An Intelligent Self-Driving Truck System for Highway Transportation

Dawei Wang[1], Lingping Gao[2], Ziquan Lan[2], Wei Li[2]*, Jiaping Ren[2], Jiahui Zhang[2], Peng Zhang[2], Pei Zhou[2], Shengao Wang[2], Jia Pan[1]*, Dinesh Manocha[3] and Ruigang Yang[2]

[1] Department of Computer Science, The University of Hong Kong, Pokfulam, Hong Kong SAR, China, [2] Inceptio Technology, Shanghai, China, [3] Department of Computer Science, University of Maryland, College Park, MD, United States

Recently, there have been many advances in autonomous driving society, attracting a lot of attention from academia and industry. However, existing studies mainly focus on cars, extra development is still required for self-driving truck algorithms and models. In this article, we introduce an intelligent self-driving truck system. Our presented system consists of three main components, 1) a realistic traffic simulation module for generating realistic traffic flow in testing scenarios, 2) a high-fidelity truck model which is designed and evaluated for mimicking real truck response in real world deployment, and 3) an intelligent planning module with learning-based decision making algorithm and multi-mode trajectory planner, taking into account the truck's constraints, road slope changes, and the surrounding traffic flow. We provide quantitative evaluations for each component individually to demonstrate the fidelity and performance of each part. We also deploy our proposed system on a real truck and conduct real world experiments which show our system's capacity of mitigating sim-to-real gap. Our code is available at https://github.com/InceptioResearch/IITS.

Keywords: self-driving, heavy-duty truck, high-fidelity simulation, fuel efficiency, reinforcement learning

1. INTRODUCTION

Autonomous driving technology has become a billion-dollar market worldwide (Viscelli, 2018; Fortune Business Insights, 2020). Recently, the logistics truck with SAE level four (L4) autonomy gains more spotlights in venture capital and academia as it is believed to achieve massive production much earlier than the self-driving car. With more focused yet simpler scenarios defined in operational design domains (ODD), such as highway transportation, the self-driving truck has lower requirements for perception and prediction than the self-driving car. However, there are still several challenges that need to be resolved in the self-driving truck system including precise control with complex truck system dynamics and corresponding truck aware decision making and motion planning algorithms in highway traffic flow. In this article, we tackle the problem of developing a practical self-driving system for a highway truck, especially focusing on the planning and control (PnC) modules, which largely differ from passenger cars.

It is well known that the deep learning technique forms the cornerstone of the modern autonomous driving system, which benefits the whole system from perception to localization, decision making, motion planning, and control. With regard to the PnC modules, even several learning-based algorithms, especially reinforcement learning methods (Ulbrich and Maurer, 2013; Sallab et al., 2017; Codevilla et al., 2018; Wang et al., 2019) are proposed in the most recent years, they cannot seamlessly deploy to the self-driving truck system. First, the truck's system

dynamics are much more complex compared to a passenger car due to its low power/mass ratio, the time delay of internal engine control/air brake, prominent disturbance during gear shifting, wind, and slight road grade, etc. (Lu et al., 2004). The truck's dynamic model aware PnC algorithms should be developed to meet the unique requirement of trucks. Second, reinforcement learning-related algorithms highly depend on simulation environments, which injects a wide gap between the simulation and real world operation. A straightforward way to mitigate the sim-to-real gap is system identification, which identifies the exact physical/dynamical parameters of the environment relevant to the task and models it in simulation precisely. In recent years, several simulators have been developed for autonomous driving techniques, e.g., Carla (Dosovitskiy et al., 2017), AirSim (Shah et al., 2018). However, these sophisticated simulators are mainly developed for cars and perception algorithms, extra development is still required for heavy-duty truck simulation.

In this article, we introduce an intelligent self-driving autonomous truck system combining realistic traffic simulation and high-fidelity truck simulations for mitigating the sim-to-real gap. We build the system based on the service oriented middle-ware ROS2, which makes all modules decoupled and independent with each other. Then, we developed the simulation modules including a realistic truck model based on the data collected from a real truck, and construct a simulated traffic environment based on real highway roads and realistic traffic flow. On the basis of those simulation modules, we develop an intelligent planning module for trucks, including a reinforcement learning-based decision maker and a multi-mode trajectory planner.

The presented system has been validated by both numerical and real world experiments. First, we conduct several experiments to validate the fidelity of our simulation modules and the results show that our simulation modules are highly close to the real world. Then, we conduct a test in the simulation environment for comparison between our proposed intelligent decision maker and rule-based decision. Finally, we deploy our system along with the pre-trained model to the real truck and demonstrate that the proposed system significantly mitigates the reality gap. In summary, the contributions of this article are:

- A complete self-driving truck system for real world logistics operation. The performance of each part, as well as the whole system, is examined with various numerical and real-world experiments.
- A intelligent planning framework for self-driving truck system, covering a learning-based decision maker, multi-mode trajectory planner, increasing the system ability for interaction with complex traffic scenarios.
- To tackle the challenge of sim-to-real gap and real-world deployment, we adopt a system identification method to develop a realistic traffic simulation and high-fidelity truck simulation platform, which have been demonstrated their fidelity by real world experiments.
- During our investigation, we realize that there is no truck simulation platform that is easy to access for academia. To promote the autonomous driving truck society, our system including a high-fidelity truck model and traffic simulator is released to the public.

2. RELATED STUDY

2.1. Simulation Techniques in Autonomous Driving

Many simulatorshave recently been developed for autonomous driving with different focuses, such as perception realism, traffic flow, and vehicle model.

2.1.1. Integrated Simulation Platform

Many simulators adopt computer graphics techniques to construct and render realistic environments, simulating one or more perception data channels, such as RGB-D images, LiDAR, and object segmentation etc. Popular simulators include Intel's Carla (Dosovitskiy et al., 2017), Microsoft's AirSim (Shah et al., 2018), NVIDIA's Drive Constellation (NVIDIA, 2017), and Google/Waymo's CarCraft (Madrigal, 2017). A more sophisticated simulator employs a data-driven approach to render photo-realistic environments, such as Baidu's AADS (Li et al., 2019). However, these works mainly focus on generating realistic perception data, which are more suitable for computer vision tasks.

2.1.2. Traffic Flow Simulation

There are simulators focusing on traffic flows, such as SUMO (Krajzewicz et al., 2012), Vissim (Fellendorf and Vortisch, 2010), and HighwayEnv (Leurent, 2018). In particular, SUMO provides editable traffic scenarios with heterogeneous traffic agents, including road vehicles, public transport, and pedestrians. Integrating SUMO and Carla, SUMMIT (Cai et al., 2020) focuses on simulating urban driving in massive mixed traffic. Several advanced studies by Feng et al. (2021) and Yan et al. (2021) are proposed, leveraging reinforcement learning techniques and MDP in traffic simulation, which conduct a naturalistic and adversarial environment for driving intelligence testing. HighwayEnv offers a simulator for behavioral planning in autonomous driving, which is widely used as an environment to train deep learning algorithms for high-level decision making. Although there are various works developed with HighwayEnv and SUMO, none of them has ever migrated their learned model to real world vehicles because of the huge gap between simulation and real vehicles.

2.1.3. Truck Simulation Platforms

Most aforementioned simulators assume the vehicle model of a passenger car. We are particularly interested in truck simulation which differs significantly from car simulation in terms of kinematics and dynamics models. Well-known truck simulators include TruckSim (Corporation, 2021) and EuroTruck (AG, 2021). Among them, EuroTruck is essentially a game with a python wrapper (MarsAuto, 2017), without access to the detail of its underlying truck model. On the other hand, TruckSim is commercial software that cannot be easily accessed by the public.

In this article, we describe the intelligent autonomous truck system, which also contains a high-fidelity autonomous simulation platform for trucks, integrating simulators with various strengths, such as Carla, SUMO, and TruckSim, with the aim to facilitate the development and evaluation of autonomous truck algorithms.

2.2. Decision Making and Planning in Autonomous Driving

Schwarting et al. (2018) provide a detailed review of the schema of the decision making and planning components in autonomous driving, which divides them into three categories: sequential planning, behavior-aware planning, and end-to-end planning. In our proposed system, we adopt a sequential planning framework that contains a decision maker and a planner sequentially. Hence, we briefly review the previous studies of these two aspects. Many existing approaches are proposed to the decision-making problems for autonomous cars. Ulbrich and Maurer (2013) apply an online Partially Observable Markov Decision Process (POMDP) to accommodate inevitable sensor noise and make decisions in urban traffic scenarios. Wang et al. (2018) present a reinforcement learning approach for lane-change maneuver. They deploy a DQN network to make decisions for lane change and with a safety guarantee. With a high-level decision, low-level planners are then used to generate feasible driving trajectories. Low-level trajectory planners for autonomous driving trucks include polynomial curves (Piazzi et al., 2002), state lattice (Ferguson et al., 2008), and the A* family (Urmson et al., 2008). However, compared to cars, trucks have more complex kinematics and challenging dynamics, which makes the existing decision making and planning system hard to directly migrate to heavy duty trucks.

3. SYSTEM OVERVIEW

Our proposed intelligent self-driving truck system contains three components: traffic simulation module, truck model, and intelligent planning module as shown in **Figure 1**. The traffic simulation module is designed to simulate traffic flow substituting for the perception result in real world experiments. The truck model is developed for reproducing the real truck in a simulated environment precisely. The intelligent planning module consists of a reinforcement learning based decision maker and a multi-mode trajectory planner, truck's constraints, road slope changes, and the surrounding traffic flow.

The data flow of our proposed system can be summarized as: first, the traffic simulation module will generate realistic traffic flow interacting with ego-vehicle. Then the simulated surrounding state for the ego-vehicle will be fed to a reinforcement learning based decision making module. The decision maker will output a high-level decision for the ego-vehicle. After that, the planning module will conduct a feasible trajectory considering high-level decisions, map information, collision avoidance, and fuel efficiency. Then the control module will execute the trajectory and send the corresponding control command to the high-fidelity truck model. Finally, the truck model will output the high-fidelity response.

The rest of this article is organized as follows: In Section 4, we describe the technical detail of our realistic traffic simulation first and conduct an experiment to demonstrate that our simulation can generate similar traffic which is highly close to real world data. In Section 5, we describe the implementation detail of our truck model and demonstrate its fidelity. In Section 6, we present the intelligent planning module including reinforcement learning based decision making, multi-mode trajectory planner, and fuel efficient predictive cruise control algorithm. We conduct several numerical experiments to evaluate the decision maker and fuel-saving performance. Finally, we deploy our system to the real truck and illustrate the running result in Section 7.

