Extending SUMO for Lane-Free Microscopic Simulation of Connected and Automated Vehicles

Dimitrios Troullinos, Georgios Chalkiadakis, Diamantis Manolis, Ioannis Papamichail, and Markos Papageorgiou

Technical University of Crete, Chania, Greece
{dtroullinos, dmanolis, ipapa, markos}@dssl.tuc.gr
gehalk@intelligence.tuc.gr

Abstract

This paper presents some aspects of TrafficFluid-Sim, a lane-free microscopic simulator aimed for vehicle-to-vehicle (V2V) and/or vehicle-to-infrastructure (V2I) connected and automated vehicles. The simulator builds on and extends the SUMO simulation infrastructure to model traffic environments featuring two novel vehicle characteristics: (i) vehicles can be located at any arbitrary lateral position within the road boundaries; and (ii) vehicles may exert, based on their automated driving and connectivity capabilities, "vehicle nudging" to other surrounding vehicles. Special care is taken during the development so that no inter-tool "communication" delays, which could affect efficiency and user experience, are introduced. As such, we provide the user with a dynamic library for lane-free vehicle movement control, the incorporation of which allows one to avoid all commutation cost between SUMO and an external application. Various components are involved in the development process, including the implementation of lane-free vehicle movement; how the simulator handles external traffic and appropriately inserts vehicles at entry points under the lane-free paradigm; the development of an Application Programming Interface (API) for information retrieval and vehicle control in lane-free environments; and, finally, the capability to utilize the bicycle kinematic model additionally to the double-integrator model as a more realistic model of vehicle movement dynamics.

1 Introduction

The arrival of Connected and Automated Vehicles (CAVs) [4] promises the dawn of a new era of road transportation, that is linked to the emergence of novel, safer, and more efficient than existent traffic flow paradigms [15, 6]. Even though CAVs are still fledging, the need to enhance existing [10, 1, 2] or develop entirely new traffic simulators that will allow their design and study at the traffic level is obvious and well recognized in the traffic research community. This is due to the fact that certain key aspects, and characteristics of the prevalent pre-CAV traffic are linked to assumptions or constraints that are likely to become obsolete; while novel features, enabled by the superb CAV capabilities, are emerging and need to be captured in traffic simulation.

In particular, parallel road lanes emerged as a safety-fostering measure at the mid-20th century, when the density of automobiles increased strongly in road transportation. Traffic lanes simplify the driving operations and undoubtedly improve on safety and average driving speed, yet they introduce some disadvantages. A considerable proportion of traffic accidents (10%) occur due to lane changes [17], an operation that all drivers perform on a regular basis, either for overtaking or to follow a specific route. Another important limitation is that lanes have a fixed width according to a presumed maximum vehicle width, which results in lower lateral road occupancy, and hence lower road capacity. Such compromises are unavoidable with

human drivers, given their high reaction times and limited visual capabilities. However, in traffic environments with high CAV penetration rates, such limitations will soon be strongly mitigated, as CAVs have superb sensing capabilities, allowing continuous and reliable monitoring of their environment on a 360° basis; as well as efficient computer-based decision-making. Thus, in recent years, there is a drive to conceive and investigate novel traffic paradigms—e.g., notably, the *TrafficFluid* [14] concept— which entails the need for developing corresponding new traffic simulators to accommodate the novel characteristics of these paradigms.

2 The TrafficFluid Concept

TrafficFluid [14] is a novel paradigm for vehicular traffic, which is being investigated within its namesake ERC Advance Grant, and targets fully automated vehicles (at SAE Levels 4 or 5) [13], equipped with vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication systems. *TrafficFluid* is based on two key principles:

1. *Lane-Free Traffic*: Vehicles can be in any lateral position within the road boundaries, thus lifting the lane changing task.

2. *Nudging*: Vehicles can exert a "nudging" effect, i.e., apply a pushing force, on front vehicles. This concept implies that slower downstream vehicles may be nudged to move aside so as to facilitate faster upstream vehicles to pass. More importantly, at a macroscopic level, nudging has a capacity increasing and stabilizing effect on traffic flow [14, 8].

Results obtained so far in *TrafficFluid* (e.g., in [14, 23, 26, 9]) showcase aspects of the concept's potential. In order to further examine emerging approaches and, in particular, implications at the traffic level, there is an imminent need for a traffic simulator that incorporates these core principles, while retaining a flexible approach. The simulator's structure has to be modular, so as to integrate and evaluate approaches originating from many different research fields, ranging from Control Theory [3] to Artificial Intelligence [19], Machine Learning [12, 22, 5], and Multi-Agent Systems [24].

In what follows, we present in some detail our design and implementation choices for the simulator. Among other things, we explain how lane-free vehicle movement is achieved; how the simulator handles external traffic and appropriately inserts vehicles at entry points, offering the user two alternative functionalities: (a) the option to laterally position the vehicles in such a way that the entire capacity of the road is exploited; or (b) the option to sample each vehicle's desired speed from a pre-defined distribution, and then assign a lateral position for the vehicle that is proportional to its respective desired speed. In addition, we discuss the features of the Application Programming Interface (API) for information retrieval and vehicle control under the lane-free paradigm. Furthermore, we address the possibility to model the vehicle movement dynamics, either with a bicycle kinematic model or with a simpler double-integrator model. We also present envisaged extensions of the simulator, most of which actually correspond to work currently in progress. We note that a (previous) version of our simulator has already been used for obtaining the results of [23], and more recently for the results in [26].

3 Simulator Overview

The purpose of this work is to build a simulator that enables various control strategies for lane-free vehicle movement, for CAVs, including the nudging effect. We choose SUMO [10]

(Simulation of Urban MObility) as the traffic platform to build upon, since, as an open-source project, it can easily be adjusted to our needs.

Due to the popularity of SUMO, multiple frameworks (e.g. VEINS, iCS, VsimRTI) exist that can connect with SUMO via its traffic control interface (TraCI), in order to provide additional functionalities, such as enhanced vehicle communication (see Section 9 for more details). Our primary endeavour is to create a *modular approach* regarding the road infrastructure characteristics (including space-time varying internal boundary separating the two traffic directions of a road, see [11]) and the vehicles and their movement design (physical dimensions, controllers, connectivity with surrounding CAVs and infrastructure, desired speeds, etc.), thus providing a contained and flexible environment to systematically evaluate novel approaches at various levels (vehicle movement, infrastructure, traffic control).

TraCI enables communication with SUMO, giving access to information regarding the simulation environment, and managing many aspects of it online. However, TraCI imposes certain time limitations, which escalate as the number of controllable vehicles rises. This motivated us to directly extend the code-base of SUMO, rather than relying on TraCI. While SUMO does not support connected vehicles or vehicles with custom controllers, we implement such functionalities by "overriding" its default behavior. Therefore, our contribution can be viewed as an additional plug-in to the existing SUMO implementation.

