Modelling Human Driving Behavior for Constrained Model Predictive Control in Mixed Traffic at Intersections

Johanna Bethge [*], **Bruno Morabito** [*], **Hannes Rewald** [*,**], **Adil Ahsan** [*], **Stephan Sorgatz** [**], **Rolf Findeisen** [*]

[*] *Laboratory for Systems Theory and Automatic Control, Otto-von-Guericke University Magdeburg, 39106 Magdeburg, Germany, (e-mail: {johanna.bethge, bruno.morabito, hannes.rewald, rolf.findeisen}@ovgu.de).*
[**] *Volkswagen AG, Berliner Ring 2, 38440 Wolfsburg, Germany*

Abstract: Safe autonomous passing of intersections with mixed traffic, including human drivers and autonomous vehicles, is challenging. We propose a tailored approach that provides guarantees despite uncertainties fusing learned models and model predictive control. A single autonomous vehicle is controlled by the predictive controller via acceleration and steering angle without assumption of a global controller. Each maneuver of the human behaviour is modeled with a neural network, which enters the predictive controller formulation as a constraint. As an example, we consider a single autonomous vehicle on an unsignalized intersection, which gives right-of-way to a human-driven vehicle. We show how human driving behavior can be modeled based on real recorded trajectory data and implemented in the proposed predictive control approach by dynamically changing the constraints of the optimization problem.

Keywords: Nonlinear model predictive control, multi-mode systems, machine learning, dynamic constraints

1. INTRODUCTION

Increasing travel safety, comfort and efficiency is the driver of all ongoing activity in the development of autonomous vehicles. Due to the extremely complex environment, reaching full or semi autonomy requires tackling a series of subproblems, such as minimization of fuel consumption and traveling time in urban environment (Van den Berg et al. (2004); Riegger et al. (2016)), traffic light optimization, (Guler et al. (2014); Tettamanti et al. (2008); Dresner and Stone (2004); Lin et al. (2013); Portilla et al. (2013)), obstacle avoidance (Makarem and Gillet (2013); Park et al. (2009); Widyotriatmo et al. (2009); Qian et al. (2016); Falcone et al. (2008); Borrelli et al. (2005)), overtaking (Park et al. (2009)), lane keeping (Falcone et al., 2008; Borrelli et al., 2005), autonomous parking (Hsieh and Ozguner (2008)) and navigation or path planning (Widyotriatmo et al. (2009); Alonso et al. (2011); Qian et al. (2016)).

One essential aspect of autonomous driving concerns the safety on intersections with mixed traffic, i.e. human driven vehicles and autonomous vehicles. Making decisions in such environments requires adequate prediction models for human driving behaviour. While approaches for approximate modelling of human drivers by first principle models exist, (Falcone et al., 2008; Borrelli et al., 2005), the use of machine learning approaches is most promising, (Bender et al. (2015); Gu and Hu (2002); Prokop (2001); Zyner (2018)). In this paper a Model Predictive Control (MPC) approach for the control of a autonomous vehicle is proposed. The MPC computes the optimal control actions, while considering the behaviour of the other vehicles as dynamic constraints. The behaviour of those vehicles is modeled using a neural network.

We will focus on a standalone predictive controller for a single autonomous (ego) vehicle on a 4-way unsignalized intersection with mixed traffic. The only control inputs are steering angle and acceleration of the ego vehicle. There is no further communication (V2X) or traffic light optimization assumed. Neither the traffic light nor the other vehicles are controlled. In the example, the ego vehicle gives, if necessary, right-of-way to one or more human driven vehicles (target vehicles).

The autonomous vehicle crosses the intersection following a predefined path, while considering the states of human driven vehicles as constraints. We assume that human drivers can take only a finite number of decisions i.e. turning left, right or going straight. We will refer to these decisions as *modes*. For the ego vehicle, the actual intention of the human driver (= mode) it is not known a priori. Hence, to avoid collisions, all likely modes need to be considered *at the same time* by the controller of the ego vehicle. The estimated human vehicle mode is updated online by using a machine learning model. To solve this problem, we exploited the concept of multi-mode predictive control (Bethge et al., 2018; Morabito et al.,

[⋆] BM and RF are affiliated to the International Max Planck Research School (IMPRS) for Advanced Methods in Process and Systems Engineering, Magdeburg. JB acknowledge support by the German Research Foundation in the frame of the GRK 2297.

2019). Contrary to these contributions, where the modes affect the dynamics of the system, in this paper the modes effect the constraints of the optimal control problem.

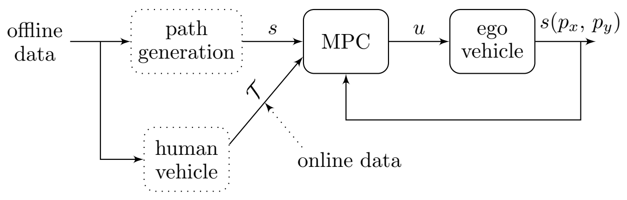

Fig. 1. Approach Architecture

The idea and structure of the approach, see Fig. 1, can be summarized as follows. First, the desired path of the ego vehicle is planned offline. The human driving behavior is learned offline from data. For each mode a separate neural network is trained. Online, the MPC controller determines the optimal control input trajectory, i.e. acceleration and steering angle, for the ego vehicle to follow the desired path, while considering the human-driven vehicles as constraints. The current mode and states of the human driven vehicles are determined from online measurements using nearest neighbor comparison (NNC). The first part of the optimal control input trajectory is applied to the ego vehicle. Afterwards, the states of the ego and the human driven vehicles are measured and the optimal control input trajectory is determined again with the updated states until the ego vehicle reaches the end of the desired path. This enables the controller to react to disturbances.