4. REALISTIC TRAFFIC SIMULATION

4.1. Implementation Details

We develop the traffic simulation module based on SUMO (Krajzewicz et al., 2012) to generate dynamic traffic environments for the ego-truck (as shown in **Figure 2**). The traffic simulation enhances SUMO with more a friendly Python interface for configuration and integration and a more intelligent mode for RL training. Technically, the traffic simulation module, which is a ROS2 Python node, consists of three sub-modules: map network, traffic controller, and vehicle meta-information. The map network module describes the road connections, routes the traces, locates the vehicles, etc. The traffic controller module provides high-level traffic control using traffic lights. The vehicle meta-information module gives the attributions of vehicles, and we can adjust the vehicle behavior through this module. For different usage and requirements, we develop four modes for realistic traffic simulation modules:

- *PureSim Mode.* This mode simulates the traffic flow with the typical car-following model [e.g., Intelligent Driver Model (IDM) Treiber and Helbing, 2001] and lane-change model (e.g., MOBIL Kesting et al., 2007).
- *InterSim Mode.* This mode is built based on PureSim and can support interactions with the ego-truck when simulating the traffic flow.
- *ReSim Mode.* This mode is designed to re-simulate the traffic flow with the saved configurations. ReSim mode can guarantee the determinism of simulation and yield consistent traffic, which is needed by some learning algorithms training.
- *RepSim Mode.* RepSim Mode is the enhanced version of ReSim, which could not only restore the saved configurations but also the stored traffic flow, e.g., trajectories of vehicles. Powered by our unique truck trajectories and some public trajectories datasets, the simulated traffic from RepSim Mode is deterministic and with very high-fidelity.

In order to describe map information in our simulation pipeline, we adopt a widely used description format, ASAM OpenDRIVE, which provides a common base for describing road networks with extensible markup language (XML) syntax, with the file extension xodr. We can support not only the maps created by hands but also the HD map of the real road networks. With the diverse maps as the environments, we can generate various traffic flows.
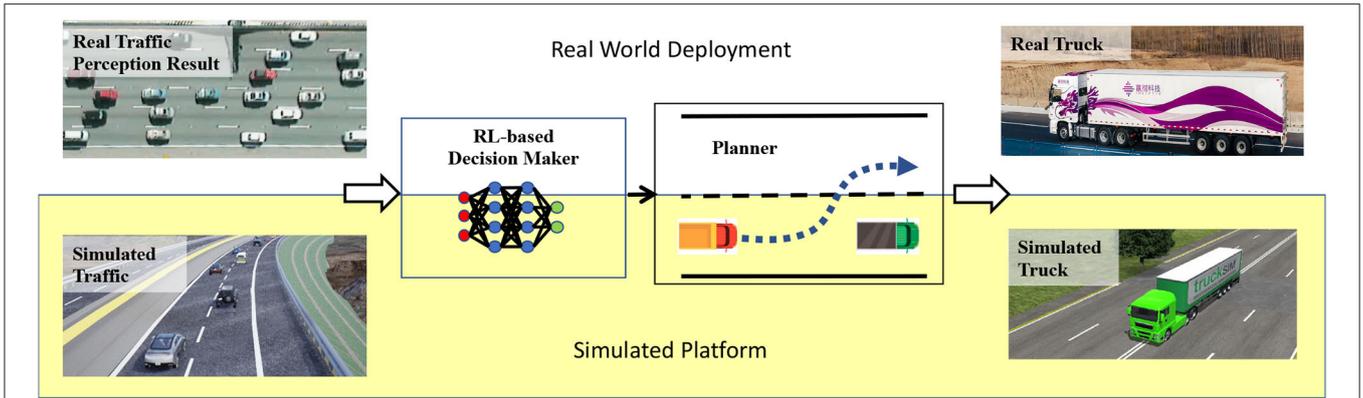
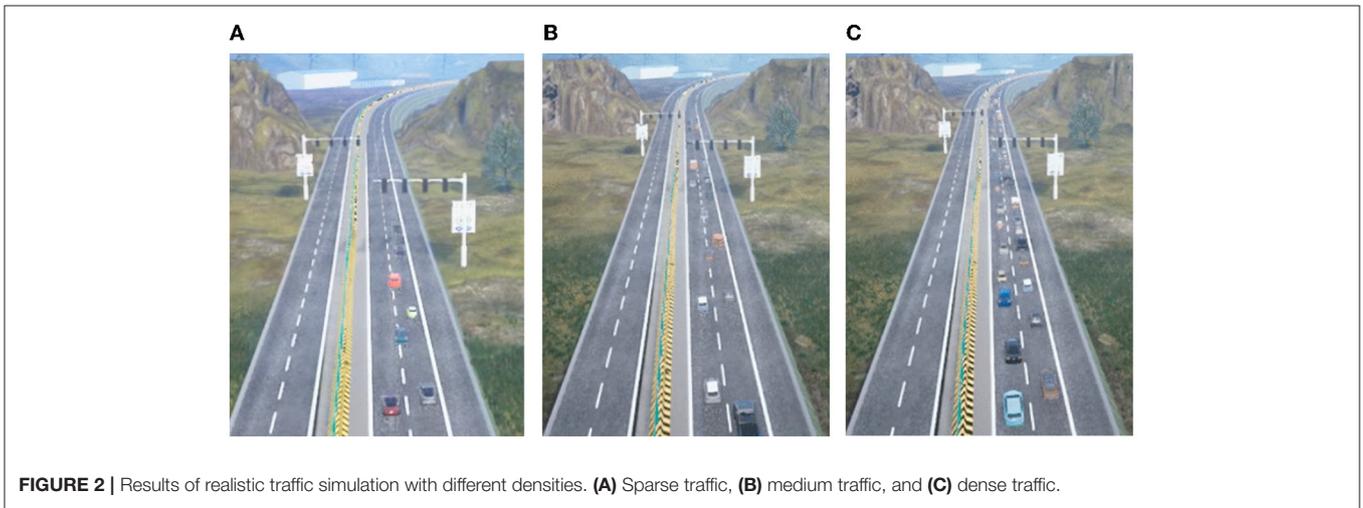**FIGURE 1 |** The overview of our proposed intelligent self-driving truck system.

FIGURE 2 | Results of realistic traffic simulation with different densities. **(A)** Sparse traffic, **(B)** medium traffic, and **(C)** dense traffic.
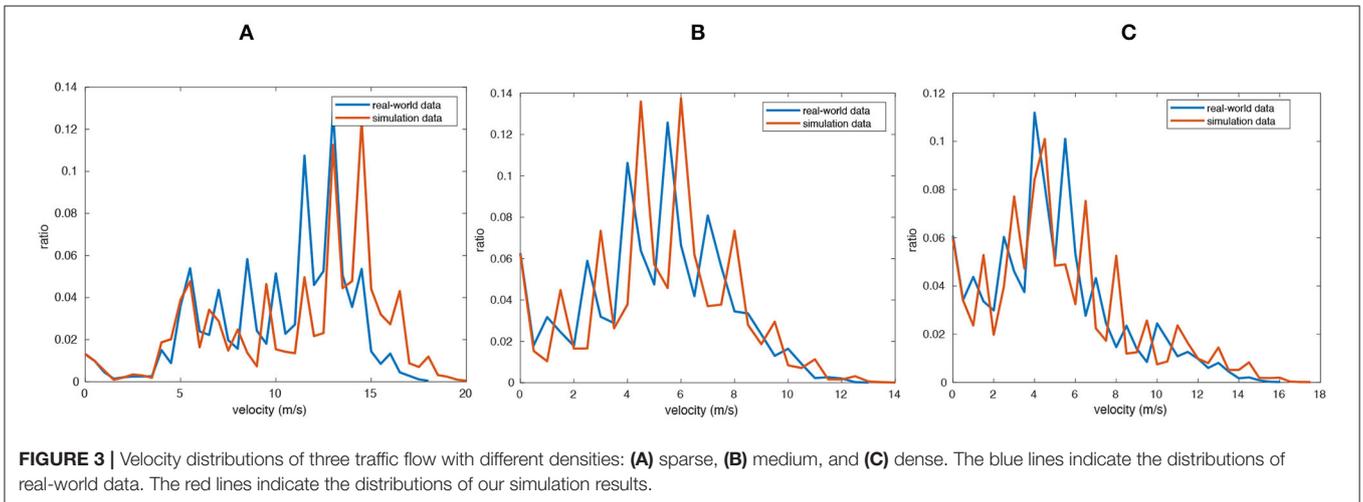

FIGURE 3 | Velocity distributions of three traffic flow with different densities: **(A)** sparse, **(B)** medium, and **(C)** dense. The blue lines indicate the distributions of real-world data. The red lines indicate the distributions of our simulation results.

4.2. Experiment Result

Currently, there are two types of evaluation for traffic simulation: user studies and statistical validations (Chao et al., 2020). In this

article, we compare the velocity distributions with those of the real-world datasets, similar to Sewall et al. (2011). Specifically, we choose three datasets with different traffic flow densities (sparse,

middle, and dense), and the simulation result is generated under RepSim mode in the system, and the real world data is collected in our test site/road in Jinan. Then, we divide our traffic flow into three levels of traffic volumes and compare the velocity distribution of real data and simulated traffic. The velocity distributions with different traffic densities are shown in **Figure 3**. The result shows that our simulated traffic flow is highly close to real world data.

5. HIGH-FIDELITY TRUCK SIMULATION PLATFORM

5.1. Overview

An accurate vehicle model forms the cornerstone of high-fidelity simulation. We implement our real truck model in the system to mimic the real truck experiments because both the truck's powertrain system and kinematics are important for system deployment and also machine learning approaches training. In addition, the administrative authority has not approved self-driving truck road testing on open roads, we also have to test our methods in simulation instead of open road testing. We build our truck model based on a widely used truck simulator, Trucksim. At the same time, we re-implement the powertrain system and brake system with Simulink based on our real truck.

Details of our truck model are shown in **Figure 4**. The green block is the powertrain system, which consists of an engine, gearbox, clutch, engine controller, and gearbox controller. The dark gray block is the brake system, consisting pneumatic brake system, brake control suspension, tire, tire brake mechanism, and traction machine. The white block is the interface between Trucksim and Simulink. Trucksim is responsible for differential mechanism, vehicle body, and trailer body.

The interface of our truck model complies with SAE J1939 (SAE, 1939), which is the recommended vehicle bus standard published by the Society of Automotive Engineers (SAE), widely used in the heavy-truck industry. Thus, the algorithms developed on our truck model will easily meet the satisfaction of mass production on real trucks. Our truck model takes four inputs: Pedal (PedaPos_pct), Brake Control Mode (XBR1_ctrl_mode), Steering Angle (SteerWhAngle), and Deceleration (XBR1_ExternalAccele). The output includes 16 variables, including engine torque, engine speed, and shift position.

The prototype of our truck model is a truck manufactured by Sinotruk Ltd, as shown in **Figure 9A**, which is 12 wheeler heavy truck. The full load vehicle weight of our truck is 55t, and the empty load weight is 19t. The parameters and details in our model are from public data or provided by the manufacturer.

