Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-27 (2022) 89–95

Agile Multi-Agent Model Architecture for Intelligent Intersection Traffic Simulation ⋆

Alexander L. Gratzer * Alexander Schirrer * Stefan Jakubek *

* *Institute of Mechanics and Mechatronics, TU Wien, Vienna, Austria (e-mail: alexander.gratzer@tuwien.ac.at).*

Abstract: Urban traffic can naturally be modeled as a distributed and heterogeneous multi-agent dynamic system. To allow a scalable solution for simulation, control and information management, a simple but powerful generic model architecture is proposed in this work, supporting various different traffic participant types. It is shown that even though the agent actions can be restricted to a fairly small set of basic operations, a remarkably rich global system functionality and complex information flow can be modeled. The resulting traffic dynamics and information model is inherently flexible and agile with respect to participating agents and their evolution of system dynamics while enabling efficient and parallelizable simulation, control, and prediction computations. Selected use cases for traffic monitoring, management, and control are outlined, and the model capabilities are demonstrated.

Keywords: Multi-agent systems, Traffic control systems, Multi-vehicle systems, Decentralized Control and Systems, Mechatronics for Mobility Systems, Modeling and simulation of transportation systems, Information processing and decision support, Simulation, Intelligent Transportation Systems.

1. INTRODUCTION

Increasing traffic in inner-city areas aggravates the conflict between safety, efficiency, and environmental impact. In particular, intersections are critical nodes in any urban traffic network, so modeling their traffic dynamics accurately is needed to enable real-time simulation, optimization, and advanced control to improve the traffic system as a whole. In principle, modern information and communications technology (ICT), communication and control concepts offer the potential to comprehensively manage and optimize intersection traffic in a fine resolution in real time, considering the requirements of all road users, interpret the situations in the best possible way and implementing individual, coordinated, cooperative control strategies to realize a holistic optimum. Currently, however, intersection management is largely simplified in practice, mostly only controlled via fixed, pre-defined traffic light phase schedules. Inflexible, sub-optimal traffic light control can lead to unnecessary traffic jams and emissions.

A broad body of research has developed with respect to traffic modeling, control, and optimization. Microscopic traffic models represent each traffic participant as an individual particle. Such a model provides high resolution and allows to directly integrate control schemes and behavioral models while simulating interactions between individual participants. The main drawbacks are the significant computational cost (scaling with the number of considered traffic participants) and the necessity to provide valid and calibrated behavior models for all participants. Such models allow offline traffic signal optimization tasks (Chao et al., 2015) with a limited number of traffic participants. Commercial and open-source traffic simulation software is available, such as PTV Vissim (Fellendorf and Vortisch, 2010), SUMO (Lopez et al., 2018) or CARLA (Dosovitskiy et al., 2017). Large-scale traffic systems are typically modeled via traffic flow and density fields in a distributed-parameter approach ("macroscopic model"), instead of resolving individual particles.

Centralized control architectures rely on the central collection and availability of all system data in real-time, on having a complete system model, and on being able to solve the associated (massive) global optimization problem. This theoretically allows network-wide traffic optimization (Ye et al., 2019) with globally optimal performance, but generally does not scale well as systems grow larger and more complex. In turn, hierarchical, decentralized and distributed control concepts Christofides et al. (2013) have been proposed which divide the large-scale problem into smaller subsystems of manageable size. These concepts typically achieve sub-optimal performance but scale better and come with reduced communication requirements and improved robustness properties. Often, cooperation-establishing methods are put forward to improve stability, convergence, and performance in the large-scale system perspective (Killian et al., 2016).

Multi-agent traffic simulation approaches have been proposed in the literature: Doniec et al. (2008) investigate behavioral modeling of each agent (instead of classical car-following models) to realize realistic decision-making, including norm violations, in intersection traffic situations. Wei et al. (2019) provide a survey on recent multi-agent-

⋆ This work was supported by the Austrian Research Promotion Agency (FFG) via the research project *Intelligent Intersection* (ICT of the Future, grant 880830).

10.1016/j.ifacol.2022.10.493

based traffic signal control methods, particularly covering approaches based on Reinforcement Learning.

To minimize the design complexity in multi-agent traffic models, such model framework should provide each agent with simple access to the relevant neighborhood information, including neighbor motion prediction functionality, as well as efficiently incorporate the relevant agent dynamics, both for simple models as well as for heterogeneous types of traffic participants or high-fidelity models. To the best of the authors' knowledge, no multi-agent traffic model is available which provides these features efficiently in a control-related software environment such as MATLAB®. In contrast to full micro-simulation tools, it is desirable to obtain a lean model structure that can easily be integrated into optimization and model-based control tasks. It is this research gap that we attempt to close by the model proposed in this work.

The main contribution of this work is an agile multi-agent model that allows the simple and flexible treatment of typical simulation cases and control decisions in a distributed traffic context. Agile refers to the fact that, by design, the model architecture allows to incorporate heterogeneous agents in a time-varying system structure.

One core feature of the proposed model architecture is to organize real-time access to the set of current, relevant neighbor agents, so that any ego agent can directly implement basic control with respect to this environment. This structure facilitates advanced control designs such as obstacle-avoidance model-predictive vehicle control. Also, it aids the implementation of social-force pedestrian dynamics.