There are various datasets publicly available, which can be used for autonomous driving related tasks, e.g. (Yin and Berger, 2017). Datasets are suitable for specific areas of application such as highways (He, 2017), intersections (Bender et al., 2015) (Zyner, 2018) etc. Driving datasets may require preprocessing prior to the application of machine learning algorithms (Feng and Zhu, 2016; Zheng, 2015). We used a dataset provided by the Australian Center for Field Robotics (Bender et al., 2015).

The proposed fusion of a model predictive controller and a learning approach, has the following advantages:

- The human driving behavior can be modeled from real data.
- The predicted human driving behavior learned by a neural network can directly be considered as constraints in the controller.
- All likely modes can be considered as constraints and not only the most probable one.
- The probability of the modes can be evaluated online (at every time step).
- The multi-mode MPC reduces conservativeness as modes, which are very unlikely, can be removed online from the constraints of the optimal control problem.

The remaining paper is structured as follows. In Section 2 we describe how the human driving behavior was modeled from real data, while Section 3 outlines how the actual mode of the human drivers can be determined online. Path generation of the ego vehicle is covered in Section 4. In Section 5 the model predictive control scheme with multi-mode constraints for human drivers is outlined. Some simulation results for proof-of-concept are shown in Section 6, finally some short conclusions follow in Section 7.

2. MODELLING OF HUMAN DRIVERN VEHICLES

Based on a model, the proposed controller predicts the trajectory, i.e. the future states, of the human-driven vehicles. Here, neural networks were used as prediction models. The human vehicle states are predicted by regression of the observed data.

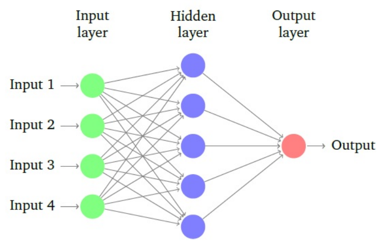

Fig. 2. Structure of a neural network.

The general structure of a neural network (Kruse et al., 2016; Goodfellow et al., 2016; Maas et al., 2013; Mitchell, 1997) is shown in Fig. 2. In general a neural network has one input layer, one output layer and an arbitrary number of hidden layer. Each neuron (shown as circles) receives an input, which is either a part of the feature vector ξ (input of the neural network) or a weighted combination of the neurons in the previous layer. The input of the neuron is proceeded by an activation function $\sigma(\xi)$, where the result is saved on the neuron itself. The output of the neuron is either a weighted connection (shown as arrows) to neurons in the next layer or to the output $h(\cdot)$ of the neural network. The weights θ are optimized trough a cost function by minimizing the error between the output from the training set and the hypothesis $h(\cdot)$, which is the output of the neural network.

The dataset (Bender et al. (2015)) includes measurements of the human-driven vehicles on an unsignalized intersection. Each data entry contains vehicles' latitude, longitude and orientation in addition to the timestamp at which the observation was taken. We consider 6 different modes. Each mode m is represented by two letters, the first letter indicating the initial cardinal direction of the target vehicle, and the second letter the final cardinal direction. Directions are defined as West (W), East (E), South (S).

$$\begin{aligned} m \in M &= [1, n_{\text{mode}}] \\ &= \{\text{WE, WS, EW, ES, SW, SE}\}. \end{aligned} \tag{1}$$

For each mode m we train a separate hypothesis $h(\theta_m, \cdot)$ with one hidden layer. Each feature in the feature vector $\xi(k) = (p_x(k), p_y(k), \psi(k))$ is represented by a single neuron in the input layer. The output of the neural networks $\hat{z}_m(k+1|k) = (p_x(k+1), p_y(k+1), \psi(k+1))$ are the predicted states $\hat{z}_m(\cdot)$, i.e. location (p_x, p_y) and yaw angle ψ, of a human-driven vehicle for the next time instant $k+1$. Hence, we have three neurons in the input layer as well as in the output layer. The hidden layer contains seven neurons. We use a sigmoid activation function (2), which leads to the following neural network (3):

$$\sigma(\xi) = \frac{1}{1+e^{-\theta\cdot\xi}}, \tag{2}$$
$$\mathbf{z}(l|k) = (\xi(k-l), \ldots, \xi(k))$$
$$\hat{z}_m(k+1) = h\left(\theta_m,\, \sigma(\cdot),\, \mathbf{z}(l|k)\right). \tag{3}$$

The neural network predicts the human driving behavior $\hat{z}_{m,j}(k+1)$ at the next time step $k+1$ for mode m and target vehicle j from the data points. Our neural network was trained using 70% of the dataset for each mode, while the test set contains 30% of the real trajectories for each mode. The data of vehicles trajectories were pre-processed to eliminate outliers and noise, and re-sampled to get constant time steps of $\Delta t = 0.1\,\mathrm{s}$. The algorithm was implemented using the Matlab Deep Learning Toolbox (Mat, 2018b). The weights θ of the connections are optimized during the training with Levenberg Marquardt algorithm by minimizing a cost function (Kruse et al., 2016; Mitchell, 1997). In our example, the cost function is the mean squared error between the predicted state and the measured state taken from the dataset.

