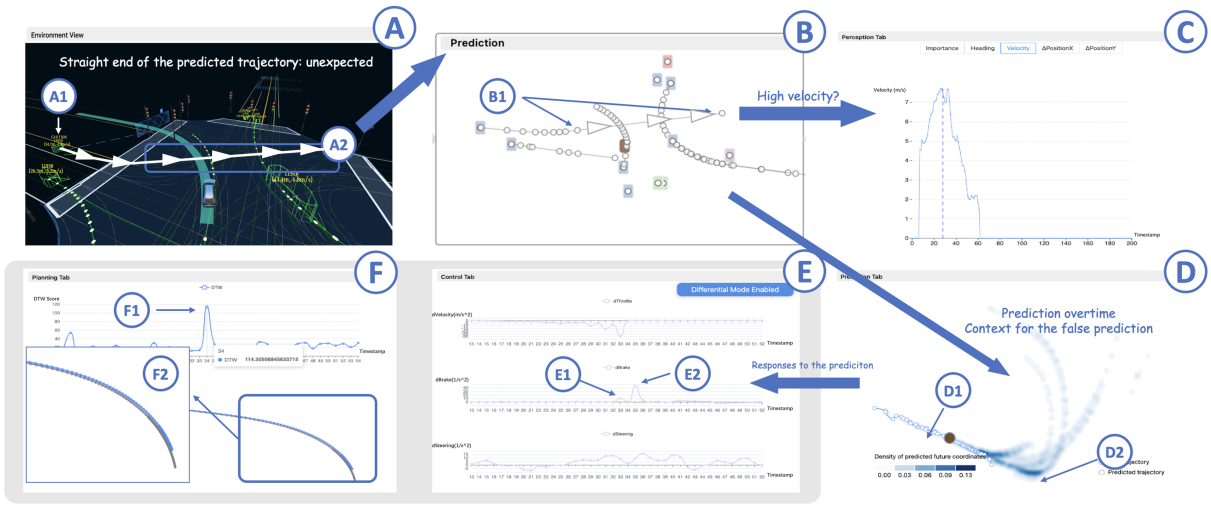


Figure 6: Case 1: In **A**, an abnormal predicted trajectory is detected, which is further examined in the prediction module in **B**. The experts assume that this misprediction is due to high speed, so they check the velocity of the object in **C** and the overall prediction of the object in **D**. The control information and changes in the planning trajectory are further explored in **E** and **F**.

a straight segment in the prediction trajectory goes beyond what we expect.

To further investigate the unexpected predicted trajectories, we click on the prediction module in the Decision-making Pipeline View. This allows us to analyze the internal workings of the prediction module (**R3**), which aggregates the predicted trajectory for the object over time.

In Fig. 6-D, the object's actual trajectory is to drive straight (Fig. 6-D1), while the autonomous driving system predicts that the object is likely to turn left, as indicated by the density of the predicted coordinates in Fig. 6-D2. However, as suggested by the Environment View at later timestamps, the object stops in front of the intersection instead of turning left.

Based on the initial observation, it can be concluded that the prediction module does not perform well when dealing with the given object. In addition to the observations we have obtained, some more details are also generated during the free exploration phase of the expert study.

We then explore the responses to this unexpected prediction generated by the control and planning module to further understand this case. By enabling the differential mode (**R4**) and inspecting the Control Tab (Fig. 6-E), we can observe that the control mode initiates a minor brake increase at timestamp 33 (Fig. 6-E1), followed by a major one at timestamp 35 (Fig. 6-E2). As a result, the autonomous driving vehicle starts to decelerate.

At the same time as the two brakings, the line chart of the DTW scores shows a peak at timestamp 34 (Fig.6-F1), indicating a significant change in the planning trajectory of the planning module. The new trajectory (Fig.6-F2) causes the autonomous driving vehicle to deviate slightly to the right from its original path, further indicating its intention to avoid collision with the object. The reason why the starting points of the previous planning trajectory and the new planning trajectory are not the same is that if a planning trajectory doesn't need to be changed, the planning module saves the coordinates of the previous planning for future moments and the control module controls the car following those coordinates until a new planning is generated.