2. METHODOLOGY

As main contribution of this work, a fundamentally simple but versatile multi-agent system architecture for traffic management is proposed in the context of heterogeneous road traffic. Therein, the agents represent individual traffic participants (such as motorized human-driven vehicles, automated vehicles, bicycles, or even pedestrians). Also, control entities such as classical or advanced (adaptive) traffic lights can be regarded as agents. The overarching modeling goal is to obtain a well-defined traffic model suitable for microscopic simulation, short-term prediction, as well as control design and optimization. Thereby the prediction horizon should cover at least the time-to-standstill to enable safe control/driving decisions. It becomes evident in the following that all these agents share a common set of generic tasks that have to be carried out by each agent repetitively. For simplicity, a uniform sampling time $T_\mathrm{s} \in \mathbb{R}^+$ is utilized for all considered agents in this work.

Besides defining these generic agent tasks, global topological information (such as road or intersection topology, global traffic rules, predefined routes, and system boundary conditions) are defined and managed through a global model, corresponding to the modeled "world". The set of all considered agents $\mathcal{A}$ can grow or shrink over time as agents are created (i.e., spawned) or removed, respectively. We indicate superscript indices to identify individual agents where helpful for clarity, such as denoting the state vector of agent $i \in \mathcal{A}$ at time $t_k = T_\mathrm{s}\, k$ as $\boldsymbol{x}_k^i$. The subscripted index k indicates discrete time step $k \in \mathbb{N} \cup \{0\}$.

2.1 Generic agent tasks

The following generic agent tasks are found to provide the necessary functionality to model reasonable traffic interactions for each traffic participant. These are executed in sequential order in each model time step, for each agent in the model. This realizes a so-called "parallel" multi-agent system architecture, as the actions of each agent depend on the current state of itself and a specific set of neighboring agents at that time. The tasks carried out in the agents' time step (`step()` method) are listed in Algorithm 1. By carrying out the last step *after* all agents have evaluated their control laws in the current time step, the sequence in which the agent actions are computed does not affect the results. This evaluation can easily be parallelized, but a synchronization is required within each time step.

Algorithm 1 Generic Agent Tasks

T1) `getEgoStateEstimate()`:
Retrieve/estimate ego state, e.g. through communication, sensor fusion or an observer.

T2) `getEgoReferences()`:
Retrieve control goals/references.

T3) `getNearbyAgents()`:
Retrieve set of relevant neighbor agents, e.g. inside of detection radius.

T4) `collectObstaclePredictions()`:
Retrieve/measure/estimate neighbor state and motion predictions or create proxy predictions. Identify nearest relevant agent for car-following control via `getNearestObstacle()`.

T5) `[u] = AgentController.step()`:
compute control law

T6) `setInput(u)`:
apply control law

T7) `EOMstep(obj.GroundTruth.x, obj.Input)`:
compute ego EOMs / evolve time step

Fig. 1 sketches an exemplary situation of an ego agent instance (agent 1, blue box) of a connected automated vehicle (CAV), as well as several spatially nearby other agents (2-9) of various types. The most relevant neighbors, defined by suitable detection criteria, are considered as potential obstacles or interaction partners for the ego agent (inside the dashed-bounded region), whereas some agents lie outside and are not considered further in determining the ego agent behavior. In Section 3, some typical implementations of specific agent types are illustrated, showing that the proposed generic agent tasks indeed allow to cover a considerable variety of traffic participant behavior.

2.2 Agent states and dynamics

In the scope of this work, we consider each agent i, $i \in \mathcal{A}$, as a particle that performs horizontal planar motion. The agent position $\boldsymbol{p}^i = [X^i, Y^i]^T$, together with heading (attitude) in terms of a yaw angle ψ^i (right-hand-side convention) form the *pose* $\boldsymbol{\pi}^i = [X^i, Y^i, \psi^i]^T$. The position $\boldsymbol{p}^i$ is defined in a cartesian global reference frame (X, Y)

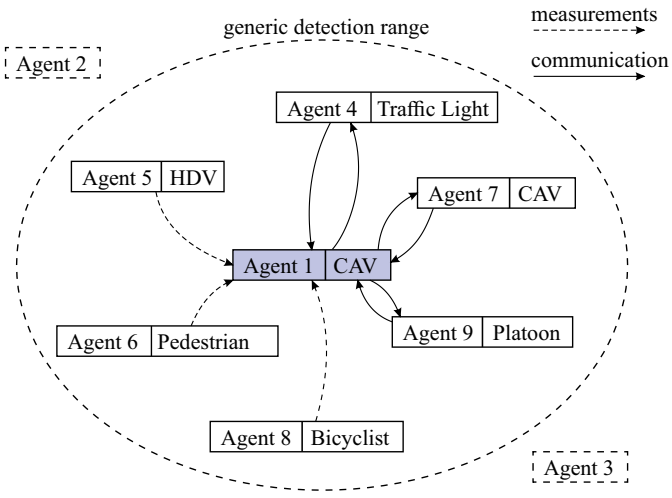

Fig. 1. Illustration of an ego agent and relevant neighbors.

("world frame"). The agent velocities $\boldsymbol{v}^i = [v_x^i, v_y^i]^T$ (longitudinal and lateral, respectively) are expressed in a body-fixed frame (x^i, y^i) related to the world frame (X, Y) by the rotation matrix $\boldsymbol{R}(\psi^i) = \begin{bmatrix} \cos\psi^i & -\sin\psi^i \\ \sin\psi^i & \cos\psi^i \end{bmatrix}$. Hence, this transformation enters the system equations:

$$\dot{\boldsymbol{p}}^i = \boldsymbol{R}(\psi^i)\, \boldsymbol{v}^i \tag{1}$$

In the scope of this work only planar motion is considered. Altitude variation and line-of-sight-based obstacle detection is a potential future research topic.

