Coupling Grid-based Stochastic Model Predictive Control with Recurrent Neural Networks for Occupancy Grid Prediction

Diego Martí Monsó and Paul Delseith
Department of Electrical and Computer Engineering, Chair of Automatic Control Engineering (LSR)
Technical University of Munich
Munich, Germany
{diego.marti; paul.delseith}@tum.de

***Abstract*—Recurrent Neural Networks have proven a high ability to predict the future development of dynamic traffic scenarios. For this purpose, grid-based representations, such as Occupancy Grids, are often used to describe the driving environment around an Ego Vehicle. Similarly, Occupancy Grids are also the basis of a number of trajectory planning algorithms for autonomous driving. In this work, we propose a general framework to combine the efficient grid-based Stochastic Model Predictive Control planner with a deep learning grid predictor. We then demonstrate the effectiveness of the approach with two conceptually different Recurrent Neural Networks in simulations of driving scenarios. Furthermore, we suggest a variety of parameters to achieve the desired driving behavior in different situations.**

***Index Terms*—Stochastic Model Predictive Control, Occupancy Grids, Recurrent Neural Networks**

I. Introduction

One of the most challenging, yet crucial, tasks of an autonomous vehicle is to predict the future motion of external road users. Forecasting future traffic events not only increases safety by directly helping to avoid obstacles on the road, but also by facilitating smoother driving paths and thus reducing the forces on the vehicle. In turn, less aggressive accelerations contribute to decreasing energy consumption, tire deterioration, and acoustic contamination, while increasing passenger comfort. Thus, an autonomous vehicle has to understand the scene context and analyze behavioral and motion patterns of other traffic participants to draw hypotheses about the future. However, traffic scenarios typically involve a high degree of uncertainty, due to the deep probabilistic dependencies of traffic events, an almost unlimited amount of possible outcomes, and the irrationality introduced by humans. Hence, trajectory planning algorithms need to cope with the uncertainty inherent to driving environments.

Model Predictive Control (MPC) is a trajectory planning algorithm for automated driving, which repeatedly solves an open-loop optimal control problem at every sampling instant in a receeding horizon manner. With the help of a dynamic system model, a cost function with state and input constraints is minimized over a sequence of control inputs that extends from the current time step to the control horizon. Even though only the first control action of the sequence is applied, making predictions about future time steps allows to anticipate events and avoid overshoot. As the control process advances, updated measurements – acting as a feedback to the MPC – are considered and the horizon recedes. Stochastic Model Predictive Control (SMPC) is an optimal control method that introduces chance constraints to standard MPC to handle the probabilistic incidence of uncertainties. The SMPC approach has proven its effectiveness in planning the trajectories of automated vehicles as it ensures that the risk of collision remains below an adjustable risk parameter β [1], [2], [3]. Nevertheless, the chance constraints significantly augment the computational complexity of the optimal control problem. Therefore, grid-based SMPC has been introduced in [3] to reduce the time cost of the planning algorithm by leveraging on Occupancy Grids (OGs). An OG maps the perceived environment around the controlled entity, which is the Ego Vehicle (EV), into a grid where each cell is assigned an occupancy probability. Additionally, a method is described in [4] to derive object tracks of dynamic road obstacles based on evidential OGs. These object tracks allow a concise description of all dynamic agents within the detection range of the EV, which we denominate as the Target Vehicles (TVs). In practice, any dynamic obstacle on the road can be a TV. However, we restrict the definition solely to vehicles for simplicity.

In SMPC and its variants, the future states of the EV are predicted as the output of a system model, given the future system inputs that are the optimization variable of the stochastic optimization problem. In the previous works of [1], [2], [3], the motion of the TVs is forecast in a similar way, whereas the inputs to the respective TV system models are estimated from probabilistic models and hand-crafted heuristics. However, due to the multimodal nature of driving scenarios and the probabilistic dependencies of traffic events, such prediction models are not able to scale and generalize to complex and diverse scenarios. Conversely, Recurrent Neural Networks (RNNs) have shown an outstanding performance to forecast the future development of grid driving environments by inference from previous measurements [5], [6], [7].

For instance, a difference learning RNN model is presented

in [7] that shows a high degree of accuracy. In addition, the convolutional network to forecast future OGs in urban driving scenarios proposed in [8] is outperformed by the authors with an RNN network in [6], which takes OGs sequentially as inputs and produces separate static and dynamic predictions. A probabilistic RNN model to predict future vehicle positions is introduced in [5]. In contrast to the previously discussed works, the data used in the model of [5] is not in the format of OGs, but as state vectors that describe the motion of individual vehicles. However, the network is provided with little contextual information about other traffic participants for the inference, which severly limits the scalability of the approach, especially when interactions between vehicles play an important role.