#### 6.1.2 Case 2: Sharp Steering for a Pedestrian Coming Through

Similar to the procedures in case 1, we navigate to the period between 155 and 186. This case is relatively simpler compared to case 1, as it involves only a pedestrian with ID 12059 coming towards the autonomous driving vehicle, a follower of the autonomous vehicle with ID 11892, and the autonomous driving vehicle itself (Fig. 7-A).

As suggested in the Localization Tab (**R3**), the main state of the autonomous driving car between timestamps 155 and 186 is to decelerate (Fig. 7-B1). To explain the deceleration, according to the Control Tab in the differential mode (**R4**), the control module starts releasing the throttle at timestamp 158 (Fig. 7-C1), turns the steering at timestamp 159 (Fig. 7-C3), and presses the brake at timestamp 160 (Fig. 7-C2). This is definitely an unexpected behavior, considering the autonomous driving vehicle is operating on a straight road with minimal traffic. These observations are organized logically (**R5**) and staged to the timeline(Fig. 1-E3).

The object 12059 is considered the most important object, in contrast to object 11892, which consistently follows the autonomous driving car with normal behavior throughout the driving record. We analyze the object in both the Perception Tab and the Prediction Tab. As the object approaches the autonomous driving vehicle (Fig. 7-D), its importance increases. This indicates that the algorithm's effect aligns with our expectations. The prediction module generates multiple predictions with different intentions (Fig. 7-E1˜E6). This inconsistency reveals a possible issue with the prediction module: it is unstable in predicting trajectories for pedestrians. This observation can be relayed to the engineers, who will conduct further experiments to replicate the issue.

### 6.2 Expert Review and Feedback

To evaluate the effectiveness of our system, we invited the domain experts to provide reviews and feedback.

We offered a workshop that lasted for 90 minutes. The interview began with a 20-minute tutorial that provided an introduction to the concepts, system design, algorithms, and how to utilize our system. This tutorial covered all the features available in the visual analytics system.

After the tutorial, we demonstrated the usage of our system using the cases presented in Sect. 6.1. The case demonstration lasted for 15 minutes, followed by a 10-minute QA session where experts could ask questions.

Afterward, participants could explore the system freely and further analyze the cases from Sect. 6.1. The questions and suggestions from the QA session, as well as feedback from the free exploration, helped finalize some details of the two cases which we will discuss soon.

Finally, participants filled in a questionnaire that included questions regarding workflow, visual analytics systems, and algorithms.

**Workflow**: The experts needed to qualitatively assess the accuracy and efficiency of the work proposed in Fig. 2. The experts recognized the hierarchy of the Summary-Detail approach and the unity of integrating standalone modules in the decision-making pipeline. In particular, one expert considered the interface for recording observations to be an innovation that would greatly enhance workflow efficiency in analysis.


Figure 7: Case 2: **A** Two objects are detected; **B** The autonomous driving vehicle primarily decelerates between timestamp 155 and 186; **C**: The control module releases the throttle at timestamp 158, turns the steering at 159, and presses the brake at 160; **D**: The importance of object 12059 increases as it approaches; **E** Unstable predictions on object 12059.

**Visual Analytics System**: Experts needed to assess the accuracy and efficiency of different views qualitatively when used in the exploration process. In this section, there are ten questions evaluating the accuracy and efficiency of five views introduced in Sect. 5 (i.e., 5 views and 2 sub-questions (accuracy/efficiency) per view) where the experts needed to assign a score between 1 and 5 to evaluate the accuracy or efficiency of the specific view. The overall accuracy of the five views is 20.5/25 points on average, while the efficiency is 19.67/25 points. This reflects that the visual analytics system has a positive effect and can help domain experts analyze unexpected behaviors accurately and efficiently.

**Algorithms**: This part includes two questions corresponding to AHP/TOPSIS and DTW, respectively. Each of the questions has two sub-questions, one for the rationality and the other for the efficiency of the corresponding algorithm. Each sub-question is worth 5 points, as in the Visual Analytics System part.. Rationality refers to whether the algorithm is suitable for the analysis task, while efficiency refers to whether the algorithm can help experts gain insight efficiently. The overall rationality and efficiency scores for both sets of algorithms are 8.33/10 and 8.67/10 on average, respectively. One expert commented that "You calculate importance using AHP and TOPSIS, which are approximate methods. This is a good solution given that you do not have access to confidential data from the autonomous driving control center" regarding the rationality of applying AHP/TOPSIS.