To allow heterogeneous agent dynamics, but also to enable a common understanding of pose and velocity information across all agents, each agent must be able to express its pose and velocity vectors as functions of the agent state vector in the form

$$\boldsymbol{p}^i = \boldsymbol{p}^i(\boldsymbol{x}^i) \text{ respectively } \boldsymbol{\pi}^i = \boldsymbol{\pi}^i(\boldsymbol{x}^i) \tag{2}$$

$$\boldsymbol{v}^i = \boldsymbol{v}^i(\boldsymbol{x}^i), \tag{3}$$

and also define its system equation ODE in the form

$$\dot{\boldsymbol{x}}^i = \boldsymbol{f}^i(\boldsymbol{x}^i, \boldsymbol{u}^i). \tag{4}$$

The state vector $\boldsymbol{x}^i \in \mathbb{R}^{n^i}$ represents the agent i's "ground truth" and may be chosen freely according to the agent type and its specific dynamics, provided (2)–(3) can be expressed. The flexible choice of the input vector $\boldsymbol{u}^i \in \mathbb{R}^{n_u^i}$ allows for kinematic or dynamic mechanical models of varying complexity and structure. As (4) is nonlinear, a numerical discretization in time is adopted. Here, a simple forward-Euler scheme is applied,

$$\boldsymbol{x}_{k+1}^i = \boldsymbol{x}_k^i + T_\mathrm{s}\, \dot{\boldsymbol{x}}_k^i. \tag{5}$$

After all agent's control laws have been evaluated in task T5 (which utilized current, consistent information of its ego and neighbor agents' states or predictions), the agent inputs $\boldsymbol{u}_k^i, i \in \mathcal{A}$ are defined and all system states are evolved in task T7 according to (4) and (5), providing the new states $\boldsymbol{x}_{k+1}^i$. In the following selected agent tasks are described in more detail.

2.3 Ego state estimation

The first task in Algorithm 1 is for an agent $i \in \mathcal{A}$ to obtain its current state estimate $\widehat{\boldsymbol{x}}^i$. Trivially, for example in simple simulations, this can directly depict the ground truth of the agent states ($\widehat{\boldsymbol{x}}^i = \boldsymbol{x}^i$), but in more realistic settings this step would perform state estimation and/or sensor fusion actions. The estimated state $\widehat{\boldsymbol{x}}^i$, respectively the extracted estimated pose $\widehat{\boldsymbol{\pi}}^i = \boldsymbol{\pi}^i(\widehat{\boldsymbol{x}}^i)$ and estimated velocities $\widehat{\boldsymbol{v}}^i = \boldsymbol{v}^i(\widehat{\boldsymbol{x}}^i)$ will then be utilized for prediction, communication to other agents, and control.

2.4 Selection of nearby neighbor agents

A crucial functionality, being highly valuable to each agent, is to obtain the subset of nearby neighbor agents which are close, relevant, or critical due to their proximity, type, or motion. Utilizing this information actually realizes feedback coupling between agents.

We define the *exemplary* set $\mathcal{O}^i$ of relevant neighbors, i.e. potential obstacles for agent i as

$$\mathcal{O}^i = \big\{ j \in (\mathcal{A} \backslash i) : \ |\boldsymbol{p}^j - \boldsymbol{p}^i| < d_\mathrm{max} \ \wedge \ \measuredangle(\boldsymbol{e}_x^i, \boldsymbol{p}^j - \boldsymbol{p}^i) \in [-\pi/2, \pi/2] \big\} \subset \mathcal{A}\,. \tag{6}$$

These *exemplary* criteria select those neighbor agents closer to agent i than a specified detection radius $d_\mathrm{max} \in \mathbb{R}^+$ and located in the forward-halfspace of agent i. The unit vector $\boldsymbol{e}_x^i = \boldsymbol{R}(\psi^i)\,[1,\,0]^T$ points into agent i's forward direction, and the operator $\measuredangle(\boldsymbol{a}, \boldsymbol{b}) \in (-\pi, \pi]$ represents the angle of vector $\boldsymbol{b}$ with respect to $\boldsymbol{a}$, wrapped into the indicated interval.

Note that the selection criteria can easily be adapted with respect to the desired simulation output, e.g. when simulating lane change dynamics also agents in the negative half space of the ego vehicle may be of relevance.

If the subset $\mathcal{O}^i$ (constructed only from the (estimated) pose information of the agents) is much smaller than $\mathcal{A}$, i.e. $|\mathcal{O}^i| \ll |\mathcal{A}|$, the number of considered interaction partners and evaluation effort is significantly reduced. In large-scale multi-agent systems, advanced data structures such as binary space partitioning trees could be utilized to avoid the need to iterate through all agents in the neighbor selection process, see Dong et al. (2019) and references therein for an overview.