3. CLASSIFICATION OF HUMAN VEHICLES

Because the human vehicles can execute multiple modes, c.f. Fig. 3, and to reduce conservativeness, there is a desire to determine the true mode of the human vehicles. When no information is available, all modes are considered equally likely. Hence all modes are taken into account as constraint in the optimal control problem. To reduce conservativeness, the probability of each mode is updated online, when new measurements are available. The modes, whose probabilities lie below a certain threshold will be eliminated from the constraints. To update the probability, we use nearest neighbor comparison (NNC) (Tan et al., 2017). The algorithm takes as an input a reference datasets, one for each potential mode. Here the main steps of NNC is summarized:

(1) At the current time step k, the partial trajectory $\mathbf{y}_j(k) = (\xi_j(1), \xi_j(2), \ldots, \xi_j(k))$ is observed. The nearest neighbor trajectory $\mathbf{y}^{(j)}_{m,r}$ for a target vehicle j is identified within the sets of recorded trajectories $\mathbf{Y}_m$ for each mode m:
$$\mathbf{y}^{(j)}_{m,r}(k) = \arg\min_{m,r} d(\mathbf{y}_j(k),\, \mathbf{y}_{m,r}(k)),$$
$$\forall\, m \in M,\, \forall r \in \mathbf{Y}_m.$$
This is done, using the Euclidean distance $d(\cdot)$:
$$d(\mathbf{y}_j(k),\, \mathbf{y}_{m,r}(k)) = \sqrt{\sum_{l=1}^{k} \left((\xi_j(l) - \xi_{m,r}(l)) \oslash \eta\right)^2}\ .$$
The normalization factors $\eta \in \mathbb{R}^3$ are chosen a priori ($\oslash$ is element-wise division).

(2) For each mode, distance $d(\cdot)$ between nearest neighbor trajectory $\mathbf{y}^{(j)}_{m,r}(k)$ and observed data points is converted to similarity $\gamma(\cdot)$:
$$\gamma(\mathbf{y}_j(k),\, \mathbf{y}^{(j)}_{m,r}(k)) = \frac{1}{1 + d(\mathbf{y}_j(k),\, \mathbf{y}^{(j)}_{m,r}(k))}\ .$$

(3) The probability p of the observed sequence is calculated, separately, for each mode $m \in M$:
$$p(\mathbf{y}_j(k) \mid m) \approx \frac{\gamma(\mathbf{y}_j(k),\, \mathbf{y}^{(j)}_{m,r})}{\sum_{m\in M} \gamma(\mathbf{y}_j(k),\, \mathbf{y}^{(j)}_{m,r})}\ .$$

(4) The normalized probability $\hat{p}$ and the maximum likelihood naive Bayes estimate lead to the predicted mode $\hat{m}_j$:
$$\hat{p}(\mathbf{y}_j(k) \mid m) = \frac{p(\mathbf{y}_j(k) \mid m)}{\sum_{m\in M} p(\mathbf{y}_j(k) \mid m)}\,,$$
$$\hat{m}_j(k) = \arg\max_{m\in M}\, \hat{p}(\mathbf{y}_j(k) \mid m).$$

This algorithm enables to classify the mode online after each new observation. However, determining the nearest neighbor trajectory in each mode (step 1 of the algorithm) is computationally expensive, especially for big datasets. For this reason, we compared NNC with a prototype path generation and comparison algorithm, c.f. Vasquez and Fraichard (2004). The classification accuracy
$$\text{accuracy}(k) = \frac{1}{k}\sum_{l=1}^{k} \begin{cases} 1, & \hat{m}_r(l) == m_r \\ 0, & \hat{m}_r(l) \neq m_r \end{cases}$$
for both algorithms after each time step $\Delta t = 0.1\,\mathrm{s}$ are shown in Fig. 4. It can be seen that nearest neighbour comparison provides significant better accuracy. Especially, when only a few observed data points are available. Therefore, nearest neighbour comparison is used in the example in Section 6. However, for real-time experiments, prototype path generation and comparison might be the better choice due to the lower computational time.

4. PATH GENERATION

The path of the ego vehicle is planned offline, using optimal rapidly exploring random trees (RRT) and the MATLAB Automated Driving Toolbox (Mat, 2018a). Once the path is defined, the controller of the ego vehicle follows this path by adapting the control input, while avoiding human driven vehicles on the intersection.

Here, the main idea of RRT is given. The path is modeled as a tree with vertices representing points in space, i.e. vehicle position $(p_x,\, p_y)$, and edges representing the paths connecting the points, see Fig. 5. The algorithm starts with an initial state q_i represented by the green vertex in Fig. 5 and with a goal state q_r represented by the red vertex. Then, a point q_a is randomly sampled from the configuration space. If the point does not satisfies the constraints, e.g. street border, it is discarded. Otherwise, the algorithm adds the point as a vertex. The vertex is then connected by an edge to the nearest reachable vertex q_n in the existing tree by evaluating the cost to reach this vertex. This process is repeated, until a feasible path that reaches the goal state q_r is obtained. The vertices are connected by Dubins segments (Dubins, 1957) to generate the path. To generate a smooth path, cubic spline interpolation is used.