We should mention here that, besides TraCI, one alternative option would be to consider Libsumo, a low-level Application Programming Interface (API) that does not exhibit the computational limitations imposed by socket communication. However, this choice was deemed to be rather restrictive, and thus inappropriate for our requirements. Note that Libsumo does not support the GUI application, meaning that it works only with the command line version of SUMO. In addition, the functionalities of the APIs (TraCI or Libsumo) would be limiting for some of the extensions we provide; or would require additional computational tasks to monitor or control the traffic environment properly. By working directly with the open-source code-base of SUMO, we can meticulously exploit the existing infrastructure and extend it appropriately.

Thus, we retain the current functionality and develop complementary components that can both monitor online a lane-free traffic environment, and also control vehicles considering the lane-free paradigm. This is achieved via an additional *dynamic library* we implemented, which allows development in C/C++.[1] In more detail, the library provides the user with the ability to follow a core coding structure, which contains an *initialization* (and *finalization*) function. These functions, as the naming suggests, are executed once at the initialization (respectively, the termination) of the traffic environment. Also, a function that is executed in every simulation time-step is introduced. There, a user can develop code in order to test lane-free vehicle controllers, emulating vehicle connectivity aspects via retrieving of information from SUMO, since we provide a concrete and easy to use API, enabling the direct communication with SUMO, without any time-related overhead (that TraCI imposes). Controllers not necessarily tied to TrafficFluid can be also incorporated, such as [7], which tackles lane-free environments and shares similarities to the TrafficFluid concept [14]. However, SUMO, and by extension TrafficFluid-Sim, is a discrete-time simulator, therefore controllers with continuous-time models should be transformed to quasi-equivalent discrete-time controllers beforehand. Figure 1 shows an overview of the components that form the TrafficFluid-Sim tool.

By means of this library, a user can utilise the provided API in order to obtain information regarding:[2] the vehicles' properties (type, length, width, etc.) or current status (position, speed, etc.); properties of a specific road segment (IDs of vehicles operating in this segment, length,

[1] .dll file (or .so file in Linux)

[2] Here we simply highlight the features, as they will be presented in more detail in the following sections.

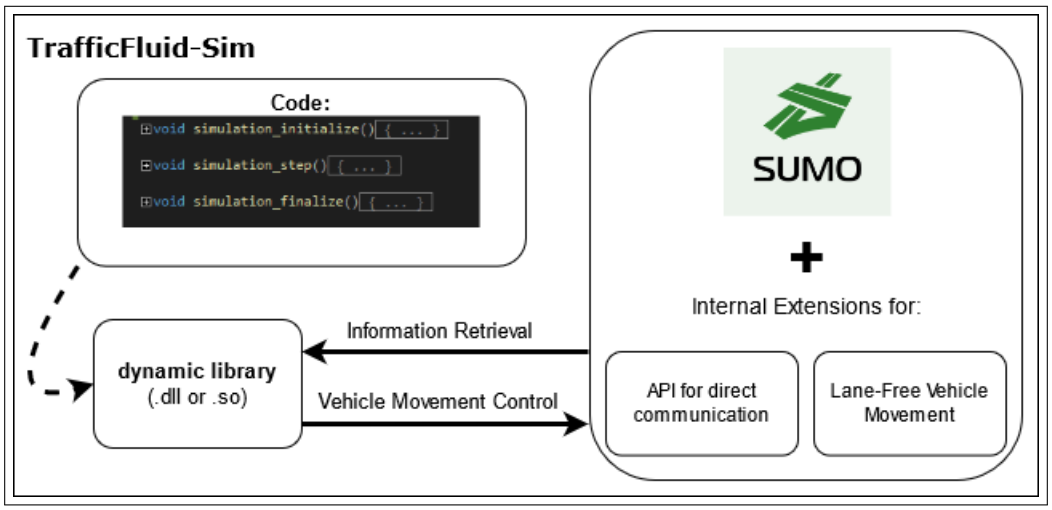

Figure 1: An overview of the components that constitute our lane-free microscopic simulation tool.

width, additional information for given regions of the road); and online information from loop detectors. Additionally, one can control the vehicles through their unique ID with the API, where, depending on the selected vehicle model, either longitudinal and lateral accelerations or steering angle and forward acceleration are supplied as control inputs to the relevant function.

Figure 2 shows a snapshot of the application for a lane-free traffic scenario with an on-ramp. It is evident that the vehicles operate with no restrictions whatsoever regarding lateral placement within the road boundaries. Lane-markings only exist to indicate the acceleration lane for the vehicles to merge on the lane-free mainstream.

In the following sections, we outline key features and functionalities offered; and present the different aspects of the implementation in more detail.

4 Lane-Free Vehicle Movement

In this section we first discuss how SUMO handles longitudinal and lateral dynamics for lane-based applications and how to adapt them appropriately for lane-free settings. Then, we present the two alternative options for lane-free vehicle movement, namely the double-integrator model and the bicycle kinematic model.

4.1 Longitudinal and Lateral Dynamics in SUMO

Networks in SUMO are composed by multiple connected road segments (edges), essentially depicted by a graph representation, and SUMO maintains local coordinates for each vehicle, with respect to the road segment the vehicle is residing. For the longitudinal coordinate, the value is essentially reflecting the distance from the respective segment's starting point. In a lane-free environment, this value is suitable, hence no substantial changes are needed. However, this is not the case for the lateral coordinate, where the value we can obtain from SUMO (besides information about the lane we are currently in), is the lateral distance from the respective lane's center. Moreover, SUMO does not actually consider lateral dynamics, since it does not

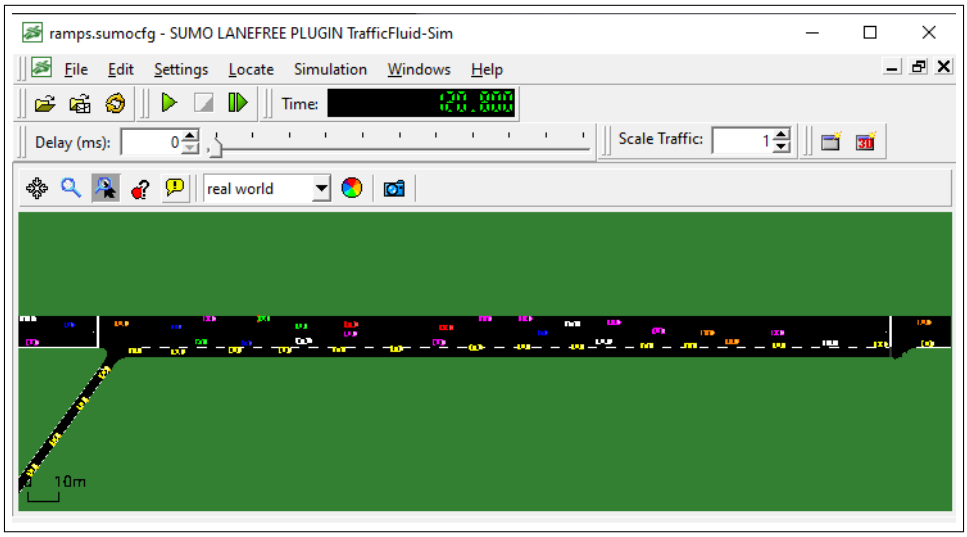

Figure 2: A snapshot of a lane-free traffic scenario with an on-ramp.

involve anywhere lateral speed or acceleration. One can find information about lateral speeds only within the Sublane model[3] of SUMO, but this is only relevant to the lane-changing model, whereby the vehicles simply move by providing a new lateral distance deviation value. The adjustment of the Sublane model's behavior and of its safety precautions is essentially tied with a lane-based environment, which led to the decision to directly interfere within the code and provide an independent module that explicitly handles lateral dynamics.