2.5 Obstacle predictions

At time index k, the set $\mathcal{O}^i$ now contains all potential obstacles for agent i. For further use of information on these obstacles for the ego control decisions, it is vital to collect both a current state estimate as well as the predicted trajectories of the pose and velocity of agent $j \in \mathcal{O}^i$ over a specified prediction horizon $N_\mathrm{p} \in \mathbb{N}$. Depending on the communication capabilities of agent i and $j \in \mathcal{O}^i$, the following cases can be distinguished:

- no communication: Agent i estimates the current state $\widehat{\boldsymbol{x}}^{j|i}$ of agent j and an exogenous prediction of agent j's pose and velocity $\widehat{\boldsymbol{p}}_{k+l|k}^{\mathrm{pred},j|i}, \widehat{\boldsymbol{v}}_{k+l|k}^{\mathrm{pred},j|i}, l = 1, .., N_\mathrm{p}$.
- communication of state estimates: Agent i retrieves the current state (pose, velocity) from agent j, but no predicted trajectory. Then, agent i estimates an exogenous prediction of agent j's pose and velocity as above.

- full communication: Agent i retrieves both the current state estimate and predicted trajectory from agent j via communication. These data are utilized, assuming that they contain "first-hand" model information of agent j.

Note that the motion predictions of the agents in $\mathcal{O}^i$ are inherently uncertain which needs to be considered for traffic safety as discussed in Gratzer et al. (2022a).

2.6 Ranking and selecting the critical obstacle

To transform the obstacle information into a simple and usable form for the agent's control laws, it is useful to establish some criticality ranking of the set of potential obstacles $\mathcal{O}^i$, eventually providing one "most critical" obstacle which is regarded by the control law in a suitable way. As another contribution of this work this approach allows to directly implement classical car-following models which will be discussed later.

Formally, we can define a quantitative criticality criterion $C^{i,j} > 0$ where obstacles with smaller values are considered more critical (closer, or more imminent) than those with larger values.

The authors propose, for an obstacle $j \in \mathcal{O}^i$, the following simplified method to evaluate criticality $C^{i,j}$ based on the ego state $\widehat{x}^i$, the ego reference path $\boldsymbol{p}^{\mathrm{ref},i}(s)$ (arc length parameter s), and obstacle motion predictions $\widehat{\boldsymbol{p}}_{k+l}^{\mathrm{pred},j|i}$ ($l = 1, .., N_\mathrm{p}$). The ego vehicle is assumed to be close to its reference path, and the arc length parameter $s \in \mathbb{R}$ increases continuously in ego forward direction. Let $\sigma^{j\perp i} = \arg\min_s |\boldsymbol{p}^{\mathrm{ref},i}(s) - \widehat{\boldsymbol{p}}^j|$ denote the arc length parameter obtained when projecting agent j onto the ego reference path. Consequently, $\sigma^{i\perp i}$ is the arc length parameter of the projected ego position.

- If the current obstacle position $\widehat{\boldsymbol{p}}_k^{j|i}$ lies sufficiently close to the ego reference path $\boldsymbol{p}^{\mathrm{ref},i}$, set $C^{i,j} = \sigma^{j\perp i} - \sigma^{i\perp i}$.
- Otherwise, if the obstacle prediction $\widehat{\boldsymbol{p}}_{k+l}^{\mathrm{pred},j|i}$ lies sufficiently close to the ego reference path $\boldsymbol{p}^{\mathrm{ref},i}$ *and* $\measuredangle(\boldsymbol{e}^{\mathrm{ref},i}, \widehat{\boldsymbol{v}}^{j|i}) > 0$, set $C^{i,j} = \sigma^{j\perp i} - \sigma^{i\perp i}$. The angle condition (reference path orientation vector $\boldsymbol{e}^{\mathrm{ref},i}$ at the intersection point vs. obstacle velocity vector) tests whether the obstacle is coming from the right (right-of-way in right-lane-traffic systems).

Finally, obstacles with $C^{i,j} \leqslant 0$ are disregarded as they correspond to obstacles behind the ego agent, and the most critical obstacle $j^* = \arg\min_j C^{i,j}$ is chosen and provided as relevant obstacle to the control law. The typical car-following quantities (such as projected relative velocity Δv, projected distance d, projected time-to-collision TTC) can be evaluated accordingly and are utilized in evaluating the control law.

The proposed criterion could also be formulated with time-to-collision instead of longitudinal distance, or extended to include the predictions more accurately.

When employing more advanced control schemes it can be useful to identify and consider multiple critical obstacles.

3. IMPLEMENTATION

3.1 Human-driven vehicle (HDV)

As an exemplary implementation of a powered human-driven vehicle, a basic single-track model is employed to define the uncontrolled vehicle equations of motion. Their inputs are chosen as the steering angle and the longitudinal acceleration, so lateral and longitudinal control is required. This is accomplished here by suitable, well-known surrogate controllers taken from literature. Obstacle detection and information is processed and fed to the longitudinal control so that it can react accordingly. Lateral obstacle avoidance capabilities can be added in the same fashion, but are out of scope of this work.

Vehicle model Choosing the agent dynamics according to a kinematic single-track model leads to the following equations of motion (EOM), whereby the index i is dropped:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\psi} \\ \dot{v}_x \\ \dot{v}_y \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} \\ v_x \tan(\delta)/L_\mathrm{wb} \\ a_x \\ 0 \end{bmatrix}, \tag{7}$$

with state vector $\boldsymbol{x} = [X,\ Y,\ \psi,\ v_x,\ v_y]^\mathrm{T}$ and input vector $\boldsymbol{u} = [\delta,\ a_x]^\mathrm{T}$ (steering angle and longitudinal acceleration, respectively). The parameter L_wb represents the wheel base distance. First order dynamics for the lower level controller $v_x = f(a_x)$ can be implemented for more realistic vehicle dynamics.