In general, the workflow, visual analytics system, and algorithms received positive feedback from domain experts. This demonstrates the effectiveness of our approach.

During the free exploration phase, experts proposed some details of the cases based on their explorations. They suggested that a possible explanation for the unexpected prediction of the object in case 1 is that the velocity of the object is too high. To support this explanation, the experts selected the object and opened the perception tab (Fig. 6-C). The maximum velocity for the object exceeds $7m/s^2$, which is abnormal for an object approaching an intersection. The experts further commented that if the velocity measurement is accurate, the prediction should be optimized to be more conservative when dealing with objects in such cases. On the other hand, if the velocity measurement is inaccurate, the perception module should be refined to improve the task of measuring object velocity. For the pedestrian in case 2, the experts suggested that the behaviors of pedestrians are quite flexible and unpredictable. The autonomous driving system needs a few timestamps to fully ensure that the pedestrian does not intend to cross the road. The improvements for the algorithms should focus on shortening the time for judging the intention of the pedestrian.

There are some valuable suggestions proposed by the experts that warrant discussion. However, many of these suggestions are limited by the availability of data. For instance, one expert emphasized the importance of map information in autonomous driving systems, stating, "The map information is a crucial component of the autonomous driving system. Whether it is SD map or HD map data, it should be able to be displayed and presented in the design." Unfortunately, obtaining sensitive information, particularly road data, can be difficult due to various business and legal considerations. Additionally, in the perception module of the Decision-making Pipeline View, one expert wondered, "Is it possible to further refine this classification? For example, by having separate categories for trucks or cars within the 'vehicle' category, or by creating a specific category for SUVs?" This request is hindered by the limitation of the available data, which only provides coarse-grained information.

While it is currently difficult to address all of the suggestions due to limited data availability, it is worth noting that some of them hold value. For instance, one expert suggested that we can expand the mathematical features beyond derivatives. This includes calculating the relative velocity between objects and the autonomous driving vehicle. Furthermore, incorporating metrics about objects in the environment can enhance the system's understanding of how the autonomous driving system interacts with its surroundings. Additionally, upgrading the case view to support case replay enables repetitive analysis of cases for deeper insights.

## 7 Discussion And Conclusion

There are some points worth discussing regarding the approach and system we propose.

The Metric View (Fig. 1-A) enables domain experts to easily identify periods with unexpected behaviors. However, this convenience comes at the cost of concealing the precise values of the metrics. Nonetheless, these values can be accessed from various Module Tabs (Fig. 1-D). We can explore methods to integrate the exact values into the Metric View to eliminate the need for navigating to the module tabs to obtain them.

The Environment View (Fig. 1-B) provides the geographic context for the autonomous driving car's location. The elements in this view are comprehensive in order to enhance cooperation with other views in the system. It is worth considering if any elements can be removed without causing confusion and making the view less cluttered.

The Case View serves as an interface for recording observations, which is acknowledged by domain experts. Beyond the current record format, which only contains metrics and their values, we can extend the supported format for recording. For example, we can record the density map of prediction points.

Overall, in this paper, we propose a visual analytics approach that considers both the internal workings of specific modules and their context within the overall decision-making pipeline. We showcase two case studies that demonstrate the usability of our system in analyzing the autonomous driving system under cases with unexpected behaviors. Our system also helps in organizing observations that can be submitted to algorithm engineers for further analysis or algorithm improvement.

## Acknowledgments

This work is supported by Natural Science Foundation of China (NSFC No.62202105) and Shanghai Municipal Science and Technology General Program (No. 21ZR1403300), Sailing Program (No. 21YF1402900).

## References

[1] N. Akai, L. Y. Morales, T. Yamaguchi, E. Takeuchi, Y. Yoshihara, H. Okuda, T. Suzuki, and Y. Ninomiya. Autonomous driving based on accurate localization using multilayer lidar and dead reckoning. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6. IEEE, 2017.