More specifically, Sublane is not adequate for our purposes, as, in order to model lane-free movement, we need to consider both *longitudinal* and *lateral* dynamics with constant respective accelerations for each simulation time-step. SUMO supports this only for the longitudinal motion of vehicles, through the 'Ballistic-Update' option of SUMO. Thus, we added information about the lateral speed of the vehicles "internally" (i.e., by altering the existing SUMO simulator code), and modified the update process of the vehicles' lateral position.

Regarding the positioning of the vehicles, the *local* coordinates (x, y) of the rectangular-shaped vehicles that we consider are as follows: The longitudinal position x of the vehicle is now the distance of its center point from the starting point of the road segment. Accordingly, the lateral position y measures the distance from the right road boundary to the vehicles' center point. Hence, the vehicles observe a single lateral position, without knowledge about lateral placement w.r.t. to lane centers, as this information is not meaningful anymore for designing a vehicle movement control strategy under lane-free settings. Although TrafficFluid-Sim addresses lane-free environments, it could also potentially simulate lane-based traffic with continuous lateral dynamics (as the Sublane model), provided that the associated vehicle controller is designed accordingly, i.e., conforming to virtual lane placements.

4.2 Double-Integrator Model

The 'Ballistic-Update' option of SUMO for the longitudinal movement, and the incorporation of lateral dynamics in a similar manner, constitute a double-integrator model for the vehicle

[3]Sublane depicts more realistic lane-changing behavior.

movement dynamics. The orientation of the vehicle is always in parallel with the road for the double-integrator. This approximation is good enough for highways, where we typically consider vehicles moving with high longitudinal speeds. In the next Section 4.3, we present an alternative approach, namely the bicycle kinematic model, which provides a more realistic depiction of movement dynamics, since it incorporates the orientation of the vehicle as well.

The associated controller for vehicles employing the double-integrator model should provide a longitudinal and lateral acceleration (in m/s^2) value in every time-step. Now, under the lane-free paradigm, where vehicles can be placed anywhere laterally within the road boundaries, a collision between two vehicles is reported when their rectangular forms overlap. This is a straightforward check, given the vehicles' information regarding positioning and their dimensions (length, width), and provided that their orientation is always in parallel with the road.

4.3 Bicycle Kinematic Model

An important development, necessary to enable microscopic simulation of approaches such as [9], is the incorporation of the *bicycle kinematic model* [16] into the simulator. In this model, the vehicle's front (and back) wheels are abstracted to a unique front (and back) wheel located at the respective middle point, where only the front wheel is steerable, controlling the vehicle's orientation. We refer the interested reader to the relevant paper [16] for more details regarding this kinematic model. This model is generally more accurate in describing vehicle movements than the double-integrator model and is particularly interesting in a lane-free environment, when vehicles are driving in curves at relatively low speeds, such as in urban networks and roundabouts. In such cases, the user has the option to select an alternative method for vehicle movement dynamics in lane-free traffic. The bicycle model's state variables are, beyond the longitudinal and lateral vehicle position (x, y), the vehicle orientation θ and the forward speed v; while the control variables are the forward acceleration and the steering angle. Figure 3 illustrates the state variables of the bicycle model, where we do not have two separate dynamics for longitudinal and lateral movements, but rather a unified and more realistic movement, depicting the actual orientation resulting from steering the vehicle.

In terms of technical developments for the simulation, this involved an internal implementation regarding the position (x, y) and speed v update process of the vehicles, according to the bicycle kinematic model. Information regarding the orientation θ of the vehicle and the ability to directly change it is already available through SUMO, so we can adjust its value through internal extensions within the source code. The local coordinates (x, y) of the vehicle again refer to its center, as discussed earlier, and as depicted in Figure 3. However, the bicycle model utilizes the position corresponding to (x_b, y_b) in Figure 3, i.e., vehicles' orientation changes with respect to this point. This is considered within our implementation, and this point is also directly available to the user who wishes to design a movement strategy using the bicycle model. User access to the orientation of the vehicles is granted through the API, providing either global (with respect to the global coordinate system of SUMO) or local (with respect to the longitudinal axis of the current road vehicles reside) coordinates. The associated lane-free controller should simply provide the value for the forward acceleration F (in m/s^2) and the steering angle δ (in rad) as control inputs for every time-step (instead of the longitudinal and lateral accelerations).

Again, the local coordinate system of SUMO is utilized for the bicycle model, and as such, controllers do not need to take into account the geometry of the residing road. Hence, the operating orientation of the vehicles is local, with respect to the residing road, so turning oper-

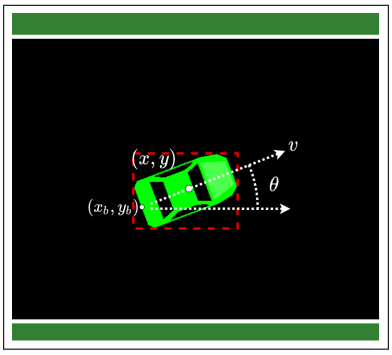

Figure 3: Illustration of a vehicle utilizing the bicycle model.

ations (e.g., at junction points) are handled through SUMO. However, for certain applications, we may be interested in controlling this operation instead of letting SUMO automatically handle such procedures. We also provide the option to do so, in a way that does not affect or cause regression to existing functionalities. One can utilize the bicycle kinematics along with global coordinate control for more realistic behavior, that does not succumb to the road structure, but is rather based on the global Cartesian coordinates (x, y). This feature serves to facilitate an impending application that involves lane-free movement in a roundabout, where we wish to work with junctions for vehicles entering and exiting, and utilize polar coordinates instead of Cartesian ones for vehicles operating within the roundabout. This provides a much more realistic depiction of the actual traffic behavior compared to the use of the standard local coordinate system SUMO provides, of course with the expected trade-off being that the we need to properly design the above-mentioned behaviors.

For this feature, the use of internal mapping from global to local coordinates (and vice-versa) was crucial, given that the existing infrastructure and our extensions rely on the local coordinate system that SUMO provides. If we completely neglected the local coordinates in favor of simplicity in the development process, then certain features would not function properly, e.g., the observation capabilities on surrounding vehicles (see Section 6.1), and as such, essential information for the design of vehicle movement strategies would not be available. Global coordinate control is tied only with the bicycle kinematics, since it would be quite restrictive for the double-integrator model, due to the absence of orientation control.

As of now, we have a simplified way that incorporates the orientation of the vehicles for the collision check. Essentially, we draw a rectangle that is in parallel to the road and contains the vehicle (red rectangular in Figure 3), and report a collision if these rectangular regions of two vehicles overlap. Of course, this procedure may report a collision even if two vehicles do not actually collide. However, it also guarantees that no collision will be disregarded.