We define the following costs for path generation:
$$\text{cost}_{\text{path}} = \left\{ \begin{array}{ll} 0.1,\ (p_x,\, p_y) \in & \text{ego vehicle's lane} \\ 0.5,\ (p_x,\, p_y) \in & \text{target vehicle's lane} \\ 1.0,\ (p_x,\, p_y) \notin & \text{road boundaries.} \end{array} \right\}.$$

The generated paths, given for the specified waypoints, are shown in Fig. 6. The reference path $\mathcal{P}$ maps path progress s to location $(p_x,\, p_y)$ and curvature $(dX,\, dY)$:

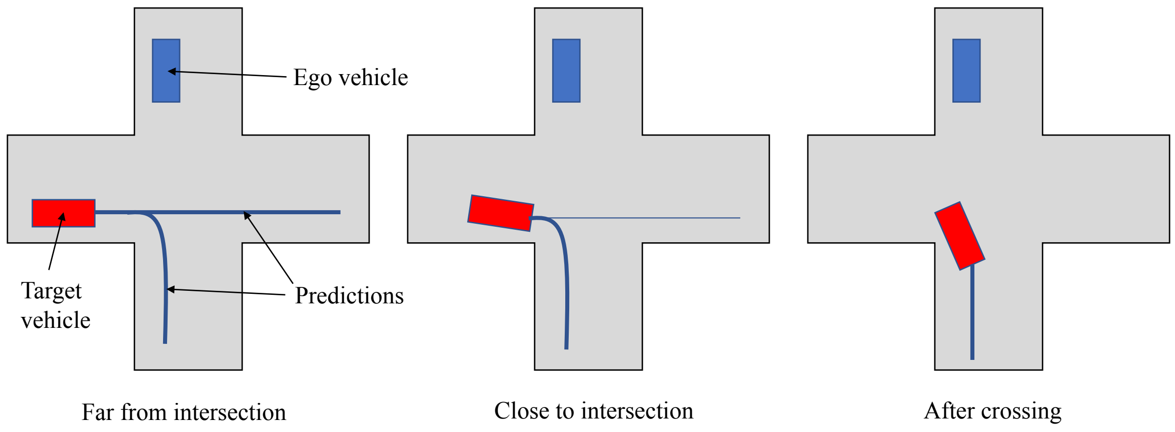

Fig. 3. The mode of the target vehicle is determined online, to predict its future trajectory. At the initial time step (left) all modes are possible and equally likely. After obtaining more measurements (middle), one mode is considered more likely than the other. Finally the likelihood of the mode lies below a certain threshold, the prediction of that mode is removed from the constraints (right).

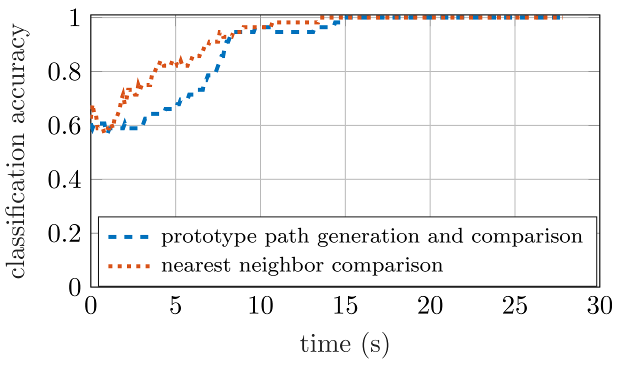

Fig. 4. Comparison of classification accuracy at each time step k for nearest neighbor comparison algorithm and prototype path generation and comparison.

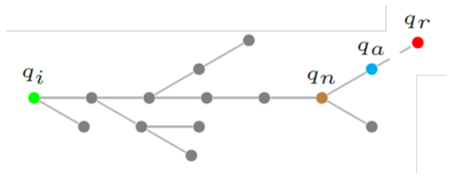

Fig. 5. Exploration of a rapidly random tree.

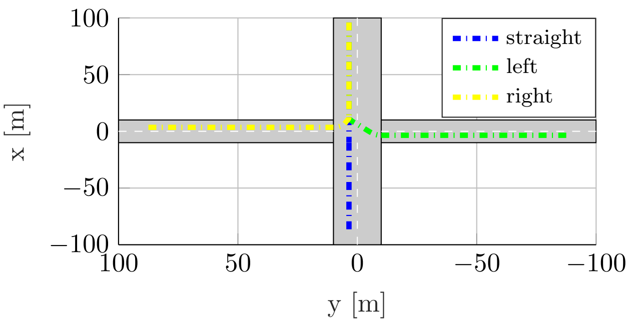

Fig. 6. Path of the ego vehicle coming from north defined a priori.