[2] M. Al-Qizwini, I. Barjasteh, H. Al-Qassab, and H. Radha. Deep learning algorithm for autonomous driving using googlenet. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 89–96. IEEE, 2017.

[3] M. Bansal, A. Krizhevsky, and A. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018.

[4] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd international conference on knowledge discovery and data mining*, pp. 359–370, 1994.

[5] M. Bojarski, A. Choromanska, K. Choromanski, B. Firner, L. J. Ackel, U. Muller, P. Yeres, and K. Zieba. Visualbackprop: Efficient visualization of cnns for autonomous driving. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4701–4708. IEEE, 2018.

[6] A. Broggi, M. Buzzoni, S. Debattisti, P. Grisleri, M. C. Laghi, P. Medici, and P. Versari. Extensive tests of autonomous driving technologies. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1403–1415, 2013.

[7] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE international conference on computer vision*, pp. 2722–2730, 2015.

[8] L. Chi and Y. Mu. Deep steering: Learning end-to-end driving model from spatial and temporal visual cues. *arXiv preprint arXiv:1708.03798*, 2017.

[9] K. Chitta, A. Prakash, and A. Geiger. Neat: Neural attention fields for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15793–15803, 2021.

[10] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 4693–4700. IEEE, 2018.

[11] T. J. Crayton and B. M. Meier. Autonomous vehicles: Developing a public health research agenda to frame the future of transportation policy. *Journal of Transport & Health*, 6:245–252, 2017.

[12] F. Dong, Y.-N. Zhao, and L. Gao. Application of gray correlation and improved ahp to evaluation on intelligent u-turn behavior of unmanned vehicles. In *2015 8th International Symposium on Computational Intelligence and Design (ISCID)*, vol. 1, pp. 25–29. IEEE, 2015.

[13] L. Gou, L. Zou, N. Li, M. Hofmann, A. K. Shekar, A. Wendt, and L. Ren. Vatld: A visual analytics system to assess, understand and improve traffic light detection. *IEEE transactions on visualization and computer graphics*, 27(2):261–271, 2020.

[14] W. He, L. Zou, A. K. Shekar, L. Gou, and L. Ren. Where can we help? a visual analytics approach to diagnosing and improving semantic segmentation of movable objects. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):1040–1050, 2021.

[15] Y. Hou, C. Wang, J. Wang, X. Xue, X. L. Zhang, J. Zhu, D. Wang, and S. Chen. Visual evaluation for autonomous driving. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):1030–1039, 2021.

[16] S. Jamonnak, Y. Zhao, X. Huang, and M. Amiruzzaman. Geo-context aware study of vision-based autonomous driving models and spatial video data. *IEEE transactions on visualization and computer graphics*, 28(1):1019–1029, 2021.

[17] M. Jaritz, R. De Charette, M. Toromanoff, E. Perot, and F. Nashashibi. End-to-end race driving with deep reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2070–2075. IEEE, 2018.

[18] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah. Learning to drive in a day. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8248–8254. IEEE, 2019.

[19] J. Kim and J. Canny. Interpretable learning for self-driving cars by visualizing causal attention. In *Proceedings of the IEEE international conference on computer vision*, pp. 2942–2950, 2017.

[20] J. Kim, T. Misu, Y.-T. Chen, A. Tawari, and J. Canny. Grounding human-to-vehicle advice for self-driving vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10591–10599, 2019.

[21] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, et al. Towards fully autonomous driving: Systems and algorithms. In *2011 IEEE intelligent vehicles symposium (IV)*, pp. 163–168. IEEE, 2011.

[22] J. Liu and J.-M. Park. "seeing is not always believing": detecting perception error attacks against autonomous vehicles. *IEEE Transactions on Dependable and Secure Computing*, 18(5):2209–2223, 2021.

[23] A. Lotz, N. Russwinkel, T. Wagner, and E. Wohlfarth. An adaptive assistance system for subjective critical driving simulation: understanding the relationship between subjective and objective complexity. *Proceedings of the Human Factors and Ergonomics Society Europe*, pp. 97–108, 2020.