In this work, we propose a framework to couple grid-based SMPC with RNN-based predictions of OGs. In [3], the stochastic uncertainty is considered in the TV vehicle model used for inference. We substitute the TV system model by different RNN prediction models and suggest corresponding methodologies to account for the prediction uncertainty in the SMPC Optimal Control Problem (OCP).

A. Report structure

The remainder of this work is structured as follows: in Section II, we first explore the preliminaries, then we present the EV system model used in the OCP and the later simulation, and finally we introduce grid-based SMPC. Next, Section III opens with the characteristics of the training data used to train the prediction models, followed by the two grid prediction models explored in this work, and the method is completed with the procedure to derive a linear constraint from the predicted OGs. In Section III, we explain the simulation setup and the driving scenario used for evaluation, and then we analyze and discuss the results. Lastly, Section V is the conclusion.

II. Problem Setup

In this section, we briefly present OGs and a special case of the RNN as the preliminaries to the rest of the report. Then, we introduce the EV system model and the grid-based SMPC control method, where we give a low-level description of the problem setup.

A. Preliminaries

1) Occupancy Grid: An OG is a two-dimensional, space-discrete representation of the local environment that provides a rectangular shaped bird's-eye view of the setting with length l_{env} and width w_{env}. The surrounding of the EV is divided into a grid $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{\frac{l_{\text{env}}}{l_{\text{cell}}} \times \frac{w_{\text{env}}}{w_{\text{cell}}}}$ of cells $c_{i,j} \in \boldsymbol{\mathcal{G}}$ of length l_{cell} and width w_{cell}, where the subscripts x and y denote the two spatial dimensions, and i and j are indices. Additionally, the fractions $\frac{l_{\text{env}}}{l_{\text{cell}}}$ and $\frac{w_{\text{env}}}{w_{\text{cell}}}$ are required to be integers. The size of the cells, which can be seen as the resolution of the grid, poses a trade-off between accuracy and computational cost. Each of the cells is assigned an occupancy value that approximates the posterior probability of the presence of an object. Different algorithmic approaches exist to estimate the occupancy values, typically as Bayesian occupancy estimations of inexact fused sensor measurements. In addition, variants of the standard OG may include diverse channels to code supplementary information, such as dynamic estimates.

2) Long Short-Term Memory: The RNN is an artificial neural network that is able to learn to process sequential data, such as time series data [9]. At every time step k, the hidden state $\boldsymbol{h}_k$ of the standard RNN cell is fed back to its input in the following time step $k+1$. In this way, information is carried on sequentially through time and the output of the cell is influenced by the previous events. However, the RNN suffers from the vanishing gradient problem during training, which prevents the network from learning long-term dependencies in the input sequence.

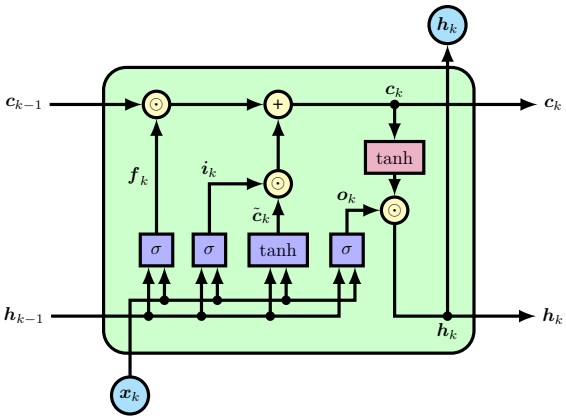


Fig. 1: Internal structure of a standard LSTM cell. Operators are colored yellow, internal functions purple, activation functions blue, and external inputs and outputs cyan. Weight matrices and bias vectors have been left out.

To solve this issue, the Long Short-Term Memory (LSTM) complements the standard RNN with an additional cell memory $\boldsymbol{c}_k$ and a gating mechanism that controls the information flow within the cell according to the following recursive equations:

$$\boldsymbol{f}_k = \sigma\left(\boldsymbol{W}_{fh}\boldsymbol{h}_{k-1} + \boldsymbol{W}_{fx}\boldsymbol{x}_k + \boldsymbol{b}_f\right), \tag{1a}$$
$$\boldsymbol{i}_k = \sigma\left(\boldsymbol{W}_{ih}\boldsymbol{h}_{k-1} + \boldsymbol{W}_{ix}\boldsymbol{x}_k + \boldsymbol{b}_i\right), \tag{1b}$$
$$\boldsymbol{o}_k = \sigma\left(\boldsymbol{W}_{oh}\boldsymbol{h}_{k-1} + \boldsymbol{W}_{ox}\boldsymbol{x}_k + \boldsymbol{b}_o\right), \tag{1c}$$
$$\tilde{\boldsymbol{c}}_k = \tanh\left(\boldsymbol{W}_{ch}\boldsymbol{h}_{k-1} + \boldsymbol{W}_{cx}\boldsymbol{x}_k + \boldsymbol{b}_c\right), \tag{1d}$$
$$\boldsymbol{c}_k = \boldsymbol{f}_k \odot \boldsymbol{c}_{k-1} + \boldsymbol{i}_k \odot \tilde{\boldsymbol{c}}_k, \tag{1e}$$
$$\boldsymbol{h}_k = \boldsymbol{o}_k \odot \tanh\left(\boldsymbol{c}_k\right), \tag{1f}$$