5.2. Model Design

5.2.1. Kinematics

Figure 5A shows the kinematics of the vehicle-trailer system, which contains two parts: a bicycle model with a unicycle model. The global reference is $\{XOY\}$, the body-fixed reference on the bicycle model is $\{xoy\}$, and the body-fixed reference on the

unicycle model is $\{x'o'y'\}$. The system does not consider tire-slip angles, the vehicle velocity is v and the trailer velocity is v'. L_1 refers to the tongue length and L_2 is the hitch length. The wheel base is L. ϕ, θ, θ' refer to the steering angle, vehicle heading, and trailer heading. Therefore, the vehicle-trailer system model is directly given as follows:

$$
\begin{aligned}
\dot{x} &= v * cos(\theta) \\
\dot{y} &= v * sin(\theta) \\
\dot{\theta} &= (v * tan(\phi))/L \\
\dot{\phi} &= \omega \\
\dot{\psi} &= -v * \left(\frac{sin(\psi)}{L_2} + \frac{L_1}{LL_2} * cos\psi \, tan\phi + \frac{tan\phi}{L}\right)
\end{aligned}
\tag{1}
$$

5.2.2. Dynamics

The vehicle dynamics can be first modeled as a tire-ground model *via* Newton's law and Lagrange's equations:

$$
\begin{aligned}
F_{xf} cos\phi + F_{xr} - F_{yf} sin\phi &= -F_c sin\beta + m\dot{v}cos\beta \\
F_{yf} cos\phi + F_{yf} + F_{xf} sin\phi &= F_c cos\beta + m\dot{v}sin\beta \\
J\ddot{\theta} &= F_{xf} sin\phi L_f + F_{yf} cos\phi - F_{yf} L_r
\end{aligned}
\tag{2}
$$

Figure 5B illustrates the truck dynamics with the trailer, which can be decomposed into two aspects: longitudinal part and lateral part.

5.2.2.1. Longitudinal–Drive-line and Brake Subsystems

Regarding the drive-line subsystem, also known as Powertrain, our intelligent truck system introduces truck engines, ECU (Engine Control Unit), AMT (Automated Manual Transmission), and TCU (Transmission Control Unit). We built those 4 modules from scratch with MATLAB Simulink using a system identification method based on our truck prototype.

5.2.2.2. Lateral–Dynamic Yaw-Sideslip Model

A dynamic yaw-sideslip model shown in **Figure 5B** is designed to describe the truck's lateral motion in Simulink instead of the TruckSim Model. The Lagrangian mechanics is introduced to provide a governing equation for a tractor-trailer dynamic system with tandem (multiple) axles on tractor and trailer, the governing equation can be written as:

$$
\frac{d}{dt}\left(\nabla_{\dot{q}} T\right) = Q,
\tag{3}
$$

where $q = \left[x_c, y_c, \beta_c, \beta_t\right]^T$ is the generalized coordinates, Q is the generalized force, and T represents the total kinetic energy.

For the energy term on the left-hand side of Equation (3), the combined vehicle's total kinetic energy can be written as:

$$
T = \frac{I_c \, \omega_c^2}{2} + \frac{I_t \, \omega_t^2}{2} + \frac{m_c \left(v^2 + v_{yc}^2\right)}{2} + \frac{m_t \left(v^2 + v_{yt}^2\right)}{2},
\tag{4}
$$

where I stands for rotational inertia, ω stands for yaw rate, m stands for mass, v represents the longitudinal speed, and the
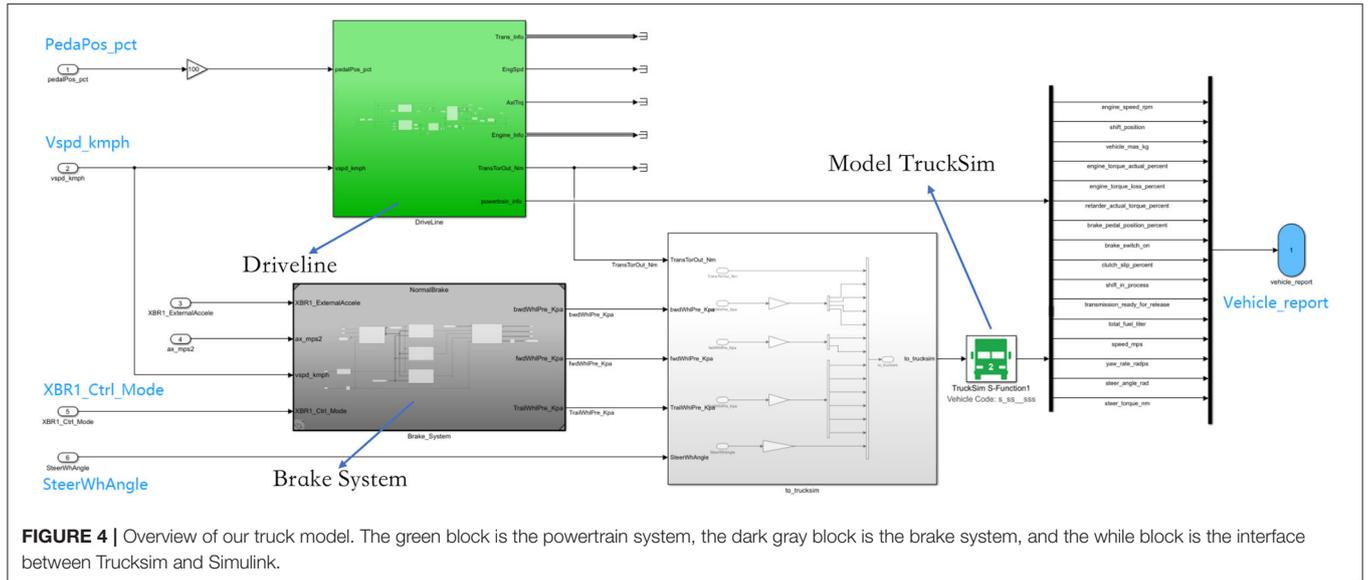
FIGURE 4 | Overview of our truck model. The green block is the powertrain system, the dark gray block is the brake system, and the while block is the interface between Trucksim and Simulink.
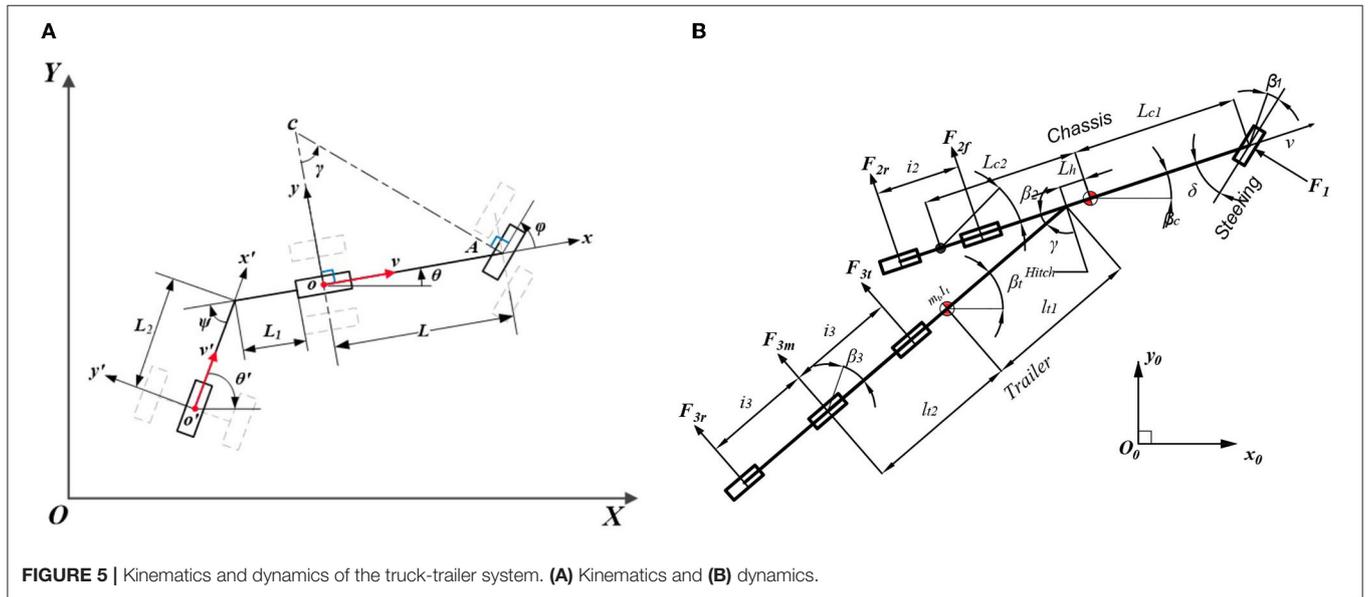

FIGURE 5 | Kinematics and dynamics of the truck-trailer system. **(A)** Kinematics and **(B)** dynamics.

additional subscript c and t are used to indicate the tractor (chassis) and the trailer, respectively. The kinematics of the tractor-trailer system also stipulates:

$$v_{yt} = v_{yc} - L_h \omega_c - L_{t1} \omega_t. \tag{5}$$

Then the gradient of energy can be written as:

$$\nabla_{\dot{q}} T = \begin{pmatrix} m_c + m_t & 0 & 0 & 0 \\ 0 & m_c + m_t & -L_h m_t & -L_{t1} m_t \\ 0 & -L_h m_t & m_t L_h^2 + I_c & L_h L_{t1} m_t \\ 0 & -L_{t1} m_t & L_h L_{t1} m_t & m_t L_{t1}^2 + I_t \end{pmatrix} \dot{q}, \tag{6}$$

where $\dot{q} = \left[v, v_{yc}, \omega_c, \omega_t \right]^T$.

For the force term on the right-hand side of Equation (3), the virtual work done by DOF perturbations can be written as:

$$\begin{aligned}
\Delta W &= (-F_1 \delta - (F_{3f} + F_{3m} + F_{3r}) \Delta \beta_t) \Delta x_c \\
&+ F_1 \sqrt{1 - \delta^2} (\Delta y_c + L_{c1} \Delta \beta_c) \\
&+ F_{2f} (\Delta y_c - (L_{c2} - i_2) \Delta \beta_c) \\
&+ F_{2r} (\Delta y_c - (L_{c2} + i_2) \Delta \beta_c) \\
&+ F_{3f} (\Delta y_t - (L_{t2} - i_3) \Delta \beta_t) \\
&+ F_{3r} (\Delta y_t - (L_{t2} + i_3) \Delta \beta_t) \\
&+ F_{3m} (\Delta y_t - L_{t2} \Delta \beta_t) + m_c \omega_c v \Delta y_c + m_t \omega_t v \Delta y_t,
\end{aligned} \tag{7}$$

where $y_t = y_c - \beta_c L_h - \beta_t L_{t1}$ is complying with the kinematic constraints. In order to obtain a linear system, coefficient $\sqrt{1 - \delta^2}$ is approximated as 1. The generalized force term Q can be represented as the gradient vector of the virtual work

with respect to the virtual displacements along each DOF (Δq):
$Q = \nabla_{\Delta q}(\Delta W)$.