$$\mathcal{P} = \{s \in \mathbb{R} \mid 0 \le s \le 1\}. \tag{4}$$

5. MPC WITH MULTI-MODE CONSTRAINTS

The autonomous driving is realised by an MPC approach considering the human driven vehicles as constraints. We model the ego vehicle as a nonlinear system with states $x(t) = (v(t), p_{\mathrm{x}}(t), p_{\mathrm{y}}(t), \psi(t)) \in \mathbb{R}^{n_x}$ and control input $u(t) = (a(t), \delta(t)) \in \mathbb{R}^{n_u}$. The nonlinear dynamics $x(k+1) = f(x(k), u(k))$ of the ego vehicle are described by the following kinematic bicycle model (Kong et al., 2015):

$$\begin{aligned}\dot{p}_{\mathrm{x}} &= v\,\cos(\psi+\beta)\\ \dot{p}_{\mathrm{y}} &= v\,\sin(\psi+\beta)\\ \dot{v} &= a\\ \dot{\psi} &= \frac{v}{l_r}\sin\beta\\ \beta &= \tan^{-1}\frac{l_r}{l_f+l_r}\tan\delta.\end{aligned} \tag{5}$$

The states x of the ego vehicle are its velocity v, x- and y-position (p_x, p_y) and orientation ψ. The parameters $l_r = 1.8\,\mathrm{m}$ and $l_f = 1.4\,\mathrm{m}$ describe the distance of the center of gravity to the rear axles and the front axles, respectively. The control inputs are the steering angle δ and the acceleration a of the ego vehicle. Furthermore, the states $x \in \mathcal{X}$ and the control input $u \in \mathcal{U}$ are bounded in the compact and connected set $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ and the compact set $\mathcal{U} \subset \mathbb{R}^{n_u}$, respectively.

The target vehicles' behavior up to the time $k+l$ is iteratively modeled with a neural network trained offline (3) as follows

$$\begin{aligned}\hat{\mathbf{z}}_{j,m}(k+g|k) &= (\hat{z}_{j,m}(k+1|k), \dots, \hat{z}_{j,m}(k+g|k))\\ \hat{\mathbf{z}}_{j,m}(k+g+1|k) &= h(\theta_m, \sigma(\cdot),\\ &\qquad \mathbf{z}_j(l-g|k), \hat{\mathbf{z}}_{j,m}(k+g|k))\end{aligned} \tag{6}$$

where $\mathbf{z}_j(\cdot)$ denotes the previously measured states of the target vehicle j. Each target vehicles can operate in different modes. All likely modes whose probability $p(m, z)$ is larger than ϵ_p of all target vehicles $j \in [1, n_{\text{vehicle}}]$ are taken as constraints to the controller. The real target mode is determined online (see algorithm and results in Section 3).

The overall model predictive controller is formulated as

$$\min_{\hat{u}(\cdot)} J(\hat{u}(\cdot),\, \hat{x}(\cdot))$$

$$s.t. \quad J(\cdot) = \sum_{i=k}^{k+N-1} \left(\| \hat{u}(i) \|_{R_1}^2 + \| \Delta\hat{u}(i) \|_{R_2}^2\right) + \sum_{i=k+1}^{k+N} \left(\| \hat{x}(i) - x_{\text{ref}}(s) \|_{Q_1}^2 + \| \hat{s}(i) \|_{Q_2}^2\right),$$

$$\hat{x}(k+i) = f(\hat{x}(\cdot),\, u(\cdot)), \quad \hat{x}(k) = x(k), \tag{7a}$$

$$\hat{\mathbf{z}}_{j,m}(k+i) = h(\theta_m,\, \sigma(\cdot),\, \mathbf{z}_j(\cdot),\, \hat{\mathbf{z}}_{j,m}(\cdot)), \tag{7b}$$

$$\Delta u = \hat{u}(k) - \hat{u}(k-1), \quad \Delta u(k) = 0, \tag{7c}$$

$$x(k) \in \mathcal{X} \ominus \mathcal{T}_{j,m}(k), \tag{7d}$$

$$u(k) \in \mathcal{U},\, \Delta u(k) \in \mathcal{U}_\Delta,$$

$$\forall j \in [1, n_{\text{vehicle}}], \quad \forall m \in M \mid p(m, \mathbf{z}_j(\cdot)) \geq \epsilon_p, \quad \forall k$$

where $J(\cdot)$ is the objective function, $d(\cdot)$ is the distance between the predicted states of the target vehicle $\mathbf{z}_{j,m}(\cdot)$ for mode m and predicted states $\hat{x}(\cdot)$ of the ego vehicle. The set $\mathcal{T}_{j,m}(k)$ is defined as

$$\mathcal{T}_{j,m}(k) = \{x \in \mathcal{X} \quad | \quad \|(\hat{z}_{j,m}(k) - x)\|_2 \leq \epsilon_d\} \tag{8}$$

and represents a ball of radius ϵ_d that circumscribed the target vehicle. Note that this set is subtracted from the nominal feasible set (the street borders) using Minkowski subtraction operator $\ominus$. A mode is likely, and therefore considered in the constraints, if the probability $p(\cdot)$ of this mode based on all previously measured target states z is greater or equal a threshold ϵ_p.

The equations (7a) and (7b) describe the dynamics of the ego vehicle j (5) and the target vehicle (6), respectively. The state and input constraints as well as the constraints of the path progress $s(\cdot)$ and the change of input Δu are described by (7c). The path progress $s(k)$ is given by (4) in the optimal control problem (OCP).