Driver model The driver model comprises a longitudinal and a lateral surrogate controller. The driver controls the speed of the ego vehicle in relation to the vehicle ahead of it. We identify the traffic participant "in front" of the ego agent here by the obstacle detection and selection criteria outlined in Sections 2.4–2.6. The identified nearest or most critical obstacle is considered as predecessor and could be an actual vehicle preceding the ego vehicle, an obstacle (such as another vehicle crossing the ego path), or the stopping line corresponding to a traffic light that soon turns red or already shows red. This unified treatment allows simple car-following models to be used and react plausibly in the presence of obstacles or traffic lights. For longitudinal control we implement the *Intelligent Driver Model (IDM)* (Treiber and Kesting, 2013) in this example agent, which results in the acceleration given by

$$a = a_\mathrm{max}\left(1 - \left(\frac{v}{v_\mathrm{ref}}\right)^{\delta_\mathrm{IDM}} - \left(\frac{s^*(v, \Delta v)}{\Delta s}\right)^2\right) \tag{8}$$

with the reference velocity v_ref, the bumper-to-bumper distance Δs and relative velocity Δv to the predecessor vehicle, the acceleration exponent δ_IDM and the desired distance

$$s^*(v, \Delta v) = s_0 + \max\left(0, h\,v + \frac{v\,\Delta v}{2\sqrt{b\,a_\mathrm{max}}}\right) \tag{9}$$

with the standstill distance s_0 and the comfortable braking deceleration b. Of course any of the well known car-following models can be used, e.g. the psycho-physical *Wiedemann99 model* (Wiedemann, 1974), the *Human Driver Model (HDM)* (Treiber and Kesting, 2013) or the

Krauß car-following model (Krauß, 1998). The lateral position of the ego vehicle is controlled with respect to its current velocity and yaw angle ψ via the geometrical path-tracking *Stanley* controller (Hoffmann et al., 2007):

$$\delta = \begin{cases} \psi + \arctan \frac{K\, e}{v} & \text{if } |\psi + \arctan {}^{K\, e}/_{v}| < \delta_{\max} \\ \delta_{\max} & \text{if } \psi + \arctan {}^{K\, e}/_{v} \geqslant \delta_{\max} \\ -\delta_{\max} & \text{if } \psi + \arctan {}^{K\, e}/_{v} \leqslant -\delta_{\max} \end{cases} \quad (10)$$

with saturation at $\pm\delta_{\max}$, the cross-track error e of the front wheels to the nearest point on the ego reference path $\boldsymbol{p}^{\mathrm{ref},i}$, and the position gain K.

3.2 Connected automated vehicle (CAV)

Using the same vehicle model as in Section 3.1 and depending on the connectivity of the vehicle, adaptive cruise control (ACC) or cooperative ACC (CACC) strategies are used for longitudinal and lateral control. The fact that the proposed system model architecture provides each agent with a pre-processed real-time environment of relevant obstacles with their motion predictions greatly simplifies the design of advanced, distributed model-predictive control. One such control approach has been proposed in Thormann et al. (2020) and extended in Gratzer et al. (2022b), in which a distributed model-predictive platoon control law is augmented by collision safety constraints and additionally designed to provide robust string stability. Such a control concept can be realized in the present model architecture in a straightforward way, which is subject of ongoing research work.

3.3 Pedestrian

To model pedestrian behavior, the well-known social force model (Helbing and Molnár, 1995) can be implemented in the proposed generic agent structure in a straightforward manner. The social force model is founded on the idea that pedestrians move as a result of various "social force" terms that act repulsive or attractive versus/towards other nearby pedestrians or obstacles. As such, the outlined mechanism of finding and providing access to neighboring agents in each time step enables an elegant way to evaluate these social force terms, which can be understood as the control law of such pedestrian agents. The EOM are the free-mass double integrators in X and Y ($\psi \equiv 0$).

3.4 Traffic light

A traffic light with predetermined static signal phase plan is an example of an (immobile) infrastructure agent. The EOM are irrelevant, i.e. $\boldsymbol{p} = \text{const.}, \boldsymbol{v} \equiv \boldsymbol{0}$. A red traffic light is considered as a static obstacle. All other functionality (such as obstacle detection and communication) can be utilized beneficially in the present setting. We introduce an additional state in traffic light agents, the so-called *Time to Activate* $TTActivate$ which indicates when the traffic light next turns red and thus becomes an obstacle. For traditional traffic lights with a green-yellow-red switching sequence, this state is utilized to convey the remaining time to red during the yellow-phase to other agents without further communication requirements. TTActivate carries the information whether the traffic light is red ($TTActivate = 0$), green ($TTActivate = \infty$, then the traffic light is not considered as an obstacle by other agents) or yellow/green flashing ($0 < TTActivate < \infty$). Other agents which utilize traffic light timing predictions can thus interpret the TTActivate state accordingly to decide about braking or still passing the intersection.

4. RESULTS & DISCUSSION

To illustrate the descriptive power of the proposed multi-agent architecture, a synthetic intersection traffic test problem is simulated. The proposed model structures have been implemented in the MATLAB® software environment because it allows for efficient implementation of various control laws and strategies for the agents.