By defining the system states as $s = \left[v_{yc}, \omega_c, \omega_t, \gamma\right]^T$ and front-wheel steer angle input δ, the dynamics of the states can be obtained as:

$$H\dot{s} = A_h s + B_h \delta, \tag{8}$$

in which A_h, B_h are calculated from differentiating Q with respect to s, and H is derived from reducing the kinetic potential in Equation (6):

$$H = \begin{pmatrix} m_c + m_t & -L_h\,m_t & -L_{t1}\,m_t & 0 \\ -L_h\,m_t & m_t\,L_h{}^2 + I_c & L_h\,L_{t1}\,m_t & 0 \\ -L_{t1}\,m_t & L_h\,L_{t1}\,m_t & m_t\,L_{t1}{}^2 + I_t & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{9}$$

Finally, the state space model in the formal form is:

$$\dot{s} = As + B\delta, \tag{10}$$

where $A = H^{-1}A_h$ and $B = H^{-1}B_h$.

5.3. Fidelity Evaluation

5.3.1. Fidelity of Longitudinal Model

To verify the accuracy of the longitudinal model, we collect some data from a real truck running with a human driver ten times, then replay the truck control commands in our simulator, and compare the difference in vehicle response. **Figure 6** shows some representative results of the experiments on verifying the driveline and brake subsystems, with **Figures 6A,B** corresponding to the truck's acceleration and brake motions, respectively, where the red lines indicate the simulated results and the blue lines are the real data from trucks. For the acceleration test, we keep the throttle position at 40% for 40 s, and collect the vehicle status about gear (0-10), longitudinal acceleration (m/s^2), and velocity (m/s) from real vehicle and simulation module simultaneously. The result is shown in **Figure 6A**, the red lines are very close to the blue lines, demonstrating that our simulated model's response is highly close to the real truck. For the brake test, we keep the throttle at 0 all the time and activate the brake at 17 s (CtrlMode means brake activation status, 0 indicates deactivated, 2 indicates brake activated. ExternAccel means deceleration, we keep it at $-1.5m/s^2$ during our test), and collect the longitudinal velocity (m/s), acceleration (m/s^2), and brake pressure (Kpa). The result is shown in **Figure 6B**, the red lines are close to their corresponding blue ones, demonstrating that our model's brake system is close to the real truck. Overall, the longitudinal velocity accuracy of all experiments is 89%.

5.3.2. Fidelity of Lateral Model

In this article, we adopt two lateral dynamic models, one is the TruckSim model which relies on a commercial license, another one is our proposed yaw-sideslip model, which will be released in our open-source system. The Trucksim model combines a steering system, solid axles, and tire system together so that the physical characteristics of the real truck can be restored to

the utmost extent. Same as the evaluation experiment of the longitudinal model, we replay the control commands in our simulator and compare the responses. Results in **Figure 7A**, the input command of steer is shown in the top figure and the longitudinal velocity during the testing is presented in the middle. We collect the yaw rate of the tractor (degree/s), the blue line indicates the actual yaw rate of the real truck, and the red line indicates the yaw rate response in our simulated model. The result shows that our truck model (TruckSim)'s lateral response is highly close to real truck experiments, and the overall lateral yaw rate steady accuracy is 92% in our experiments.

5.3.3. Fidelity of Dynamic Yaw-Sideslip Model

Since the TruckSim is a commercial, closed source software, we designed the dynamic yaw-sideslip model in Matlab Simulink as an alternative choice of TruckSim model, thus, we conduct an experiment to compare the dynamic yaw-sideslip model with the TruckSim lateral model. During the comparison experiment, we set the truck speed at 80 km/h, and we fed a step steering command of 15 degrees to each model. Results are shown in **Figure 7B**, we collect the yaw rate response of tractor and trailer, the lateral position of the tractor, and the hitch angle between the tractor and trailer, the blue lines show the result of the simulation results from TruckSim lateral model, and the red lines indicate the results from our proposed dynamic yaw-sideslip model. The result shows that our proposed yaw-sideslip lateral model is highly close to our TruckSim lateral model, which has already been demonstrated its fidelity in the previous section. In total, the overall accuracy of these experiments is 96%.

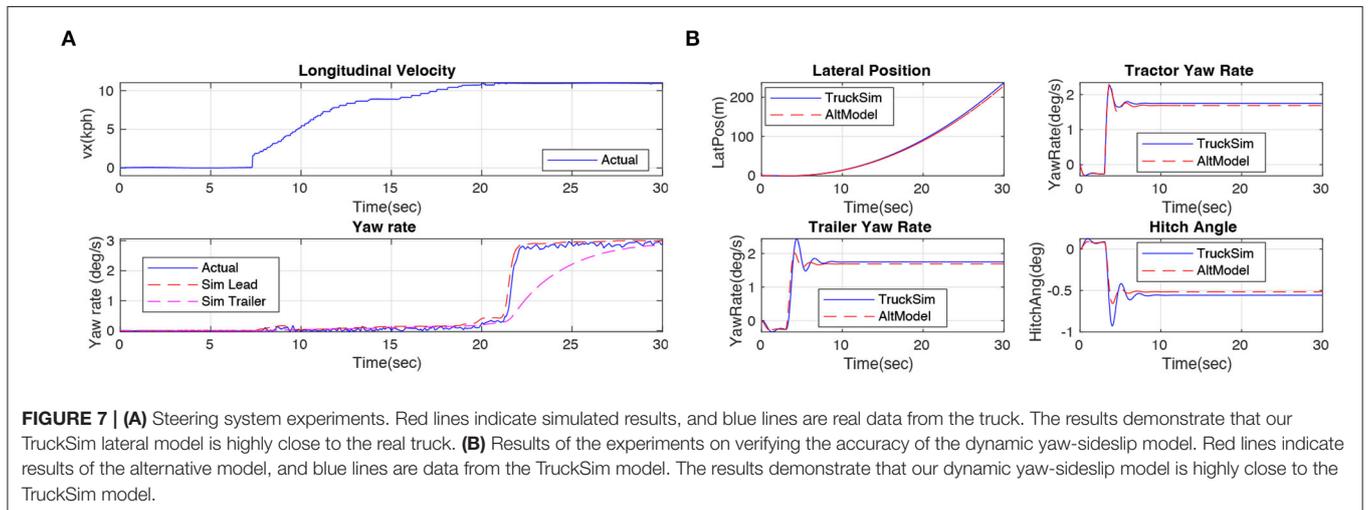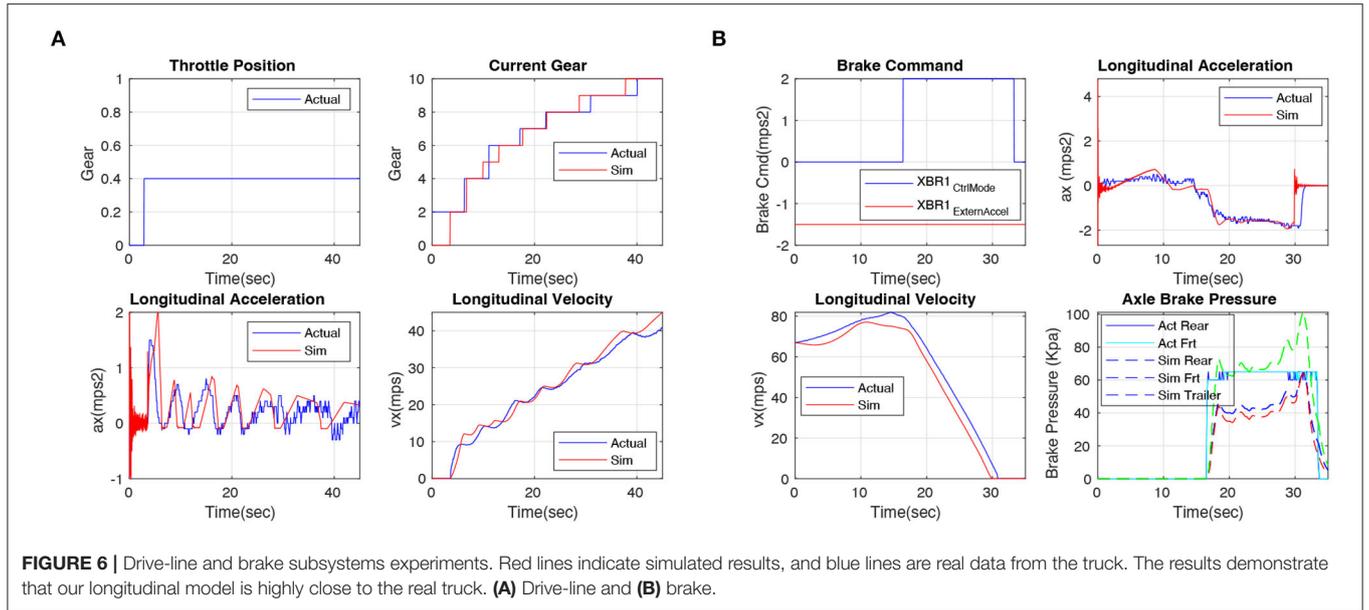## 6. INTELLIGENT DECISION AND PLANNING

In this section, first, we will describe the reinforcement learning based decision making, then we will present the technical detail of the planning module in our system, finally, we will conduct several experiments to evaluate our proposed intelligent decision making module.

6.1. Reinforcement Learning Based Decision Making

In this section, we introduce our reinforcement learning based decision maker, including problem formulation and network design.

6.1.1. Problem Formulation

The self-driving trucks evaluate and improve their decision-making policy by interacting with the environment including surrounding vehicles and lanes in a trial-and-error manner. This process can be formulated as a sequential decision-making problem, which can be solved using a reinforcement learning framework (Van Hasselt et al., 2016). In the RL settings, the problem is formulated as a Markov Decision Process (MDP), which is composed of a five-tuple $(\mathcal{S}, \mathcal{A}, r(s_t, a_t), \mathcal{P}(s_{t+1} \mid$

FIGURE 6 | Drive-line and brake subsystems experiments. Red lines indicate simulated results, and blue lines are real data from the truck. The results demonstrate that our longitudinal model is highly close to the real truck. **(A)** Drive-line and **(B)** brake.

FIGURE 7 | (A) Steering system experiments. Red lines indicate simulated results, and blue lines are real data from the truck. The results demonstrate that our TruckSim lateral model is highly close to the real truck. **(B)** Results of the experiments on verifying the accuracy of the dynamic yaw-sideslip model. Red lines indicate results of the alternative model, and blue lines are data from the TruckSim model. The results demonstrate that our dynamic yaw-sideslip model is highly close to the TruckSim model.