This OCP is repeatedly solved at every time step k and only the first part of the optimal control input sequence $\{\hat{u}^*(0), \hat{u}^*(1), ..., \hat{u}^*(N)\}$ is applied to the ego vehicle. At the next time step, the states of the ego vehicle and the target vehicle are updated by measurements, and the OCP is solved again. This enables the controller to react to disturbances and (measurements) noise. It is assumed that all system states $x(k)$ and all target states $z(k)$ can be measured directly and instantaneously at all time steps $k \geq 0$.

The steps of the proposed algorithm can be summarized as follows.

Offline:

(1) Preprocessing and mining of real driving data.
(2) Learning human driving behavior by generating one model (Neural Network) for each mode from real data.
(3) Modelling of the ego vehicle as simple bicycle model.

Online:

(1) Solve the OCP (7) for the ego vehicle (while taking into account all likely modes of the target vehicle.
(2) Classify the trajectory of the target vehicle based on all observed data up to the current time step using nearest neighbourhood comparison.
(3) Remove modes with probability p lower than threshold ϵ_p, from the OCP constraints as these modes are not likely anymore.
(4) Apply first part of optimal control input (solution of OCP (7)).
(5) Repeat from step (1).

6. SIMULATION RESULTS

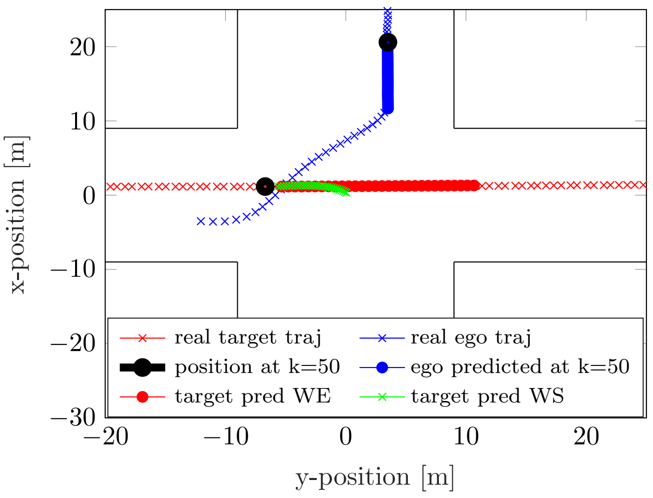

Fig. 7. Intersection with trajectories of the ego (blue) and target (red) vehicle. Crosses denote the real trajectory at the end of the simulation, while dots denote the prediction at time step $k = 50$, i.e. $t = 5$ s. Predictions start at the black circle of the ego (blue) and target vehicle (red), respectively.

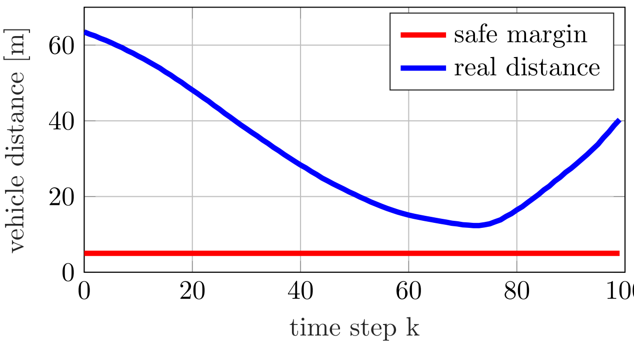

Fig. 8. Safe margin (red) and real distance (blue) between the centers of gravity of ego vehicle and target vehicle, respectively.

Table 1. Minimum distance between ego and target vehicle for 20 trajectories per scenario.

Mode	Straight	Left-Turn	Right-Turn
WE	4.7 m	9.1 m	8.7 m
WS	6.6 m	12.2 m	5.3 m
EW	4.6 m	9.9 m	7.4 m
ES	4.8 m	8.7 m	11.6 m
SW	5.3 m	16.9 m	9.1 m
SE	6.6 m	15.6 m	10.2 m

We consider an autonomous vehicle on a 4-way unsignalized intersection, which may need to give right-of-way to a single human-driven target vehicle. The autonomous

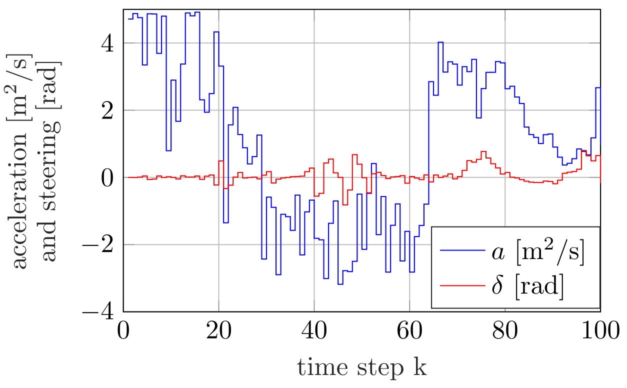

Fig. 9. Input of the ego vehicle, chosen from the MPC with multi-mode constraints and learning based model.

vehicle comes from north and performs a right turn, while the target vehicle goes straight through the intersection from West to East. The left-hand driving setup is shown in Fig. 7 with blue and red dots denoting the ego and target vehicle real trajectory, respectively.