[24] K.-W. Meng, Y.-n. Zhao, L. Gao, and H.-c. Tan. Evaluation of the intelligent behaviors of unmanned ground vehicles based on information theory. In *CICTP 2015*, pp. 410–419. 2015.

[25] E. Ohn-Bar and M. M. Trivedi. Are all objects equal? deep spatio-temporal importance prediction in driving videos. *Pattern Recognition*, 64:425–436, 2017.

[26] E. Perot, M. Jaritz, M. Toromanoff, and R. De Charette. End-to-end driving in a realistic racing game with deep reinforcement learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 3–4, 2017.

[27] A. Prakash, K. Chitta, and A. Geiger. Multi-modal fusion transformer for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7077–7087, 2021.

[28] M. S. Ramanagopal, C. Anderson, R. Vasudevan, and M. Johnson-Roberson. Failing to learn: Autonomously identifying perception failures for self-driving cars. *IEEE Robotics and Automation Letters*, 3(4):3860–3867, 2018.

[29] T. L. Saaty. A scaling method for priorities in hierarchical structures. *Journal of mathematical psychology*, 15(3):234–281, 1977.

[30] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani. Deep reinforcement learning framework for autonomous driving. *arXiv preprint arXiv:1704.02532*, 2017.

[31] A. Sauer, N. Savinov, and A. Geiger. Conditional affordance learning for driving in urban environments. In *Conference on robot learning*, pp. 237–252. PMLR, 2018.

[32] E. Segel and J. Heer. Narrative visualization: Telling stories with data. *IEEE transactions on visualization and computer graphics*, 16(6):1139–1148, 2010.

[33] A. Tampuu, T. Matiisen, M. Semikin, D. Fishman, and N. Muhammad. A survey of end-to-end driving: Architectures and training methods. *IEEE Transactions on Neural Networks and Learning Systems*, 33(4):1364–1384, 2020.

[34] A. S. A. team. AVS streetscape: An open-source 3d autonomous vehicle and traffic simulation platform. `https://avs.auto/#/streetscape.gl/overview/introduction`, 2021. Accessed: April 1, 2023.

[35] B. Team. Baidu Apollo: An open autonomous driving platform. `https://apollo.baidu.com/`, 2021. Accessed: April 1, 2023.

[36] B. Team. Apollo studio. `https://apollo.baidu.com/workspace`, 2023. Accessed: Oct 2, 2023.

[37] Tesla. Autopilot — tesla. =https://www.tesla.com/autopilot, 2023. Accessed: Sep 10, 2023.

[38] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of field Robotics*, 25(8):425–466, 2008.

[39] O. S. Vaidya and S. Kumar. Analytic hierarchy process: An overview of applications. *European Journal of operational research*, 169(1):1–29, 2006.

[40] J. Wang, Y. Li, Z. Zhou, C. Wang, Y. Hou, L. Zhang, X. Xue, M. Kamp, X. Zhang, and S. Chen. When, where and how does it fail? a spatial-temporal visual analytics approach for interpretable object detection in autonomous driving. *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[41] Waymo. Waymo driver. =https://waymo.com/waymo-driver/?ncr, 2023. Accessed: Sep 10, 2023.

[42] J. Wei, J. M. Snider, J. Kim, J. M. Dolan, R. Rajkumar, and B. Litkouhi. Towards a viable autonomous driving research platform. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pp. 763–770. IEEE, 2013.

[43] C. Xu, W. Zhao, and C. Wang. An integrated threat assessment algorithm for decision-making of autonomous driving vehicles. *IEEE transactions on intelligent transportation systems*, 21(6):2510–2521, 2019.

[44] K. Xu, Y. Wang, L. Yang, Y. Wang, B. Qiao, S. Qin, Y. Xu, H. Zhang, and H. Qu. Clouddet: Interactive visual analysis of anomalous performances in cloud computing systems. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1107–1117, 2020. doi: 10.1109/TVCG.2019.2934613

[45] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8:58443–58469, 2020.

[46] W. Zhou, J. S. Berrio, S. Worrall, and E. Nebot. Automated evaluation of semantic segmentation robustness for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 21(5):1951–1963, 2019.