$s_t, a_t), \gamma$). At time step t, the agent selects the action $a_t \in \mathcal{A}$ by following a policy π in the current state. The agent is transferred to the next state s_{t+1} with the probability $\mathcal{P}(s_{t+1} \mid s_t, a_t)$ after executing a_t. Additionally, the environment returns a reward signal $r(s_t, a_t)$ to describe whether the underlying action a_t is good for reaching the goal or not. For brevity, we rewrite it as $r_t = r(s_t, a_t)$. By repeating this process, the agent interacts with the environment and obtains a trajectory $\tau = s_1, a_1, r_1, \cdots, s_T, a_T, r_T$ at the terminal time step T. The discount cumulative reward from time step t can be formulated as $R_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$, where $\gamma \in (0, 1)$ is the discount rate that determines the importance of future rewards. The goal of RL is to learn an optimal policy π^* that can maximize the expected overall discounted reward:

$$\pi^* = \arg \max_{\pi} E_{s, a \sim \pi, r} [R_1]. \tag{11}$$

Typically, two kinds of value functions are used to estimate the expected cumulative reward for a specific state:

$$\mathbf{V}^{\pi} (\mathbf{s}) = \mathbb{E}_{\pi} [R_1 | s_1 = s], \tag{12}$$

$$\mathbf{Q}^{\pi} (\mathbf{s}, \mathbf{a}) = \mathbb{E}_{\pi} [R_1 | s_1 = s, a_1 = a]. \tag{13}$$

To improve the robustness of the lane-change decision-maker based on reinforcement learning while reducing the difficulty of training, we discretize the action space of the lane-change problem and use the double DQN algorithm (Van Hasselt et al., 2016) to solve it. In reference to both double Q-learning (Hasselt, 2010) and DQN (Mnih et al., 2015), double DQN proposes to evaluate the greedy policy according to the online network but

using the target network to estimate its value. Its update is the same as for DQN but replacing the target Y_t^{DQN} with

$$\mathbf{Y}_t = R_{t+1} + \gamma Q\left(S_{t+1}, \arg\max_a Q\left(S_{t+1}, a, \theta_t\right), \theta_t^-\right). \quad (14)$$

Double DQN replaces the weight of the second network θ_t' with the weights of the target network θ_t^- for the evaluation of the greedy policy in comparison to double Q-learning. For the update method of the target network, double DQN still adopts the DQN method and remains a periodic copy of the online network.

6.1.2. State Space
In our formulation, the state s is defined as follows:

$$\mathbf{s} = \langle \mathbf{p}, \mathbf{v}, \mathbf{v}_t \rangle + \sum_i^N \langle^i o_p^t, {}^i o_v^t \rangle, \quad (15)$$

where p and v indicate the location and velocity of ego-car, v_t is the speed profile in the next 3 s, N is the number of the observable neighbors, o_p^t and o_v^t are relative position and velocity of neighbors, respectively.

6.1.3. Action Space
For the action a of the agent, we use R^3 vectors for our action space, which includes: change left (-1), straight forward (0), and change right (1).

6.1.4. Reward Function
As a key element of the RL framework, the reward drives the agent to reach the goal by rewarding good actions and penalizing poor actions. For a lane change process, safety and efficiency are the main concern. Therefore, our objective is to achieve the reference speed profile with a few lane changes as possible while ensuring safety.

The process of change lanes will not only increase the probability of danger but also reduce the efficiency of transportation. To avoid the meaningless lane change behavior, we give a penalizing reward r_{ch} when a lane change decision is made. In our simulator, we limit the self-driving trucks to only drive in two lanes of the road. That means when a decision to change lanes to the left is made while the truck is on the left lane, it stays in the current lane, but a penalizing reward r_{ch} will also be given.

For efficiency, self-driving should try to meet the requirements of driving at a reference speed generated by the planner. To do so, we define the following reward according to the speed of the truck:

$$\mathbf{r}_v = \lambda \left| v - v_{ref} \right|, \quad (16)$$

where v denotes the car's current speed, and v_{ref} is the reference speed planned by the speed planner last clock cycle while λ is a normalizing coefficient.

For safety, we hope that the truck can change lanes while ensuring safety. Therefore, we will use the rule-based method to determine whether the lane change decision at the current moment is dangerous according to the observation. If the truck

makes a lane change decision at a dangerous moment, we will give a larger penalty r_{sa}.

In general, our reward function goes as:

$$\mathbf{r} = r_{ch} + \lambda \left| v - v_{ref} \right| + r_{sa}. \quad (17)$$

Considering the safety and efficiency of truck transportation, we set $r_{ch} = -10, \lambda = 1, r_{sa} = -20$.

6.1.5. Network Design
In the DRL network, we take four consecutive observations, in the past 3 s $S_{t-3}, S_{t-2}, S_{t-1}, S_t$ as input. Such input can enable our agents to infer the motion of surrounding cars and, thus, make more reasonable lane change decisions. Because the method of discretizing the action space reduces the difficulty of the decision-making problem, we only use a few fully connected layers to build a policy network. The input and output layers have 216 and 3 neurons, respectively, while the total number of neurons in the hidden is (256, 512, 256). The architecture consists of two networks with the same structure: the value network for select action and the target network for evaluating the value of the underlying state. The online network's inputs are $O_t = [S_{t-3}, S_{t-2}, S_{t-1}, S_t]$ and $O_{t+1} = [S_{t-2}, S_{t-1}, S_t, S_{t+1}]$ to predict the value of the current state and action of the next state. The target network's input is $O_{t+1} = [S_{t-2}, S_{t-1}, S_t, S_{t+1}]$ to predict target value.
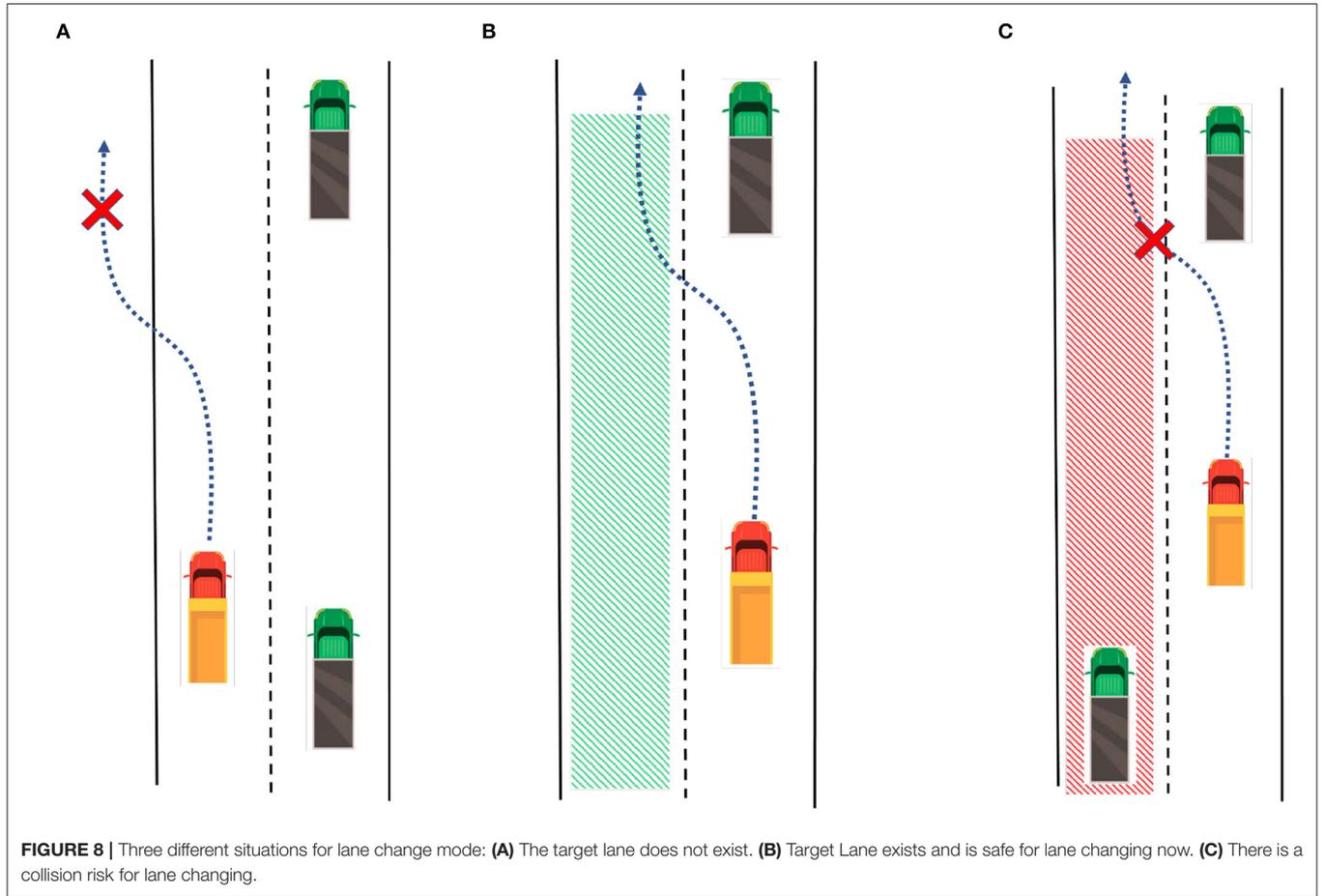
6.2. Trajectory Planning Module
To execute high-level decisions, we deploy a trajectory planning module, which can be divided into two modes: lane change mode and lane keeping mode. In addition, the reference speed profile for the trajectory is planned by a fuel efficient speed planner, which we adopt predictive cruise control (PCC) algorithm (Lattemann et al., 2004).

6.2.1. Fuel Efficient Speed Planner
Since fuel consumption achieves 30% of total operation cost in the logistic industry, fuel efficiency in the autonomous truck system becomes more and more important currently. PCC algorithm (Lattemann et al., 2004) is a widely adopted fuel saving method for heavy duty truck, which leverage the road slope change in front of the ego-vehicle and generate a sequence of control strategy (a reference speed profile in the future) to achieve fuel efficient operation goal. PCC algorithm can be formulated as an optimization problem, which is to find the optimal longitudinal distance trajectory $s^*(t)$ that minimizes:

$$J = \int_0^T \left(\dot{m}_f + k_a \ddot{s}(t)^2 \right) dt \quad (18)$$

FIGURE 8 | Three different situations for lane change mode: **(A)** The target lane does not exist. **(B)** Target Lane exists and is safe for lane changing now. **(C)** There is a collision risk for lane changing.