5 Handling of Demand

In contrast to real traffic networks, any simulated network infrastructure has upstream boundaries, at which vehicles need to be inserted, given a corresponding pre-specified demand rate. This is a delicate operation, because the simulated traffic conditions (e.g. dense traffic or congestion) may not allow for the whole specified demand to enter, as this might lead to unrealistically high vehicle densities just after insertion. Thus, vehicles should only be inserted if there is enough space to the vehicle in front; and with an insertion speed that would not lead

to immediate strong deceleration. Vehicles that cannot be inserted, due to such restrictions, are placed in a virtual queue to be considered for insertion later on. On the other hand, the vehicle insertion method should not be too conservative, as this might lead to the creation of virtual queues even in cases where the demand is lower than the road capacity.

In a regular lane-based scenario, one normally specifies a demand of vehicles either for individual lanes, or for the whole edge (road segment). For the latter option, SUMO automatically distributes the demand to each lane. Contrary to a lane-based environment, where a simple time-gap or space-gap method can be applied, such a method is not appropriate for lane-free environments, since it would not exploit laterally the actual capacity of the road, which constitutes one of the intrinsic properties when operating under the lane-free paradigm. Recall that the space-gap (s_g) is the distance from the front bumper of the vehicle attempting to enter, to the rear bumper of the front one, and the time-gap (t_g) is defined as $t_g = s_g/v_p$, where v_p is the insertion speed of the vehicle attempting to enter. Thus, if we have any two quantities, we can calculate the remainder one using the equation above.

For the reasons mentioned, we designed a new method that determines where (laterally) a vehicle shall be inserted in the road; as well as whether it should actually be inserted or stay in a virtual queue of vehicles, in case of lack of sufficient space due to other vehicles in front. In the latter case, the vehicle has to wait for the next time-step before re-attempting insertion. To this end, we need to establish how the vehicles will spawn at an entry point, when defining a global cross-section, in a way that is organically tied to a lane-free environment. Essentially, we use a constant time-gap method (provided by the user) customised for this setting, which must also account for the width of the vehicle when entering. As such, each vehicle attempting to enter (according to the demand) has a set of available lateral regions that it can spawn.

Initially, the vehicle considers a single lateral region that usually corresponds to the whole road width. Of course, in case of vehicles with preferences to lateral placements on the road, the initial lateral region can be smaller, e.g., the right hand side of the road for slow vehicles such as trucks. Then, its insertion speed will be the minimum of the user-defined departure speed and the average speed of the front $n = 5$ vehicles. Of course, this choice for n may be too small for large roads, and in the future it will be generalized so that its value depends on the width of the road. This bounding serves to prevent collisions in case the region is congested, where the vehicles move with lower speed (or even in extreme cases they cannot move at all). Given the insertion speed and the pre-specified time-gap, the corresponding required space-gap in front of the vehicle is calculated, and the next step is to obtain a lateral position to spawn according to the downstream traffic.

To achieve that, we first scan for this initial lateral region (in ascending order, according to longitudinal position) the front vehicles until one is found that conforms to the calculated space-gap. Then, for all front vehicles that have a space-gap less that the required value, we restrict, according to their widths and lateral positions on the road, the available lateral regions. Finally, the vehicle randomly spawns in one of the remaining lateral regions, which guarantees that there is no vehicle in front with less than the required space-gap. Figure 4 illustrates the aforementioned procedure. Of course, if, at any point in the above procedure, the set of available lateral regions becomes empty (i.e., there is no lateral position to insert the vehicle with an acceptable space-gap in front of it), then the procedure is stopped, and the vehicle does not spawn, but is preserved in the virtual queue of SUMO, and has a chance to enter in the next time-step, when front vehicles will have advanced. It should be noted that in the aforementioned process, we impose a minimum lateral distance between the vehicles, that is selected by the user. In Figures 5, 6, we illustrate instances of lateral vehicle placement for two different respective demands, according to the aforementioned insertion method.

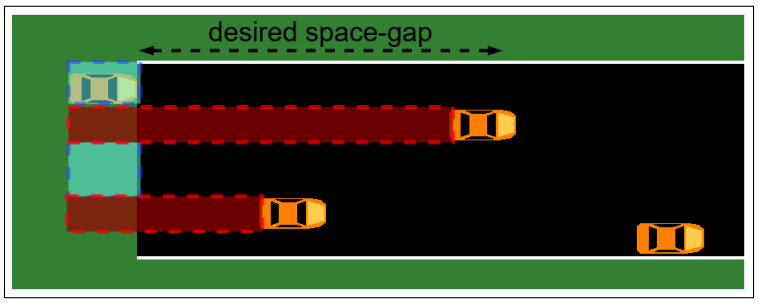

Figure 4: Vehicle is inserted randomly within the range of one of the available lateral regions (indicated with light blue). The red zones are forbidden for the entering vehicle due to other vehicles in front.

Simulated vehicles possess a desired speed that they strive to achieve when and where the traffic conditions allow. This desired speed is decided at insertion point, and accompanies the vehicles, until their exit from the network. The user may select desired speeds for the entering vehicles via the API, by providing a range of desired speeds $[v_{d,low}, v_{d,high}]$, sampled according to a specified distribution (e.g. uniform). The desired speed distribution may be independent of the lateral vehicle insertion positions, but a specific consideration for the selection of desired speeds may be to establish a mapping with the range of initial lateral placements. Specifically, some vehicle movement strategies favor a lateral vehicle position, while driving, that is proportional to the vehicle's desired speed, i.e., vehicles with higher desired speed tend to drive further left on the road, compared to vehicles with lower desired speed (e.g. trucks). In such cases, it is reasonable to relate accordingly the lateral insertion position of each vehicle with its assigned desired speed at the entry points, so as to avoid unrealistic weaving maneuvers after insertion. For instance, each desired speed v_d can be assigned based on a linear mapping from the lateral position range to the range $[v_{d,low}, v_{d,high}]$ of desired speeds (accounting also for the width of the vehicle, e.g., when the vehicle has width w_v, and the road has width w_r, the vehicle considers the lateral region $[w_v/2, w_r - w_v/2]$, so if it spawns at $w_v/2$, then the selected desired speed will be $v_{d,low}$).

The aforementioned insertion method aims to fully exploit the available lateral space. However, notice that this method, along with the mentioned linear mapping will not result in a uniform (or other desired) distribution of desired speeds. Therefore, for specific simulation scenarios requiring a specific probability distribution, we also developed an alternative insertion

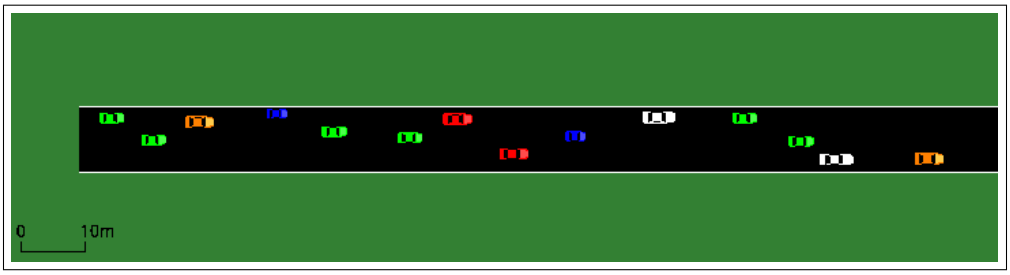

Figure 5: Vehicle placement for a demand of $7,500\,veh/h$, time-gap of $0.5\,s$, minimum lateral distance of $0.25\,m$.