The simulation duration is 25 s with a sampling time of $\Delta t = 0.1\,\text{s}$, while the prediction horizon is $N_H = 30$ steps. The road boundaries are at $x = \{-9\,\text{m}, 9\,\text{m}\}$ and $y = \{-9\,\text{m}, 9\,\text{m}\}$, while the chosen input constraints are $-5\,\frac{\text{m}}{\text{s}^2} < a < 5\,\frac{\text{m}}{\text{s}^2}$ and $-1\,\text{rad} < \delta < 1\,\text{rad}$. The probability threshold is chosen as $\epsilon_p \approx 0.17$.

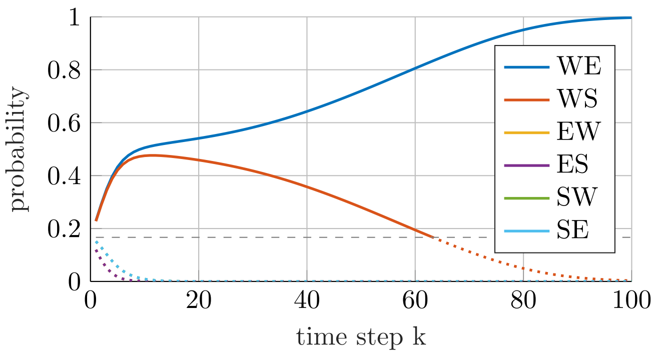

Fig. 10. Probabilities of the modes m with threshold ϵ_p. Active modes are shown as solid lines.

Throughout the first 6.4 s in the simulation (c.f. Fig. 10), two modes of the target vehicle are considered likely and thus both considered as constraints in the OCP.

Furthermore, the desired threshold ϵ_d of the minimum distance between ego and target vehicle depends on the maneuvers, i.e. for direct conflicts $\epsilon_d = 2\,\text{m}$, while for indirect conflicts a bigger threshold $\epsilon_d = 5\,\text{m}$ is chosen. Direct conflicts occur, if the ego vehicle has right-of-way, i.e. the ego vehicle goes straight and the target vehicle is reaching from south. The tuning parameters and weights are chosen as $Q_1 = \text{diag}(2 \cdot 10^{-2}, 2 \cdot 10^{-2}, 0, 1 \cdot 10^{-5})$, $Q_2 = \text{diag}(2 \cdot 10^{-2})$, $R_1 = \text{diag}(1 \cdot 10^{-8}, 0)$ and $R_2 = \text{diag}(1 \cdot 10^{-5}, 1 \cdot 10^{-5})$.

The safe margin is 5 m, which is satisfied for the whole simulation duration. This can be seen in Fig. 8, where the actual minimum distance between the center of gravity of the ego vehicle and the target vehicle is around 12 m. One reason might be that the prediction of the target vehicle position through the neural network is not exactly. Furthermore, the safe distance needs to be ensured for all probable modes. Hence, if a mode becomes unlikely the ego vehicle has less constraint restriction and can accelerate to safely and fast crossing of the intersection. This behavior can be seen in Fig. 9. Furthermore, the target vehicle might be predicted to be slower, which results in a smaller predicted distance and less acceleration of the ego vehicle, cause the ego vehicle gives way to the target vehicle. The prediction of the target vehicle and the ego vehicle positions, respectively, at time step $k = 60$ are also shown in Fig. 7. The current position of both vehicles are drawn with black circles, while the blue, red and green crosses denote the predicted positions of the ego and the two probable target vehicle modes, respectively. The control input for the ego vehicle is shown in Fig. 9, where the blue and red lines denote the acceleration $-5\,\frac{\text{m}}{\text{s}^2} < a < 5\,\frac{\text{m}}{\text{s}^2}$ and the steering angle $-1\,\text{rad} < \delta < 1\,\text{rad}$, respectively. Both control inputs satisfy their constraints. In Fig. 9 it can be seen, that the ego vehicle starting around time step $k = 20$ reduces the acceleration and than decelerates at $k = 29$ to give way to the target vehicle. At $k = 64$ the ego vehicle slowly accelerates again as the target vehicle has entered the intersection at this point, which removes the possibility that the target vehicle stops at the intersection.

The minimum distance between target and ego vehicle for each combination of ego vehicle maneuver and target vehicle mode are shown in Table 1. The minimum distance was determined considering all data points of 20 randomly chosen trajectories of the test set We can see that the minimum distance (2 m) is satisfied for all modes. Furthermore, the optimal control problem was always feasible and the constraints were satisfied at all time steps k.

7. CONCLUSION

We proposed a unifying approach fusing model predictive control and learning of dynamic constraints. The proof-of-concept simulation using a single target vehicle, modeled by offline trained neural networks, showed promising results: the optimal control problem was always feasible and the constraints were satisfied at all time steps k. However, no additional uncertainties, e.g. errors in the prediction of the neural network, are considered in the proposed approach.

REFERENCES

(2018a). Matlab automated driving toolbox 2018a. The MathWorks, Natick, MA, USA.

(2018b). Matlab deep learning toolbox 2018b. The MathWorks, Natick, MA, USA.

Alonso, J., Milanés, V., Pérez, J., Onieva, E., González, C., and De Pedro, T. (2011). Autonomous vehicle control systems for safe crossroads. *Transportation research part C: emerging technologies*, 19(6), 1095–1110.

Bender, A., Ward, J.R., Worrall, S., and Nebot, E.M. (2015). Predicting driver intent from models of naturalistic driving. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, 1609–1615. IEEE.