Subject to:

$$
\begin{aligned}
s_1 &= s(0) \\
s_T &= s_1 + v_{ref}\, T \\
\dot{s}_1 &= \dot{s}(0) \\
\dot{s}_T &= v_{ref} \\
\dot{s}_i &\leq V_{max}, i \in [2, 3, ..., n-1] \\
s_i &\leq s_i^{fence}, i \in [2, 3, ..., n-1]
\end{aligned}
\tag{19}
$$

where $\dot{m}_f$ is a nonlinear lookup table that computes the mass fuel flow rate based on the engine power shown in **Figure 11A**, K_a is a penalizing factor on acceleration so that the convex nature of J can be ensured. The engine power can be obtained from a vehicle dynamics model, as:

$$
p_e = \dot{s}\eta(sin(\theta(s)) + \mu(\dot{s}))gM_{veh} + \frac{1}{2}\rho_{air}A_f C_d\dot{s}^2 + \ddot{s}M_{veh}\dot{s} \tag{20}
$$

in which η is total power efficiency from engine torque to propulsion force, $\theta(s)$ is road gradient with respect to the distance ahead of vehicle, $\mu(\dot{s})$ is tire rolling friction, M_{veh} indicates the vehicle mass, ρ_{air} is the density of air, A_f means the front area of vehicle, and C_d is the air drag constant. We assume that $\dot{m}_f$ and

$\frac{d\dot{m}_f}{dp_e}$ are smooth for $p_e \in [0, p_{e,max}]$, the necessary condition on optimality states that, for any arbitrary small perturbation $\delta s(t)$:

$$
\int_0^T \left(\dot{m}_f \left(p_e(s, \dot{s}, \ddot{s}) \right) + k_a\ddot{s}^2 \right) dt =
$$
$$
\int_0^T \left(\dot{m}_f \left(p_e(s + \delta s, \dot{s} + \delta\dot{s}, \ddot{s} + \delta\ddot{s}) \right) + k_a(\ddot{s} + \delta\ddot{s})^2 \right) dt
\tag{21}
$$

Using Taylor expansion, we have:

$$
\dot{m}_f(s + \delta s, \dot{s} + \delta\dot{s}, \ddot{s} + \delta\ddot{s}) =
$$
$$
\dot{m}_f(s, \dot{s}, \ddot{s}) + \frac{\partial \dot{m}_f}{\partial s}\delta s + \frac{\partial \dot{m}_f}{\partial \dot{s}}\frac{d\delta s}{dt} + \frac{\partial \dot{m}_f}{\partial \ddot{s}}\frac{d^2\delta s}{dt^2} + H.O.T.
\tag{22}
$$

Therefore, the necessary condition then takes the following form:

$$
\int_0^T \left(\frac{\partial \dot{m}_f}{\partial s}\delta s + \frac{\partial \dot{m}_f}{\partial \dot{s}}\frac{d\delta s}{dt} + \frac{\partial \dot{m}_f}{\partial \ddot{s}}\frac{d^2\delta s}{dt^2} + 2k_a\frac{d^2s}{dt^2}\frac{d^2\delta s}{dt^2} \right) dt = 0
\tag{23}
$$

Then we employ the well-established finite element method to solve such an optimization problem (Liao-McPherson et al., 2018). Finally, we will get the solution:

$$
X = [s_1, \dot{s}_1, s_2, \dot{s}_2, ..., s_n, \dot{s}_n]^T \tag{24}
$$

Algorithm 1 Trajectory Planner and Speed Profile Assignment

1: Receive the high level decision a_t
2: **if** $a_t = 0$ **then**
3:     Query for the waypoints from current lane: $(s_1, s_2, s_3, ..., s_n)$
4:     $s_{Collision} = s_{FrontCar} - s_{Ego}$
5:     **for** waypoint $s_i = s_1, s_2, s_3, ...$ **do**
6:         **if** $s_{Collision} > s_n$ **then**
7:             $v_i = $ Query_Reference_Speed(s_i)
8:         **else**
9:             $v_i = Brake$
10:         **end if**
11:     **end for**
12: **else**
13:     Check the availability of target lane ($a_t = -1$ is the left lane, $a_t = 1$ is the right lane)
14:     **if** Target Lane is not available **then**
15:         Return to Lane Keeping Mode
16:     **else**
17:         Get the end pose of lane changing s_{end}
18:         $(s_1, s_2, s_3, ..., s_n) = $ QuinticPlanner(s_{ego}, s_{end})
19:         **for** waypoint $s_i = s_1, s_2, s_3, ...$ **do**
20:             $v_i = $ Query_Reference_Speed(s_i)
21:         **end for**
22:     **end if**
23: **end if**
24: Return $(s_1, v_1, s_2, v_2, s_3, v_3, ...s_n, v_n)$

This speed profile is a fuel optimal control strategy compared to constant speed cruise control with the same average speed setup. Since solving optimization problems is time consuming, which may cost about 60 s from initialization status to 95% convergence, meanwhile the optimization significantly slows down as the predictive horizon increases. Hence, we choose to execute the PCC algorithm in advance to plan a global optimal speed profile, then the trajectory planner will query the speed profile according to the given coordinates. As a result, we can not only reduce the operation cost but also accelerate online planning.

6.2.2. Trajectory Planner

We have two different modes for the trajectory planning module. If there is no lane change order sent from the superior module, the lane keeping mode will be activated. Otherwise, if a lane change signal is received, the system will enter the lane change mode. In lane keeping mode, the planner queries the map for the current lane's reference center-line, then discretizes the center-line into waypoints $(s_1, s_2, s_3, ..., s_n)$, then assign the speed profile for each waypoint. In lane change mode, the planner queries the target lane in the map and determines the safety, as shown in **Figure 8**, there are three different situations: **Figure 8A** The target lane does not exist. **Figure 8B** There is a collision risk for lane changing. **Figure 8C** The target lane exists and is safe for lane changing now. If it is invalid or not safe for lane changing, the algorithm will return to the lane keeping mode. If it is suitable for lane changing, we adopt the quintic polynomial trajectory

planning algorithm (Piazzi et al., 2002) to generate a smooth trajectory from the current position to the target waypoint in the target lane. Then the planner queries the reference speed profile and assigns it to the trajectory. The trajectory planner module is summarized in **Algorithm 1**.

Finally, we obtain a sequence of waypoints along with the corresponding speed profile, $(s_1, v_1, s_2, v_2, s_3, v_3, ...s_n, v_n)$, which will be sent to the next module.

6.3. Experiments of Intelligent Decision Making

6.3.1. Numerical Experiment Scenarios

We reconstruct our test site, which is a closed highway in Jinan, a city in Eastern China, shown in **Figure 9**, and also rebuild the texture in visualization. We illustrate the ego-truck, the traffic flow (of neighbor cars), and the trajectory on the screen, as shown in **Figure 10**. The test site is 15km in length in total, covering variable typical road conditions, e.g., slope, tunnel, and curve.

6.3.2. Experiment Setup

We build our reinforcement learning based decision making model based on Pytorch (Paszke et al., 2019) and train it on a ThinkStation P920 with Intel Xeon(R) Silver 4110 2.1GHz x32 and NVIDIA RTX 2080Ti. During the training process, the learning rate is fixed as 5e−4, the optimizer is ADAM (Kingma and Ba, 2015), and the training batch size sets as 64. The proposed system is running at 10 Hz. Note that up to six cars in front of the ego-truck and two cars behind the ego-truck are observable. The model is trained for 2 days, and the reward curve is shown in **Figure 11B**.

For performance evaluation of our RL model, we present the following metrics for quantitative evaluation:

1. Delta velocity: the average difference between ego-truck speed V and reference speed V_{ref}. It is formulated as $\nabla V = (V_{ref} - V)/V_{ref}$.
2. Count of lane change: the average count of lane change decisions during the journey.

To increase the diversity of the environment, we investigate traffic with three different densities and two average speeds. We modify the traffic spawn probability as 0.05, 0.02, 0.005, corresponding to the dense, medium, sparse density, respectively. For the average speed of traffic aspect, we set the maximum speed of the traffic flow as 12.5 m/s, 15 m/s, and our ego-truck's reference speed is fixed as 16.67 m/s (60 km/h) which is suitable for most logistics operations.

For comparison, we propose three baseline algorithms with rule-based FSM to mimic different characteristic drivers' behavior. When the ego-truck detects obstacles d m ahead of it, we will check whether there is no traffic in the neighbor lane with d m ahead of it and d m back of it. If so, the lane change decision will be sent to the trajectory planner, if not, we will keep it in this lane. Here, we use d to describe the drivers' characteristics: aggressive, neutral, and conservative, corresponding to $d = $ 50, 100, and 150.

FIGURE 9 | (A) The prototype truck from our partner SINOTRUK, a 12 wheeler truck. We reconstruct our test site **(B)** (a closed highway in Jinan Eastern China) in our system for the numerical experiment **(C)**.

FIGURE 10 | We rebuild the texture in visualization, illustrate our ego-truck, traffic flow, and also the trajectory on the screen. **(A)** Car following, **(B)** running alone, **(C)** lane change, and **(D)** overtaking.
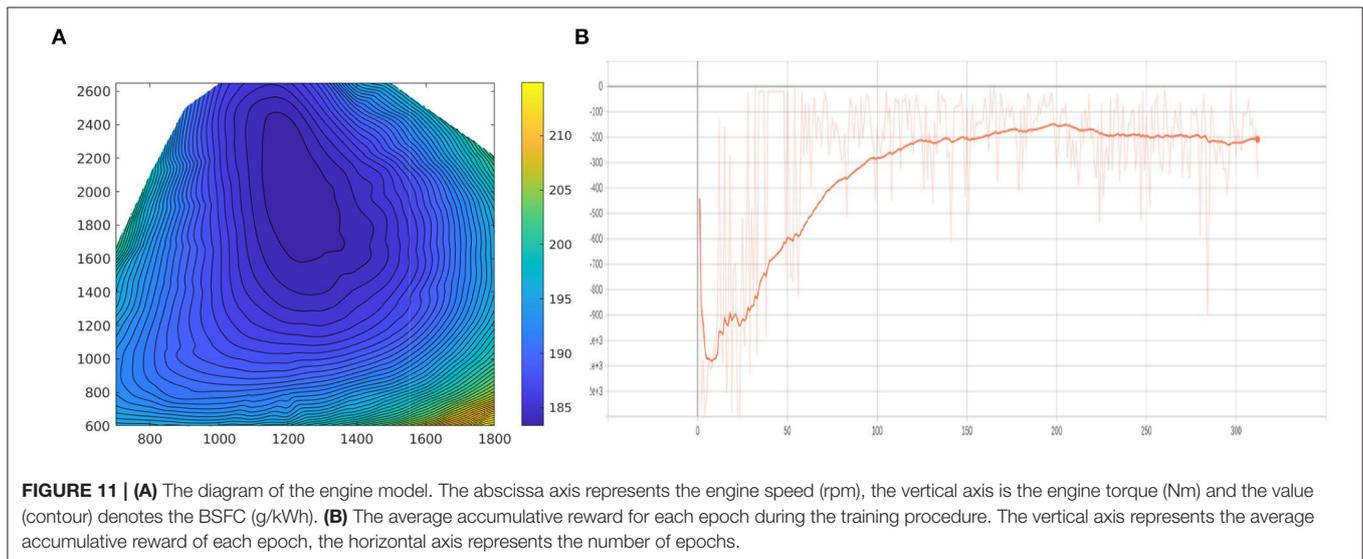

FIGURE 11 | (A) The diagram of the engine model. The abscissa axis represents the engine speed (rpm), the vertical axis is the engine torque (Nm) and the value (contour) denotes the BSFC (g/kWh). **(B)** The average accumulative reward for each epoch during the training procedure. The vertical axis represents the average accumulative reward of each epoch, the horizontal axis represents the number of epochs.