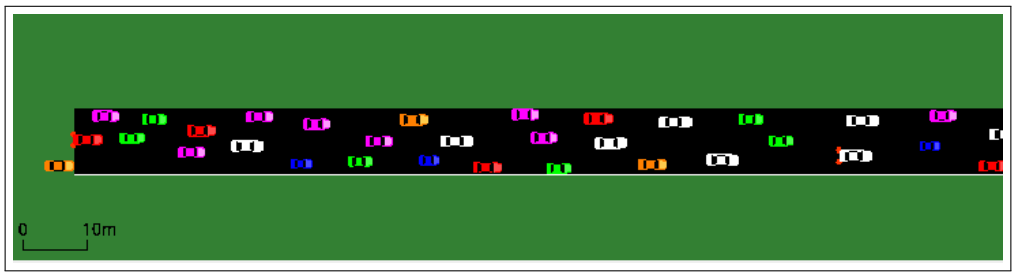

Figure 6: Vehicle placement for a demand of $12,000\,veh/h$, time-gap of $0.5\,s$, minimum lateral distance of $0.25\,m$.

method, in which we refrain from exploiting fully the lateral space.

In the alternative insertion method, we initiate the process of introducing a new vehicle to an entry point by drawing a sample v_d from the user-chosen distribution (provided by the API), within the range $[v_{d,low}, v_{d,high}]$. This sample value v_d will be the desired speed of the vehicle. Thereafter, the method decides upon the lateral placement for the vehicle. This is done via a linear mapping from the range $[v_{d,low}, v_{d,high}]$ to the range of available lateral positions $[w_v/2, w_r - w_v/2]$, according to the sampled v_d value. Next, the first vehicle in front of the vehicle to be inserted is identified, and a candidate insertion speed v_p is calculated according to the time-gap provided by the user and the space-gap between the two vehicles. Note that we consider that a vehicle is in front, if the lateral regions resulting from their lateral placements and width overlap. This is also noticeable in Figure 7, where we show visually the above procedure. The vehicle will be inserted with insertion speed $v_{insert} = min\{v_p, v_d\}$, unless $v_{insert} < min\{v_d, v_f\}$, where v_f is the speed of the vehicle in front. In that case, the vehicle will remain in the virtual queue and attempt reentering with the same desired speed (and, consequently, the same lateral position) in the next time-step.

In what follows, we showcase its use (assuming uniform distribution for the desired speeds), against that of the first method in a scenario with an extremely high value of demand, specifically, one of $18,000\,veh/h$. This is a demand clearly beyond the capacity of any lane-based

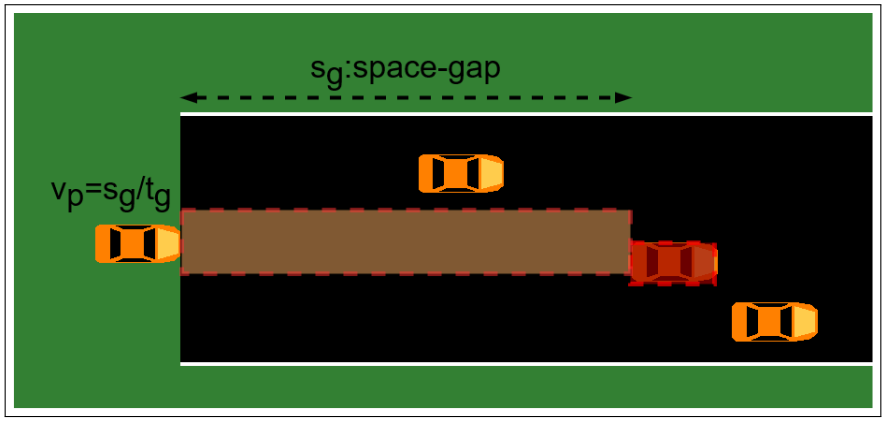

Figure 7: Vehicle has a predefined initial lateral position according to its desired speed; and it obtains an insertion speed according to the space-gap from the vehicle in the front.

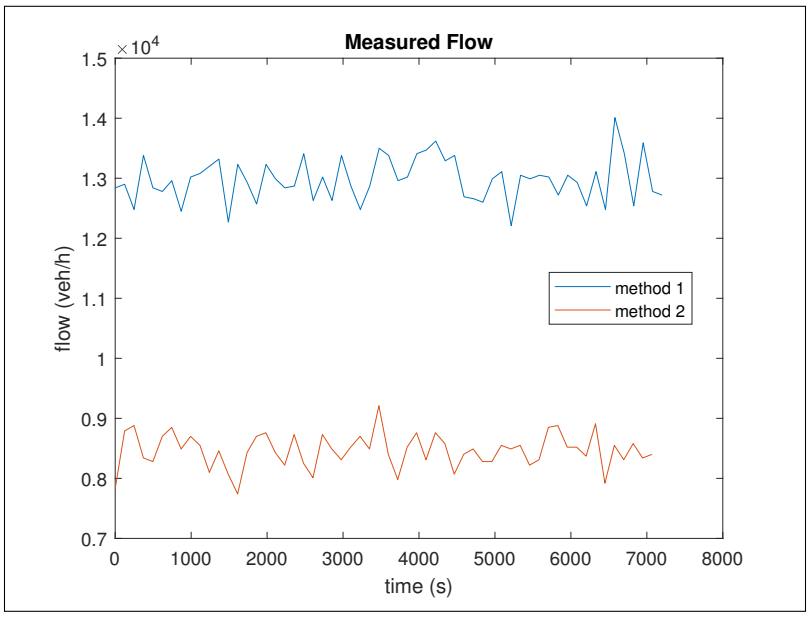

Figure 8: Measured flows of inserted vehicles from a detector placed 100 meters downstream of the entry point.

approach for a road width as the one in Table 1, corresponding to a conventional road comprising 3 lanes. We compare the two insertion methods in terms of the flow values achieved in each case.

We collect measurements of the flow with a sampling period of $T = 2\,min$. All relevant parameters are tuned according to Table 1, where $d_{lat,min}$ is the minimum lateral distance the vehicles are allowed to have upon entering; l_v, w_v are the length and width of the vehicles, respectively (which are equal for all vehicles in this scenario, for simplicity); and w_r is the width of the road.