Bethge, J., Morabito, B., Matschek, J., and Findeisen, R. (2018). Multi-mode learning supported model predictive control with guarantees. *IFAC-PapersOnLine*, 51(20), 517–522.

Borrelli, F., Falcone, P., Keviczky, T., Asgari, J., and Hrovat, D. (2005). Mpc-based approach to active steering for autonomous vehicle systems. *International Journal of Vehicle Autonomous Systems*, 3(2), 265–291.

Dresner, K. and Stone, P. (2004). Multiagent traffic management: A reservation-based intersection control mechanism. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 530–537. IEEE Computer Society.

Dubins, L.E. (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, 79(3), 497–516.

Falcone, P., Eric Tseng, H., Borrelli, F., Asgari, J., and Hrovat, D. (2008). Mpc-based yaw and lateral stabilisation via active front steering and braking. *Vehicle System Dynamics*, 46(S1), 611–628.

Feng, Z. and Zhu, Y. (2016). A survey on trajectory data mining: Techniques and applications. *IEEE Access*, 4, 2056–2067.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Gu, D. and Hu, H. (2002). Neural predictive control for a car-like mobile robot. *Robotics and Autonomous Systems*, 39(2), 73–86.

Guler, S.I., Menendez, M., and Meier, L. (2014). Using connected vehicle technology to improve the efficiency of intersections. *Transportation Research Part C: Emerging Technologies*, 46, 121–131.

He, Z. (2017). Research based on high-fidelity ngsim vehicle trajectory datasets: A review. *Research Gate*, 1–33.

Hsieh, M.F. and Ozguner, U. (2008). A parking algorithm for an autonomous vehicle. In *2008 IEEE Intelligent Vehicles Symposium*, 1155–1160. IEEE.

Kong, J., Pfeiffer, M., Schildbach, G., and Borrelli, F. (2015). Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, 1094–1099. IEEE.

Kruse, R., Borgelt, C., Braune, C., Mostaghim, S., and Steinbrecher, M. (2016). *Computational intelligence: a methodological introduction*. Springer.

Lin, S., Ling, T., and Xi, Y. (2013). Model predictive control for large-scale urban traffic networks with a multi-level hierarchy. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, 211–216. IEEE.

Maas, A.L., Hannun, A.Y., and Ng, A.Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, 3.

Makarem, L. and Gillet, D. (2013). Model predictive coordination of autonomous vehicles crossing intersections. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, 1799–1804. IEEE.

Mitchell, T.M. (1997). *Machine learning*. McGraw Hill Education.

Morabito, B., Kienle, A., Findeisen, R., and Carius, L. (2019). Multi-mode model predictive control and estimation for uncertain biotechnological processes. In *12th International-Federation-of-Automatic-Control (IFAC) Symposium on Dynamics and Control of Process Systems including Biosystems (DYCOPS)*, 709–714. Elsevier.

Park, J., Kim, D., Yoon, Y., Kim, H., and Yi, K. (2009). Obstacle avoidance of autonomous vehicles based on model predictive control. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 223(12), 1499–1516.

Portilla, C., Cortes, L.G., Valencia, F., López, J., Espinosa, J., Núnez, A., and De Schutter, B. (2013). Decentralized model-based predictive control for urban traffic control. In *Proceedings of the 8th Triennial Symposium on Transportation Analysis (TRISTAN VIII), San Pedro de Atacama, Chile.*

Prokop, G. (2001). Modeling human vehicle driving by model predictive online optimization. *Vehicle System Dynamics*, 35(1), 19–53.

Qian, X., Navarro, I., de La Fortelle, A., and Moutarde, F. (2016). Motion planning for urban autonomous driving using bézier curves and mpc. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 826–833. Ieee.

Riegger, L., Carlander, M., Lidander, N., Murgovski, N., and Sjöberg, J. (2016). Centralized mpc for autonomous intersection crossing. In *2016 IEEE 19th international conference on intelligent transportation systems (ITSC)*, 1372–1377. IEEE.

Tan, Y.V., Elliott, M.R., and Flannagan, C.A. (2017). Development of a real-time prediction model of driver behavior at intersections using kinematic time series data. *Accident Analysis & Prevention*, 106, 428–436.

Tettamanti, T., Varga, I., Kulcsár, B., and Bokor, J. (2008). Model predictive control in urban traffic network management. In *2008 16th Mediterranean Conference on Control and Automation*, 1538–1543. IEEE.

Van den Berg, M., De Schutter, B., Hegyi, A., and Hellendoorn, J. (2004). Model predictive control for mixed urban and freeway networks. In *Proceedings of the 83rd Annual Meeting of the Transportation Research Board*, volume 19.

Vasquez, D. and Fraichard, T. (2004). Motion prediction for moving objects: a statistical approach. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 4, 3931–3936. IEEE.

Widyotriatmo, A., Hong, B., and Hong, K.S. (2009). Predictive navigation of an autonomous vehicle with nonholonomic and minimum turning radius constraints. *Journal of Mechanical Science and Technology*, 23(2), 381–388.

Yin, H. and Berger, C. (2017). When to use what data set for your self-driving car algorithm: An overview of publicly available driving datasets. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 1–8. IEEE.

Zheng, Y. (2015). Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3), 29.

Zyner, A.G. (2018). Naturalistic driver intention and path prediction using machine learning.