We compare our RL based lane change decision-maker with aggressive, neutral, and conservative baseline algorithms. Note that except for the decision-maker module, all other modules remain unchanged. We run each experiment trial 30 times and compute the mean value of them.
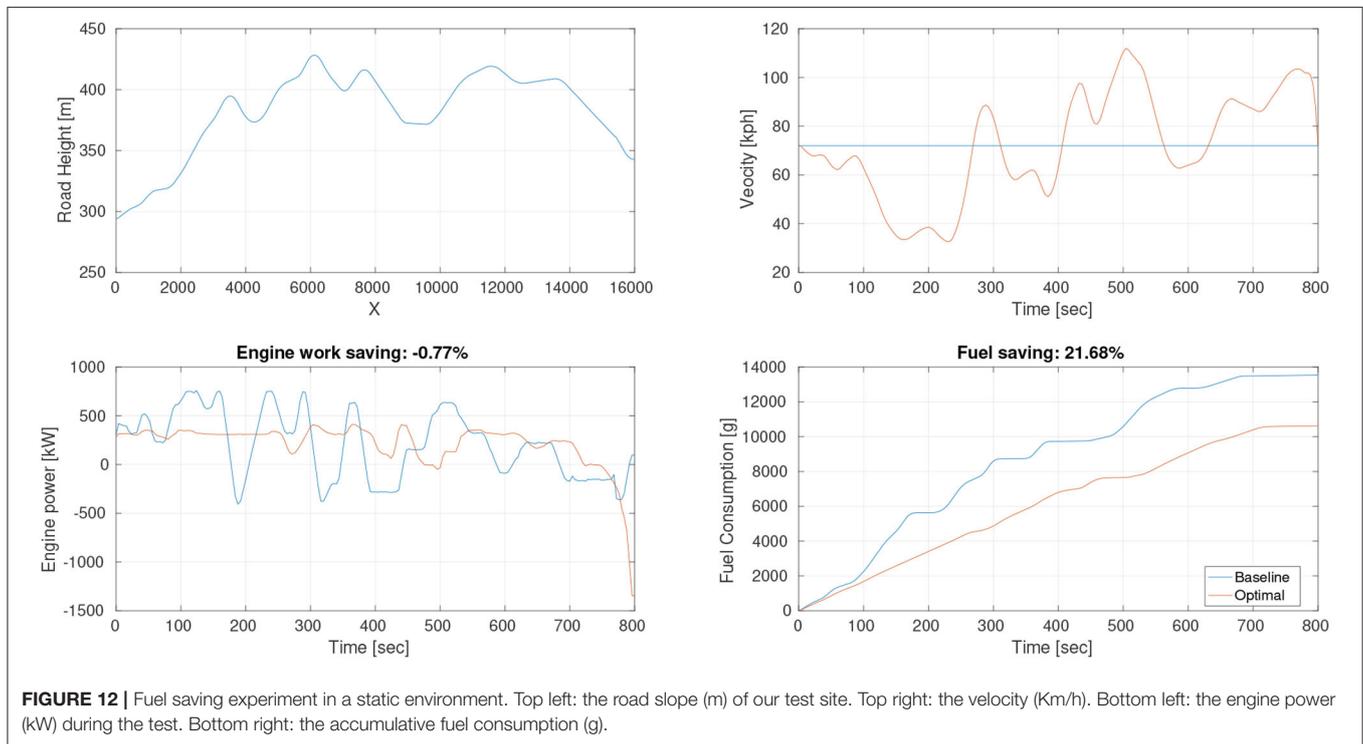
Table 1 shows the performance evaluated using different methods in different test scenarios. It can be seen that our RL base method yields the best performance according to delta velocity metrics in most cases. We noticed that our method's performance is worse than the aggressive baseline algorithm. This is because that aggressive baseline tends to change lanes frequently. Ideally, if the ego-truck can change the lane frequently enough that the influence of any car in front of it can be avoided. However, because of the character of the truck we mentioned in the introduction section, it is impossible, even dangerous, for a truck to be tap-dancing through the traffic flow. That is the reason why we add a penalty for all lane change decisions in our reward function for the RL. Our goal is to achieve the reference speed

Scenario	Trafic speed	Average delta speed				Average count of lane change			
		Conservative	Neutral	Aggressive	RL-Baseline	Conservative	Neutral	Aggressive	RL-Baseline
Dense	12.5 m/s	38.9%/0.059	28.5%/0.043	24.5%/0.038	**24.0%**/0.037	2.84/1.76	6.23/2.67	6.98/3.87	6.16/2.65
	15 m/s	30.9%/0.047	20.1%/0.031	18.8%/0.028	**17.3%**/0.027	3.06/1.82	6.89/2.85	7.13/3.91	6.24/2.68
Medium	12.5 m/s	14.1%/0.022	13.8%/0.021	6.6%/0.010	8.0%/0.012	4.58/1.23	5.09/2.37	8.93/3.39	6.22/2.67
	15 m/s	4.4%/0.007	1.6%/0.002	1.6%/0.002	**1.4%**/0.002	5.00/1.34	4.02/2.08	8.99/3.41	6.30/2.69
Sparse	12.5 m/s	5.9%/0.009	3.4%/0.005	4.4%/0.007	**3.4%**/0.005	2.55/0.68	2.83/0.75	2.99/0.80	3.02/0.81
	15 m/s	1.0%/0.001	1.0%/0.001	10.0%/0.002	**1.0%**/0.001	0.40/0.11	0.40/0.10	0.50/0.13	0.77/0.21

Note that Conservative in table is the abbreviation of the conservative baseline. Average Count of Lane Change (shown as mean/std) evaluated for different methods on different scenarios.

FIGURE 12 | Fuel saving experiment in a static environment. Top left: the road slope (m) of our test site. Top right: the velocity (Km/h). Bottom left: the engine power (kW) during the test. Bottom right: the accumulative fuel consumption (g).

with as few lane changes as possible while ensuring safety. We can observe that our RL based approach changes lanes fewer than the aggressive baseline algorithm in most cases. With the fewer change, we also achieve comparable performance compared to baseline methods.
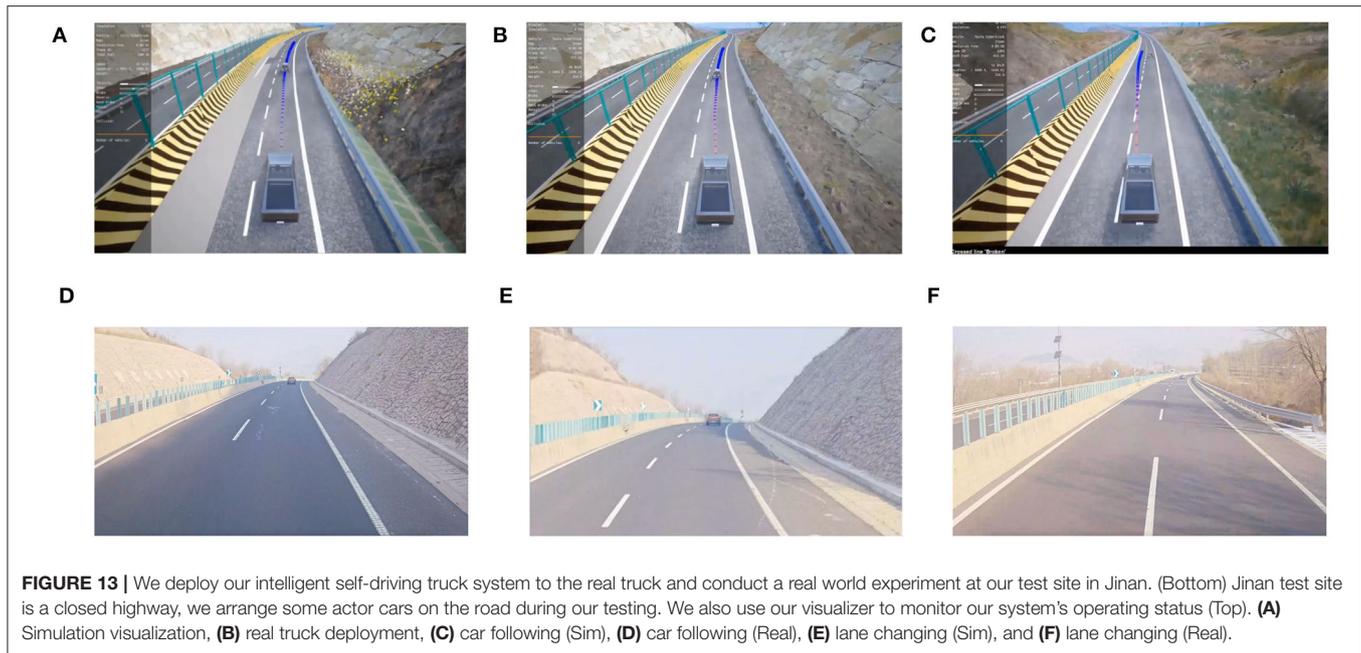
6.4. Experiment of Fuel Efficiency

We evaluate the fuel efficient speed planner described in Section 6.2.1. First, we conduct the experiment in a static environment, in which traffic is not involved. The baseline method we choose to compare with is the constant speed cruise strategy (72 km/h). As shown in **Figure 12**, the top left figure illustrates the road slope (m) of our test site, the top right figure shows the velocity (Km/h) of our proposed system (orange line) and baseline method (blue line), the bottom left figure shows the engine power (kW) during the test, the bottom right figure illustrates the accumulative fuel consumption (g) for our

method and baseline method. The results show that our proposed system can effectively allocate the engine working status, which is more stable than the baseline method. The key fuel saving capacity for heavy duty trucks on the hilly road is to utilize the conversion between kinetic energy and potential energy uphill and downhill. The velocity allocation of our proposed system is shown in the top right figure, which decelerates in the uphill process and accelerates in the downhill process. Our proposed system can save about 21.68% fuel compared to the constant speed cruise strategy.