As observed in Figure 8, where we display measured flows for the two different insertion methods, the first one (indicated as method 1 in the figure) yields an average flow of $12,998\,veh/h$. This effectively sets an upper bound for the achievable maximum flow, given the parameter choices of Table 1. The vehicle movement strategy used to obtain these results is an ad-hoc strategy presented in [14]. On the other hand, the alternative method (indicated as method 2 in the figure), using a uniform distribution $\mathcal{U}(25, 35)$, leads to under-utilisation of the road capacity. We now have vehicles spawning with desired speeds according to a uniform distribution within the range of $[25, 35]\,m/s$, as desired, but this results in a lower average flow of $8,312\,veh/h$. Thus, maintaining such a distribution comes with a "cost" of reducing by 36% the average flow inserted (w.r.t. the one achieved by method 1). These results indicate the

Table 1: Parameter choices for the demand examined.

time-gap	$d_{lat,min}$	l_v	w_v	w_r
$0.5s$	$0.25m$	$3.2m$	$1.6m$	$10.2m$

expected trade-off between demand insertion at the capacity of the road versus guaranteeing that the desired distribution of desired speeds will be exactly mapped with the lateral insertion position of entering vehicles.

6 Using a Dynamic Library for Lane-Free Vehicle Control

Normally, vehicle strategies in lane-based SUMO are structured according to two types of models, a "Car-Following" model, and a "Lane-Changing" one, that regulate the longitudinal speed and the lane-placement accordingly. This modelling structure is obviously not appropriate for lane-free vehicle controllers. Also, something crucial, that we need to address, is the fact that these models are pre-compiled and constitute a part of the SUMO environment. This means that the users can select one of the available models, and 'tune' them via model-related parameters. There is no flexibility to easily incorporate different controller designs to this environment, unless we do so by accessing the open-source code-base of SUMO (or via TraCI).

Especially for a novel traffic paradigm like *TrafficFluid*, a more flexible design, that allows for diverse vehicle control approaches and refined control of the vehicles, is fundamental. Therefore, we have made the necessary developments to make use of an external dynamic library (that is linked with the main application at execution time) in order to enable the implementation of any type of vehicle controller. such controllers receive information about the current state of the vehicle and its environment; and respond with the appropriate control input (longitudinal and lateral acceleration values or forward acceleration and steering angle) for each simulation time-step. Importantly, this essentially means that *any* vehicle strategy can be incorporated without the need to re-compile SUMO; moreover, after its compilation, the library is essentially treated for all purposes as an integral part of the application, thus avoiding all communication costs between SUMO and external frameworks (that would have been otherwise unavoidable). In conclusion, we have the benefit of flexibility, but without the cost of communication delays.

6.1 Dynamic Library Structure and API features

Our library consists of 3 core functions that can be utilised:

- *simulation_initialize*: Executed once before the first time-step; intended for initialization of variables, memory, files, etc.
- *simulation_step*: Executed once in every time-step. Within this function, users can control the vehicles and monitor online the traffic environment
- *simulation_finalize*: Executed once before exiting the simulation; intended for deallocation of memory, saving log files, etc.

The user can build upon the development environment and communicate directly with SUMO via the API provided by our implementation (see Figure 1). Regarding the memory structure that contains the vehicles for a given road segment, vehicles are always retained in ascending order, according to their respective longitudinal position x. This information is updated dynamically, since vehicles have different speeds and, also, are following a route and may thus change road segments. This information is quite convenient for many reasons. First, it allows us to perform collision checks much more efficiently; and to perform the dynamic allocation of vehicles to road segments much faster. Furthermore, this information can be provided to the user via the API. For the remainder of this subsection, we showcase the current capabilities of our API:

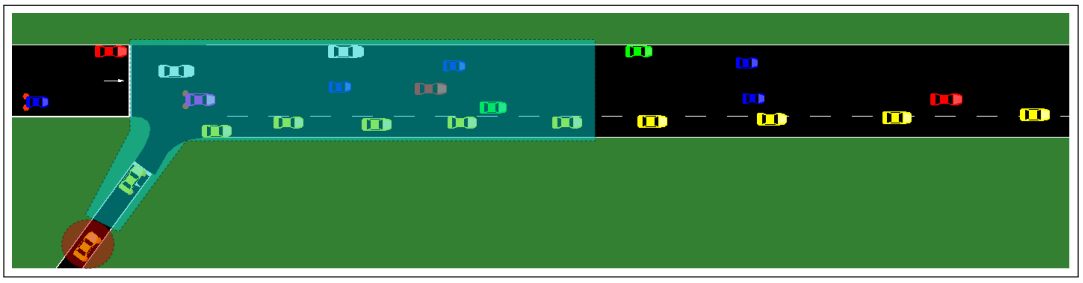

Figure 9: The appointed vehicle, located on an on-ramp and about to merge in the highway, has direct access to the vehicles located in the next segment.

6.1.1 Vehicles

A user can obtain information, such as: the IDs of all vehicles in the network; or only the IDs of vehicles in a specific road segment. In the latter case, the IDs are obtained in ascending order according to their (local) longitudinal position. Given a specific vehicle's ID value, information relevant to this vehicle is easily available, such as: its longitudinal and lateral positions (either local coordinates, as specified in Section 4, or global coordinates); longitudinal and lateral speeds, dimensions (length, width); the vehicle's type id (as defined when setting up the traffic environment); the vehicle's name (the one displayed by SUMO).

Besides information retrieval, one can *control* a vehicle via its ID, providing the appropriate control input to the relevant function. Moreover, through the API, the user can manually *insert* a new vehicle, either at the initialization phase, or at any arbitrary simulation time-step. In this case, the initial conditions and additional information should be provided by the user. That information involves: initial position and speed (longitudinal and lateral), a (valid) vehicle type, and a (valid) route for the vehicle to follow. We also retain internally a *desired speed* for each vehicle; and, through the API, we can obtain or change the existing value.

As mentioned, we use local coordinates, something quite convenient for designing a vehicle movement strategy, since the vehicle just follows a given route, and does not need to take into account the geometry of the road. However, this poses a limitation when a vehicle needs to observe downstream traffic in the next road segment, as the local coordinates of downstream vehicles correspond to a different road segment. For this reason, considering two vehicles that can potentially be located on different road segments, we provide information about the longitudinal and lateral distance from the perspective of one of the two vehicles. Complementary to the aforementioned functions is another feature that allows for each vehicle to easily observe *downstream* and *upstream traffic*, according to their routing scheme, i.e., they can observe vehicles that may be located in the upcoming or previous road segments. This essentially translates to obtaining an array of all the downstream (upstream) vehicles' ID values (up to a specified distance), in ascending order, based on the longitudinal distance from the ego-vehicle that requests such information, even if the downstream (upstream) vehicles are beyond the current road segment. For example, a vehicle in an on-ramp, which is advancing to merge on the mainstream, can directly observe downstream traffic from the main highway and be properly informed about the distances, as illustrated in Figure 9.

Moreover, we developed the necessary components to emulate a ring-road scenario using a single road segment. In this case, the vehicles are directly "transmitted" to the starting point of the segment instead of exiting. Longitudinal distance $d_x(i, j)$ is properly adjusted for this feature, e.g., a vehicle towards the end of the road segment will properly observe vehicles at

the start of the segment. In order to introduce vehicles in the ring-road, the recommended solution is to utilise the API, and place the vehicles at the desired density when initializing the simulation. When this functionality is extended to incorporate multiple road segments, the demand can be alternatively generated from an on-ramp, connected to the main highway that emulates the ring-road.