Fig. 2 depicts several time instances of this simulated scenario. It includes eight traffic light agents and 11 HDV agents with implemented detection radius $d_{\max} = 40\,\mathrm{m}$. The perspective of agent 10 is highlighted. All vehicles spawn at a distance of 100 m from the intersection center with initial longitudinal velocity $v_0 = 50\,\mathrm{km/h}$ and follow their predetermined routes. These routes define the reference paths of the vehicles $\boldsymbol{p}^{\mathrm{ref},i}$ and are considered known a priori only to the assigned agents. The route of agent 10 is displayed as a grey-dotted line. The control inputs and the longitudinal velocity of agent 10 are shown in Fig. 3. The simulation runs until $t = 20\,\mathrm{s}$ and each agent i utilizes a constant-velocity prediction to predict the trajectories of its detected obstacles in $\mathcal{O}^i$ since in this example the case of no communication is considered.

Table 1. Control and simulation parameters

Global Simulation Settings			
T_{s}	0.05	s	sampling time
$d_{\max}$	40	m	detection radius
N_{p}	30	samples	prediction horizon
v_0	50	50 km/h	initial velocity after spawning
Lateral Control Parameters (Stanley, (10))			
$\delta_{\max}$	35	°	steering angle saturation
K	2.5	s^{-1}	position gain
Longitudinal Control Parameters (IDM, (8), (9))			
v_{ref}	55	km/h	reference velocity
$a_{\max}$	2	$\mathrm{m/s^2}$	maximum acceleration
b	1.5	$\mathrm{m/s^2}$	comfortable braking deceleration
δ_{IDM}	4	/	acceleration exponent
h	1	s	time gap
s_0	1	m	standstill distance

At $t = 5.4\,\mathrm{s}$, agent 10 has not yet entered the drawing area, but already detects agents 1,2,7,8 and its predecessor vehicle 9, for which a constant velocity prediction is displayed (red circles). The (ego) constant velocity prediction of agent 10 is displayed in green. Note that the agents outside the detection range of vehicle 10 are not visible to the agent, i.e. they are not in $\mathcal{O}^{(10)}$. At $t = 6.8\,\mathrm{s}$ agent 9 reduces its velocity to give agent 11 right of way. Vehicles 12 and 13 approach the red traffic lights with reduced velocity. At $t = 9.5\,\mathrm{s}$ agent 9 leaves the intersection while agent 10 predicts the collision point with agent 15 on its path (blue hexagram) and brakes to give right of way. As soon as agent 15 cleared 1.5 m from agent 10's path after crossing it, agent 10 disregards agent 15 as nearest relevant obstacle, identifies again agent 9 as its relevant predecessor ($t = 11.5\,\mathrm{s}$) and starts to accelerate.