7. REAL WORLD EXPERIMENT

We deploy our proposed intelligent autonomous truck system to the real truck and conduct several experiments at our test site in Jinan. In this section, we describe the setup detail of the real

FIGURE 13 | We deploy our intelligent self-driving truck system to the real truck and conduct a real world experiment at our test site in Jinan. (Bottom) Jinan test site is a closed highway, we arrange some actor cars on the road during our testing. We also use our visualizer to monitor our system's operating status (Top). **(A)** Simulation visualization, **(B)** real truck deployment, **(C)** car following (Sim), **(D)** car following (Real), **(E)** lane changing (Sim), and **(F)** lane changing (Real).

truck deployment first, then we illustrate several results of the real world experiment.

In order to integrate our proposed system in to the real truck, we plug our system into Inceptio autonomous driving platform to obtain the perception result and control the real truck, as shown in **Figure 1**.

We recorded a video from the front camera on our truck, and also the visualization of our system's running. As shown in **Figure 13**, our system demonstrates robust performance in the real truck experiment and the simulated environment.

CONCLUSION AND FUTURE STUDY

In this article, we describe an intelligent self-driving truck system, which consists of three main components: the realistic traffic simulation module for generating realistic traffic flow in testing scenarios, the high-fidelity truck model for mimicking real truck response in real world deployment, and the intelligent planning module with learning-based decision making algorithm and multi-mode trajectory planner. We conduct adequate evaluation experiments for each component, and the results show the robust performance of our proposed intelligent self-driving truck system. Our system is the first open-sourced full self-driving truck system for logistic operation and mass production. In addition, the high-fidelity truck model filled the gap and demand for self-driving truck development in industry and academia. Our code is available at https://github.com/InceptioResearch/IITS.

In this article, we only focus on the self-driving truck in highway transportation scenarios, covering 90% of logistic daily operation. However, there is still a long way to full L4 autonomous driving truck, especially dealing with more complex traffic and pedestrians. We noticed many recent advanced studies have been released targeting several areas in autonomous driving, e.g., collision avoidance (Zhang and Fisac, 2021), left-turn

planning (Shu et al., 2020). For the future study, we plan to cover more logistic truck operation scenarios in addition to highways, e.g., on-ramp/off-ramp scenarios, left-turn/right-turn planning in crowd intersections. We also welcome you to contribute codes and ideas to our project.

DATA AVAILABILITY STATEMENT

The codes and datasets for this study can be found in the repository IITS: https://github.com/InceptioResearch/IITS. Further inquiries can be directed to the corresponding author/s.

AUTHOR CONTRIBUTIONS

DW developed the planning module, completed truck model integration, and wrote the first draft of the manuscript. LG, ZL, WL, JR, JZ, PZha, PZho, and SW contributed to the system development and debugging. JP, DM, and RY contributed to the conception and design of the system. All authors contributed to manuscript revision, read, and approved the submitted version.

FUNDING

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fnbot. 2022.843026/full#supplementary-material

REFERENCES

AG, M. T. B. (2021). *Euro Truck Simulator 2*. Available online at: https://eurotrucksimulator2.com

Cai, P., Lee, Y., Luo, Y., and Hsu, D. (2020). "Summit: a simulator for urban driving in massive mixed traffic," in *2020 IEEE International Conference on Robotics and Automation (ICRA)* (Paris: IEEE), 4023–4029.

Chao, Q., Bi, H., Li, W., Mao, T., Wang, Z., Lin, M. C., et al. (2020). A survey on visual traffic simulation: models, evaluations, and applications in autonomous driving. *Comput. Graphics Forum* 39, 287–308. doi: 10.1111/cgf.13803

Codevilla, F., Müller, M., López, A., Koltun, V., and Dosovitskiy, A. (2018). "End-to-end driving via conditional imitation learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)* (Brisbane, QLD: IEEE), 4693–4700.

Corporation, M. S. (2021). *Trucksim-Mechanical Simulation*. Available online at: https://www.carsim.com/products/trucksim/

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). "CARLA: an open urban driving simulator," in *Conference on Robot Learning* (Mountain View, CA: PMLR), 1–16.

Fellendorf, M., and Vortisch, P. (2010). "Microscopic traffic flow simulator vissim," in *Fundamentals of Traffic Simulation* (New York, NY: Springer), 63–93.

Feng, S., Yan, X., Sun, H., Feng, Y., and Liu, H. X. (2021). Intelligent driving intelligence test for autonomous vehicles with naturalistic and adversarial environment. *Nat. Commun.* 12, 1–14. doi: 10.1038/s41467-021-21007-8

Ferguson, D., Howard, T. M., and Likhachev, M. (2008). Motion planning in urban environments. *J. Field Rob.* 25, 939–960. doi: 10.1002/rob.20265

Fortune Business Insights (2020). *Autonomous Trucks Market*. Available online at: https://www.fortunebusinessinsights.com/autonomous-truck-market-103590

Hasselt, H. V. (2010). "Double q-learning," in *Advances in Neural Information Processing Systems* (Vancouver, BC), 23.

Kesting, A., Treiber, M., and Helbing, D. (2007). General lane-changing model mobil for car-following models. *Transp. Res. Rec.* 1999, 86–94. doi: 10.3141/1999-10

Kingma, D. P., and Ba, J. (2015). "Adam: a method for stochastic optimization," in *International Conference on Learning Representations (ICLR)* (San Diego, CA). doi: 10.48550/arXiv.1412.6980

Krajzewicz, D., Erdmann, J., Behrisch, M., and Bieker, L. (2012). Recent development and applications of SUMO-Simulation of Urban MObility. *Int. J. Adv. Syst. Measure.* 5, 128–138. Available online at: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.671.2113&rep=rep1&type=pdf

Lattemann, F., Neiss, K., Terwen, S., and Connolly, T. (2004). The predictive cruise control-a system to reduce fuel consumption of heavy duty trucks. *SAE Trans.* 113, 139–146. doi: 10.4271/2004-01-2616

Leurent, E. (2018). *An Environment for Autonomous Driving Decision-Making*. Available online at: https://github.com/eleurent/highway-env

Li, W., Pan, C., Zhang, R., Ren, J., Ma, Y., Fang, J., et al. (2019). AADS: Augmented autonomous driving simulation using data-driven algorithms. *Sci. Rob.* 4, 863. doi: 10.1126/scirobotics.aaw0863

Liao-McPherson, D., Huang, M., and Kolmanovsky, I. (2018). A regularized and smoothed fischer-burmeister method for quadratic programming with applications to model predictive control. *IEEE Trans. Automat. Contr.* 64, 2937–2944. doi: 10.1109/TAC.2018.2872201

Lu, X., Shladover, S., and Hedrick, J. (2004). "Heavy-duty truck control: Short inter-vehicle distance following," in *Proceedings of the 2004 American Control Conference, Vol. 5* (Boston, MA: IEEE), 4722–4727.

Madrigal, A. C. (2017). *Inside Waymo's Secret World for Training Self-Driving Cars*. Available online at: www.theatlantic.com/technology/archive/2017/08/inside-waymos-secret-testing-and-simulation-facilities/537648/

MarsAuto (2017). *Europilot*. Available online at: https://github.com/marsauto/europilot

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature* 518, 529–533. doi: 10.1038/nature14236

NVIDIA (2017). *Drive Constellation*. Available online at: https://www.nvidia.com/en-us/self-driving-cars/simulation/

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al. (2019). "Pytorch: an imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems* (Vancouver, BC), 32.

Piazzi, A., Bianco, C. L., Bertozzi, M., Fascioli, A., and Broggi, A. (2002). Quintic g/sup 2/-splines for the iterative steering of vision-based autonomous

vehicles. *IEEE Trans. Intell. Transportat. Syst.* 3, 27–36. doi: 10.1109/6979.994793

SAE, J. (1939). *Recommended Practice for a Serial Control and Communications Vehicle Network*. SAE J1939 Standards Collection.

Sallab, A. E., Abdou, M., Perot, E., and Yogamani, S. (2017). Deep reinforcement learning framework for autonomous driving. *Electron. Imaging* 2017, 70–76. doi: 10.2352/ISSN.2470-1173.2017.19.AVM-023

Schwarting, W., Alonso-Mora, J., and Rus, D. (2018). Planning and decision-making for autonomous vehicles. *Ann. Rev. Control Rob. Auton. Syst.* 1, 187–210. doi: 10.1146/annurev-control-060117-105157

Sewall, J., Wilkie, D., and Lin, M. C. (2011). "Interactive hybrid simulation of large-scale traffic," in *Proceedings of the 2011 SIGGRAPH Asia Conference* (Hong Kong), 1–12.

Shah, S., Dey, D., Lovett, C., and Kapoor, A. (2018). "Airsim: high-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics* (Cham: Springer), 621–635.

Shu, K., Yu, H., Chen, X., Chen, L., Wang, Q., Li, L., et al. (2020). "Autonomous driving at intersections: a critical-turning-point approach for left turns," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)* (Rhodes: IEEE), 1–6.

Treiber, M., and Helbing, D. (2001). Microsimulations of freeway traffic including control measures. *at - Automatisierungstechnik* 49, 478–478. doi: 10.1524/auto.2001.49.11.478

Ulbrich, S., and Maurer, M. (2013). "Probabilistic online pomdp decision making for lane changes in fully automated driving," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)* (The Hague: IEEE), 2063–2067.

Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M., et al. (2008). Autonomous driving in urban environments: boss and the urban challenge. *J. Field Rob.* 25, 425–466. doi: 10.1002/rob.20255

Van Hasselt, H., Guez, A., and Silver, D. (2016). "Deep reinforcement learning with double q-learning." *Proceedings of the AAAI Conference on Artificial Intelligence* (Phoenix), 30.

Viscelli, S. (2018). Driverless? autonomous trucks and the future of the american trucker.

Wang, J., Zhang, Q., Zhao, D., and Chen, Y. (2019). "Lane change decision-making through deep reinforcement learning with rule-based constraints," in *2019 International Joint Conference on Neural Networks (IJCNN)* (Budapest: IEEE), 1–6.

Wang, P., Chan, C.-Y., and de La Fortelle, A. (2018). "A reinforcement learning based approach for automated lane change maneuvers," in *2018 IEEE Intelligent Vehicles Symposium (IV)* (Changshu,: IEEE), 1379–1384.

Yan, X., Feng, S., Sun, H., and Liu, H. X. (2021). Distributionally consistent simulation of naturalistic driving environment for autonomous vehicle testing. *arXiv[Preprint]*. arXiv:2101.02828. doi: 10.48550/arXiv.2101.02828

Zhang, Z. and Fisac, J. F. (2021). "Safe occlusion-aware autonomous driving via game-theoretic active perception." in *Proceedings of Robotics: Science and Systems (Virtual)*.