6.1.2 Road Segments

Similarly, users can obtain an array of the road segments' ID values in the network. For a given road segment, a user can obtain its properties (length, width), and additional online information for user-defined regions. Essentially, at any simulation step, we can directly gain information for a specific region of the road segment regarding the traffic density at that region, or the average speed of the vehicles. This is illustrated in Figure 10, where such information will be provided only according to the vehicles within the appointed region.

In addition, there is the flexibility to use these functions only for a specific type of vehicles, i.e., we can independently monitor each type of vehicles in the network. For instance, in Figure 10, we may be interested only in the type of vehicles indicated with pink color, thus the information we will receive will be calculated from the shown 2 pink vehicles only. The outlined functionalities related to road segments are valuable for several possible applications, including real-time traffic management via emulated V2I communication.

6.1.3 Detectors

One of the features of SUMO, when designing a traffic scenario, is the support of multiple types of detectors, which can be placed anywhere throughout the network. The simple 'E1' Loop Detectors of SUMO are integrated with our internal extensions. These loop detectors are represented by unique ID values. With this, we can monitor online the number of vehicles that have passed the corresponding detector. Additionally, we retain the same flexibility here regarding monitoring specific types of vehicles.

Lastly, a set of *event-based* functions has been embedded, triggered when specific situations occur, such as: a new vehicle entering the network, a vehicle reaching a final destination; a collision[4] between two vehicles; or a vehicle being located out of the road boundaries. Obviously, these functions are accommodated with information regarding the involved vehicles' IDs.

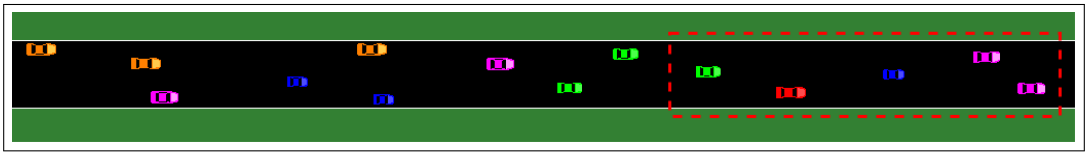

Figure 10: We can directly request information about density or average speed just by providing the upper and lower bounds of the region, e.g., if what is displayed is the total road segment, its length is 150 meters, and we are interested only in the last 50 meters, we define the region as $[100, 150]$.

[4]In lane-free vehicle movement, we consider that vehicles have a rectangular form (due to their length and width). Therefore, a collision between two vehicles is reported when their rectangular forms overlap.

7 Designing a Lane-Free Traffic Scenario

We should emphasise that we can design traffic scenarios exactly as we would normally do for the standard SUMO application. In order to design a traffic scenario, the core components that we need is a road network, specification of vehicle types along with their properties, and finally the demands. Lane-free vehicles can now be incorporated through the definition of a vehicle type. We have included an additional 'virtual' model option (besides the ones provided by SUMO), that the assigned vehicle types can now employ. When a user specifies the model as "lane-free", it will not conform to any standard lane-based model, but the user will be granted access to the control of all vehicles of this type through the dynamic library. Thus, these vehicles will now move according to the developed movement strategy. Any type of lane-free controller can be developed and adapted to the same structure, provided that it yields the appropriate control input, i.e., either the longitudinal and lateral accelerations (in m/s^2) or the forward acceleration (in m/s^2) and steering angle (in rad) (depending on the selection of movement dynamics, see Section 4) in every simulation time-step.

8 Future Extensions

Currently, vehicles are operating according to a predefined route, simply following a specified path from origin to destination. We rely on SUMO, which requires either just an origin and a destination segment (then a valid path is automatically generated upon initialization of the vehicles), or a (valid) full path consisting of the full sequence of road segments the vehicle should follow, if we wish more refined control. Also, SUMO provides some options regarding re-routing (either online or offline) of the vehicles, but none of them conforms with the flexibility we need in order to support the design of V2I applications for traffic management in a lane-free environment. One impending extension is to provide the necessary functionalities for allowing the user to assign turn probabilities at bifurcation nodes of the road network, in a lane-free traffic environment, omitting the origin-destination routing of the vehicles for specific applications.

Another forthcoming extension is the incorporation of lateral boundaries based on the desired path for vehicles to follow, along with the capability to control them at execution time through the API for emergent applications. In lane-based environments, vehicles can follow any (feasible) path by simply choosing lanes appropriately. For instance, a vehicle entering a highway from an on-ramp will typically need to perform a lane-change in order to merge on the highway. SUMO provides information regarding the availability of the road downstream through information on available lanes downstream. In our case, a vehicle adhering to the lane-free paradigm, wishing to enter a highway through an on-ramp, will again need to merge on the highway appropriately. Then, it is important to introduce an equivalent notion in a lane-free domain, since providing information on available lanes downstream is not appropriate in lane-free environments.

As of now, such maneuvers are performed with ad-hoc techniques, tied to specific traffic scenarios. Yet, there is a need to establish a formal way for vehicles to follow left and right boundaries on the road, guiding them to always operate within an appropriate lateral region, so as to comply with their routing. This is illustrated in Figure 11, where we observe a vehicle with a route initiating from the on-ramp and leading to the off-ramp. The blue lines indicate the left and right lateral bounds, relevant to the path vehicles need to follow. As such, the vehicle will be able to request information regarding these two lateral bounds through the API, at any longitudinal distance relevant to its position. Therefore, the vehicle will have the necessary time to adjust its behavior in order to be located within the bounds, and as a consequence to follow

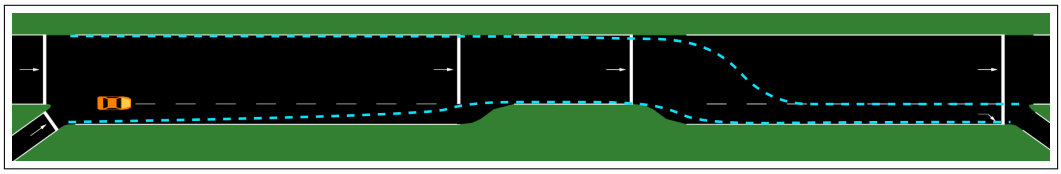

Figure 11: Left and right lateral boundaries for vehicles entering or scheduled to exit from the off-ramp.

its desired routing scheme. Besides the application for on-ramps and off-ramps, this feature is crucial to enable microscopic simulation on emerging applications such as internal boundary control in lane-free traffic for two-way streams, as introduced in [11]. This will also involve an online update process for the boundaries' lateral level through the API.

9 Related Work

In this section we provide a brief overview of some of the most popular external frameworks for SUMO, and further motivate our choice to directly extend this open-source project.

To begin with, Veins [21] is a network simulator that emphasizes on the realistic depiction of communication structures, simulating IEEE 802.11p and IEEE 1609.4 DSRC/WAVE network layers, and providing an extensive list of useful metrics. "iTETRIS Control System" (iCS) [18] delivers a platform for large-scale simulation and assessment of Cooperative Intelligent Transportation Systems (C-ITS). They additionally aspire to develop novel traffic control and communication strategies, along with new metrics that capture the overall performance of the network in terms of common traffic issues, such as: vehicle emissions, traffic congestion, travel time etc. VSimRTI [20] is aimed at providing a comprehensive and user-friendly environment. It embraces likewise realistic communication structures and energy consumption and is suitable for large-scale simulations. Flow [25] is a modular framework that facilitates the use of machine learning (in particular, reinforcement learning) methods to address the complexity of traffic dynamics. Likewise, Flow communicates with SUMO via TraCI, and is developed in Python. It also facilitates the creation of a generic simulator, appropriate for microscopic traffic modelling and the embedding of custom vehicle controllers.