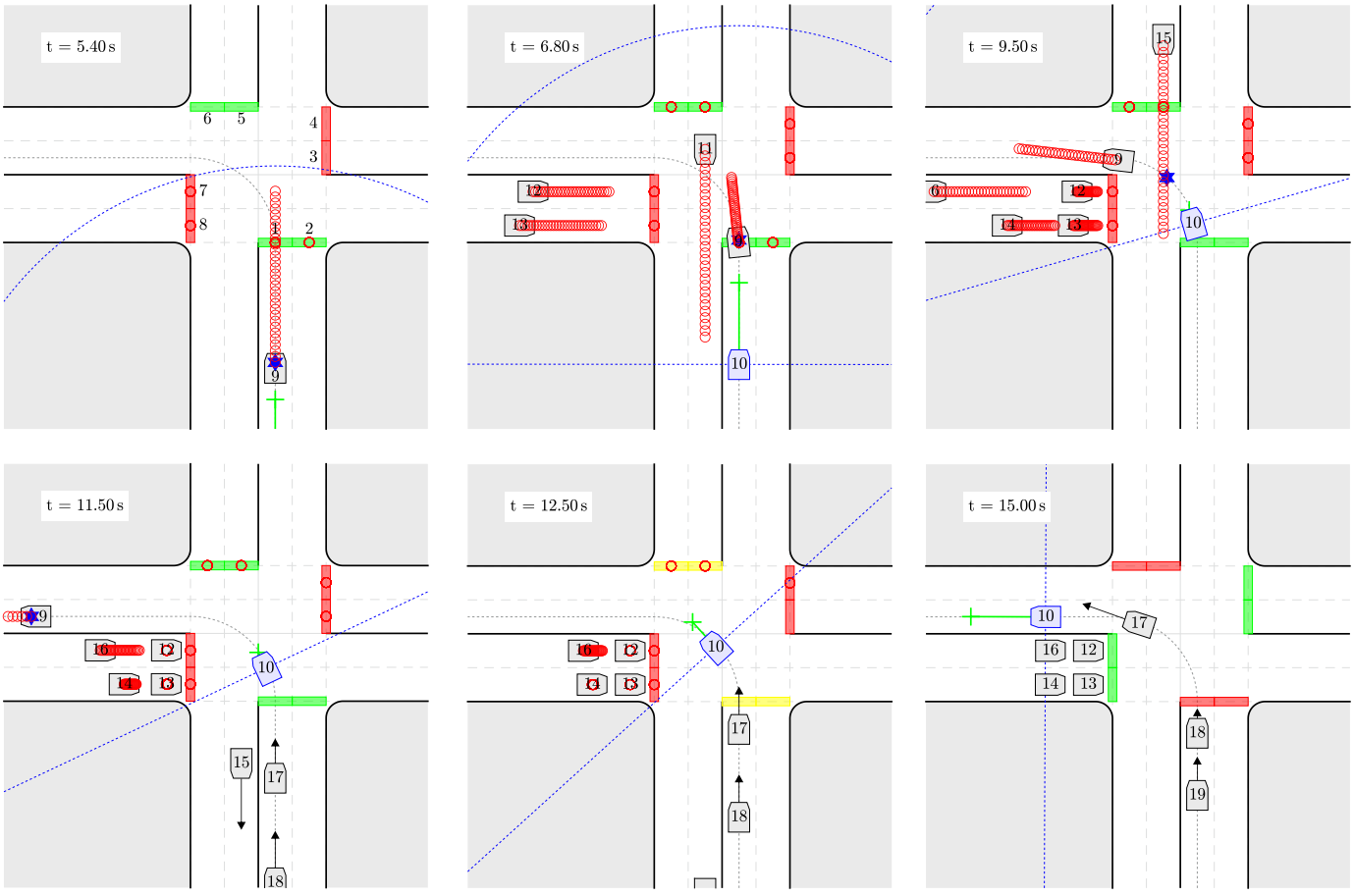

Fig. 2. Simulated intersection scenario: Left turns and passing a traffic light. The perspective of agent 10 is highlighted. Agents inside the blue dotted detection cone are detected and predicted (red circles) while the ego prediction is displayed in green. The blue hexagram indicates the predicted collision point with the most critical obstacle.

At $t = 12\,\mathrm{s}$ traffic light agents 1,2,5,6 switch to yellow (3 s yellow-phase). Agent 17 calculates its clearing-time $t_{\mathrm{clear}} = (\Delta s + d_{\mathrm{buffer}})/v_{\mathrm{ego}}$ and enters the intersection since $t_{\mathrm{clear}} > TTActivate^{(17)}$. The buffer distance d_{buffer} ensures that the intersection is (mostly) cleared when switched to red light ($TTActivate = 0\,\mathrm{s}$). Agent 18 starts reducing its velocity since it will not be able to pass the traffic light. At $t = 15\,\mathrm{s}$ the traffic light agents switch while agent 17 has already cleared the collision-safety relevant part of the intersection. Agents 12,13,14,16 start to accelerate. After completing the left turn maneuver, agent 10 is about to reach its reference velocity $v_{\mathrm{ref}} = 55\,\mathrm{km\,h^{-1}}$.

Summing up, this example showed a considerable variety of behavior (obeying red lights, turning left with correct prioritization, reasonable car-following) and is resilient by design. If new agents (such as pedestrians) are added that interfere with existing agents, or if unexpected behavior (such as red-light or priority violations) occurs, the system has been seen to react resiliently and efficiently. By resilient, we refer to robustness against communication (prediction) losses, traffic rule violations, and uncertainties in the evolution of the system dynamics or the composition of agents, which has been observed in simulations. Nominal traffic priority rules are implemented by simple decisions in the way obstacles are regarded or disregarded. The model architecture proves to be inherently agile with respect to participating agents and the evolution of their system dynamics. An analysis of more complex phenomena, e.g. deadlocks, is out of scope here and part of future work.

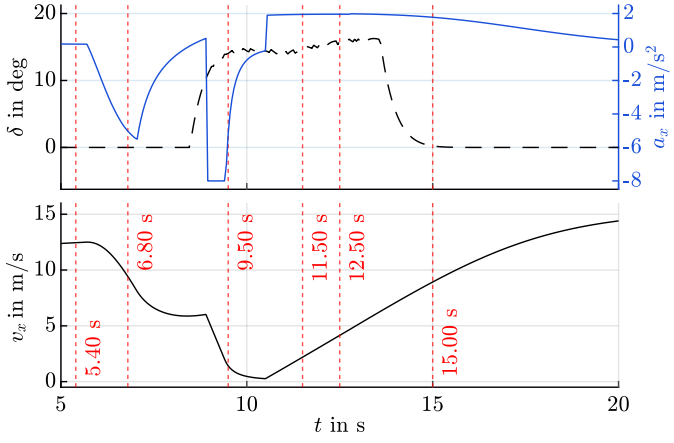

Fig. 3. Control inputs and longitudinal velocity of agent 10. The snapshot times of Fig. 2 are marked with vertical red dotted lines.

Finally, the computational effort for the conducted (arguably small-scale) simulation remains insignificant. The above mentioned simulation can be conducted in near real-time ($t_{\mathrm{sim}}^{\mathrm{calc}} = 22\,\mathrm{s}$) with a mean `step()` evaluation time $\bar{t}_{\mathrm{step}}^{\mathrm{calc}}$ per agent of 5.8 ms. A comparison with a second

simulation run, see Table 2, shows that $\bar{t}^{\text{calc}}_{\text{step}}$ per agent scales linearly with the maximum amount of considered agents $\mathcal{A}_{\max}$.