All aforementioned frameworks are flexible and extensible, at least to a certain extent. Still, to accommodate our needs, considerable source code modifications and extensions would have been required, since no single framework provides our desired capabilities for lane-free traffic environments, while any such framework would induce a substantial additional overhead (w.r.t. time and memory) for communicating with SUMO. This led us to the choice of developing a direct extension of the SUMO framework, purpose-built for modeling the novel lane-free traffic paradigm.

Finally, apart from these frameworks based on SUMO, we should mention that AIMUSUN [1] provides a Software Development Kit with an API that enables users to develop custom controllers, in a way that is slightly resembling of our dynamic library. However, due to lack of access in the open-source code, AIMSUN could not be alternatively considered for lane-free microscopic simulation.

10 Conclusions

We presented TrafficFluid-Sim, a new, modular, and extensible simulator extension of SUMO, that supports a novel paradigm of lane-free traffic, the one coined by *TrafficFluid* [14]. We discussed the main components of the extensions, as well as forthcoming implementations. The simulator is already employed for the evaluation and expansion of vehicle movement strategies [14, 23, 26] under the novel traffic paradigm. Videos illustrating the presented insertion policies, and a simulation example with one on-ramp when employing an ad-hoc movement strategy (see [14]), are available at https://bit.ly/3tf53jD.[5] We intend to expand the existing components and, following its evaluation and feedback by the community, to elevate it into a fully-fledged, feature-rich platform that will enable the rapid progress of novel lane-free traffic research approaches; and, more generally, serve the community's needs and demands.

Acknowledgment

The research leading to these results has received funding from the European Research Council under the European Union's Horizon 2020 Research and Innovation programme/ ERC Grant Agreement n. [833915], project TrafficFluid.

References

[1] Aimsun. *Aimsun Next 20 User's Manual*. Barcelona, Spain, aimsun next 20.0.3 edition, 2021. [In software].

[2] Kay Axhausen, Kai Nagel, and Andreas Horni, editors. *Multi-Agent Transport Simulation MATSim*. Ubiquity Press, London, Aug 2016.

[3] Richard Dorf and Robert Bishop. *Modern Control Systems*. Pearson, 13th edition, 2017.

[4] David Elliott, Walter Keen, and Lei Miao. Recent advances in connected and automated vehicles. *Journal of Traffic and Transportation Engineering (English Edition)*, 6(2):109–131, 2019.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[6] Paweł Gora and Inga Rüb. Traffic models for self-driving connected cars. *Transportation Research Procedia*, 14:2207–2216, 2016. Transport Research Arena TRA2016.

[7] Venkatesan Kanagaraj and Martin Treiber. Self-driven particle model for mixed traffic and other disordered flows. *Physica A: Statistical Mechanics and its Applications*, 509:1–11, 2018.

[8] Iasson Karafyllis, Dionysios Theodosis, and Markos Papageorgiou. Analysis and control of a nonlocal pde traffic flow model. *International Journal of Control*, 2020.

[9] Iasson Karafyllis, Dionysios Theodosis, and Markos Papageorgiou. Lyapunov-based two-dimensional cruise control of autonomous vehicles on lane-free roads. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 2683–2689, 2021.

[10] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *21st IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[11] Milad Malekzadeh, Ioannis Papamichail, Markos Papageorgiou, and Klaus Bogenberger. Optimal internal boundary control of lane-free automated vehicle traffic. *Transportation Research Part C: Emerging Technologies*, 126(103060), 2021.

[12] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., USA, 1 edition, 1997.

[5] The interested reader may refer also to https://bit.ly/3K4azwz and https://bit.ly/3K3olzc for work related to the simulator, and to https://trafficfluid.tuc.gr for more information regarding TrafficFluid.

[13] On-Road Automated Driving (ORAD) committee. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*, 2018.

[14] Markos Papageorgiou, Kyriakos-Simon Mountakis, Iasson Karafyllis, Ioannis Papamichail, and Yibing Wang. Lane-free artificial-fluid concept for vehicular traffic. *Proceedings of the IEEE*, 109(2):114–121, 2021.

[15] Ioannis Papamichail, Nikolaos Bekiaris-Liberis, Anargiros I. Delis, Diamantis Manolis, Kiriakos-Simon Mountakis, Ioannis K. Nikolos, Claudio Roncoli, and Markos Papageorgiou. Motorway traffic flow modelling, estimation and control with vehicle automation and communication systems. *Annual Reviews in Control*, 48:325–346, 2019.

[16] Philip Polack, Florent Altché, Brigitte d'Andréa Novel, and Arnaud de La Fortelle. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 812–818, 2017.

[17] Mizanur Rahman, Mashrur Chowdhury, Yuanchang Xie, and Yiming He. Review of microscopic lane-changing models and future research opportunities. *IEEE Transactions on Intelligent Transportation Systems*, 14(4):1942–1956, 2013.

[18] Michele Rondinone, Julen Maneros, Daniel Krajzewicz, Ramon Bauza, Pasquale Cataldi, Fatma Hrizi, Javier Gozalvez, Vineet Kumar, Matthias Röckl, Lan Lin, Oscar Lazaro, Jérémie Leguay, Jérôme Härri, Sendoa Vaz, Yoann Lopez, Miguel Sepulcre, Michelle Wetterwald, Robbin Blokpoel, and Fabio Cartolano. iTETRIS: A modular simulation platform for the large scale evaluation of cooperative its applications. *Simulation Modelling Practice and Theory*, 34:99–125, 2013.

[19] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition, 2009.

[20] Björn Schünemann. V2X simulation runtime infrastructure VSimRTI: An assessment tool to design smart traffic management systems. *Computer Networks*, 55(14):3189–3198, 2011.

[21] Christoph Sommer, Reinhard German, and Falko Dressler. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Transactions on Mobile Computing (TMC)*, 10(1):3–15, January 2011.

[22] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[23] Dimitrios Troullinos, Georgios Chalkiadakis, Ioannis Papamichail, and Markos Papageorgiou. Collaborative multiagent decision making for lane-free autonomous driving. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 1335–1343. International Foundation for Autonomous Agents and Multiagent Systems, 2021.

[24] Michael Wooldridge. *An Introduction to MultiAgent Systems*. Wiley Publishing, 2nd edition, 2009.

[25] Cathy Wu, Aboudy Kreidieh, Kanaad Parvate, Eugene Vinitsky, and Alexandre M Bayen. Flow: A modular learning framework for autonomy in traffic, 2017.

[26] Venkata Karteek Yanumula, Panagiotis Typaldos, Dimitrios Troullinos, Milad Malekzadeh, Ioannis Papamichail, and Markos Papageorgiou. Optimal path planning for connected and automated vehicles in lane-free traffic. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 3545–3552, 2021.