Table 2. Comparison of simulation times

	$\mathcal{A}_{\max}$	$t^{\text{calc}}_{\text{sim}}$	`step()` calls	$\bar{t}^{\text{calc}}_{\text{step}}$/agent
Sim 1	19	22 s	3821	5.8 ms
Sim 2	26	48 s	6095	8 ms
Δ_{12}	+37 %	+118 %	+60 %	+38 %

The simulations were carried out in a prototype MATLAB implementation by computing each agent step sequentially using an 2.3 GHz Intel® Core™ i7 CPU. Since the model structure allows simple parallelization of the computing tasks, significant reduction in simulation time is expected after further development.

Note that when increasing the complexity and size of an intersection the set of relevant neighbors $\mathcal{O}^i$ of an agent eventually saturates at a point of maximum neighbor agent density. Further increase of the overall number of simulated agents then does not increase the agent's `step()` computation time any further.

4.1 Outlook

With the successful implementation of the proposed multi-agent traffic model, a flexible tool is available for traffic simulation, optimization and control. In particular, the implementation is, albeit having nonlinear dynamics, straightforward to utilize in model-predictive control or optimization tasks. An indicative list of features that can potentially be implemented with small additional effort are:

- pedestrian recognition and prediction
- modeling the individual trust in legal and reasonable behavior of other road users (since the ranking and criticality decisions of potential obstacles lies in the responsibility of each agent)
- intelligent traffic light agents which perform optimal control / optimization tasks
- further aggregating agents such as intersection manager agents
- co-simulation use case (i.e., defer agent EOM evolution to dedicated simulation codes)
- linearization along predicted trajectory which allows efficient optimization via SQP approaches

These features are currently under development by utilizing the straightforward expandability of the proposed model architecture.

REFERENCES

Chao, Q., Deng, Z., and Jin, X. (2015). Vehicle-pedestrian interaction for mixed traffic simulation. *Computer Animation and Virtual Worlds*, 26(3-4), 405–412. doi:10.1002/cav.1654.

Christofides, P.D., Scattolini, R., Muñoz de la Peña, D., and Liu, J. (2013). Distributed model predictive control: A tutorial review and future research directions. *Computers and Chemical Engineering*, 51, 21–41. doi:10.1016/j.compchemeng.2012.05.011.

Dong, Y., Indyk, P., Razenshteyn, I., and Wagner, T. (2019). Learning space partitions for nearest neighbor search. doi:10.48550/ARXIV.1901.08544. URL `https://arxiv.org/abs/1901.08544`.

Doniec, A., Mandiau, R., Piechowiak, S., and Espié, S. (2008). A behavioral multi-agent model for road traffic simulation. *Engineering Applications of Artificial Intelligence*, 21(8), 1443–1454. doi:10.1016/j.engappai.2008.04.002.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). CARLA: An Open Urban Driving Simulator. (CoRL), 1–16. URL `http://arxiv.org/abs/1711.03938`.

Fellendorf, M. and Vortisch, P. (2010). Microscopic Traffic Flow Simulator VISSIM. 63–93. doi:10.1007/978-1-4419-6142-6{_}2.

Gratzer, A.L., Schirrer, A., Thonhofer, E., Pasic, F., Jakubek, S., and Mecklenbräuker, C.F. (2022a). Short-Term Collision Estimation by Stochastic Predictions in Multi-Agent Intersection Traffic. In *IEEE ICECET*, June.

Gratzer, A.L., Thormann, S., Schirrer, A., and Jakubek, S. (2022b). String Stable and Collision-Safe Model Predictive Platoon Control. *IEEE Transactions on Intelligent Transportation Systems*, PP, 1–16. doi:10.1109/tits.2022.3160236.

Helbing, D. and Molnár, P. (1995). Social force model for pedestrian dynamics. *Physical Review E*, 51(5), 4282–4286. doi:10.1103/PhysRevE.51.4282.

Hoffmann, G.M., Tomlin, C.J., Montemerlo, M., and Thrun, S. (2007). Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing. *Proceedings of the American Control Conference*, 2296–2301. doi:10.1109/ACC.2007.4282788.

Killian, M., Mayer, B., Schirrer, A., and Kozek, M. (2016). Cooperative Fuzzy Model-Predictive Control. *IEEE Transactions on Fuzzy Systems*, 24(2), 471–482. doi:10.1109/TFUZZ.2015.2463674.

Krauß, S. (1998). Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics. *D L R - Forschungsberichte*, (8).

Lopez, P.A., Behrisch, M., et al. (2018). Microscopic Traffic Simulation using SUMO. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2018-Novem, 2575–2582. doi:10.1109/ITSC.2018.8569938.

Thormann, S., Schirrer, A., and Jakubek, S. (2020). Safe and Efficient Cooperative Platooning. *IEEE Transactions on Intelligent Transportation Systems*. doi:10.1109/TITS.2020.3024950. URL `https://ieeexplore.ieee.org/document/9210204/`.

Treiber, M. and Kesting, A. (2013). *Traffic Flow Dynamics - Data, Models and Simulation*. Springer. URL `http://www.springer.com/us/book/9783642324598`.

Wei, H., Zheng, G., Gayah, V., and Li, Z. (2019). A Survey on Traffic Signal Control Methods.

Wiedemann, R. (1974). Simulation des Straßenverkehrsflusses. doi:10.1007/978-3-7091-8202-4{_}9.

Ye, B.L., Wu, W., Ruan, K., Li, L., Chen, T., Gao, H., and Chen, Y. (2019). A survey of model predictive control methods for traffic signal control. *IEEE/CAA Journal of Automatica Sinica*, 6(3), 623–640. doi:10.1109/JAS.2019.1911471.