where the operator $\odot$ denotes the Hadamard product (i.e., the element-wise multiplication of vectors), the hyperbolic tangent $\tanh : \mathbb{R}^\eta \to (-1,1)^\eta$ is applied element-wise, and the element-wise sigmoid logistic function $\sigma_{\text{sigmoid}} : \mathbb{R}^\eta \to (0,1)^\eta$ is used as an activation function:

$$\sigma_{\text{sigmoid}}(\boldsymbol{v})_j = \frac{1}{1+e^{-v_j}}, \tag{2}$$

for $j = 1, \ldots, \eta$ and $\boldsymbol{v} = [v_1, \ldots, v_\eta]^\top \in \mathbb{R}^\eta$. Furthermore, the input $\boldsymbol{x}_k \in \mathbb{R}^\delta$ at time step k is a vector of δ input features, the bias vectors $\boldsymbol{b}_f, \boldsymbol{b}_i, \boldsymbol{b}_o, \boldsymbol{b}_c \in \mathbb{R}^\eta$ have length η, and $\boldsymbol{W}_{fh}, \boldsymbol{W}_{ih}, \boldsymbol{W}_{oh}, \boldsymbol{W}_{ch} \in \mathbb{R}^{\eta \times \eta}$ and $\boldsymbol{W}_{fx}, \boldsymbol{W}_{ix}, \boldsymbol{W}_{ox}, \boldsymbol{W}_{cx} \in \mathbb{R}^{\eta \times \delta}$ are weight matrices. The cell memory $\boldsymbol{c}_k \in \mathbb{R}^\eta$ at time step k is updated in (1e) by applying the forget gate $\boldsymbol{f}_k \in (0,1)^\eta$ from (1a) to the previous cell memory $\boldsymbol{c}_{k-1}$, and the input gate $\boldsymbol{i}_k \in (0,1)^\eta$ from (1b) to the new cell memory candidate $\tilde{\boldsymbol{c}}_k \in (-1,1)^\eta$ from (1d). The forget gate decides about the information that will no longer be tracked in the cell memory, while the input gate chooses and scales the elements of the new cell memory candidate that will be taken over. Next, the hidden state $\boldsymbol{h}_k \in (-1,1)^\eta$ at time step k is obtained in (1f) by employing the output gate $\boldsymbol{o}_k \in (0,1)^\eta$ from (1c) on the updated cell memory, with its range limited to $(-1,1)^\eta$ by the element-wise hyperbolic tangent. The flow of information within the LSTM cell is represented in Figure 1.

Overall, the LSTM cell has an increased memory compared to the standard RNN cell. In consequence, LSTMs are often used as the core block in long-term deep learning prediction models, which operate in a sequence-to-sequence fashion.

B. Ego Vehicle Model

In this work, we employ a kinematic bicycle model, as presented in [2], to describe the dynamics of the EV. The nonlinear system is given by

$$\dot{\boldsymbol{\xi}}^{\mathrm{EV}} = \boldsymbol{f}\left(\boldsymbol{\xi}^{\mathrm{EV}}, \boldsymbol{u}\right), \tag{3}$$

where $\boldsymbol{\xi}^{\mathrm{EV}} = [x, v, y, \psi]^\top$ represents the state vector of the EV and $\boldsymbol{u} = [\delta, a]^\top$ is the input vector. The state consists of the longitudinal position x, the velocity v, the lateral position y, and the yaw angle ψ with respect to the road. Likewise, the input vector is composed by the steering angle δ at the front left and right wheels, and the acceleration a. As the OCP solved in MPC is discrete-time, we additionally discretize the system equations by a Forward Euler scheme with the sampling time T, which yields the system difference equations

$$x_{k+1} = x_k + \frac{v_k T}{1 - y_k \kappa_k(x_k)} \cos\left(\psi_k + \alpha_k\right), \tag{4a}$$

$$v_{k+1} = v_k + a_k T, \tag{4b}$$

$$y_{k+1} = y_k + v_k T \sin\left(\psi_k + \alpha_k\right), \tag{4c}$$

$$\psi_{k+1} = \psi_k + \frac{v_k}{l_\mathrm{r}} \sin\left(\alpha_k\right) - \frac{\kappa_k(x_k) v_k T}{1 - y_k \kappa_k(x_k)} \cos\left(\psi_k + \alpha_k\right), \tag{4d}$$

$$\alpha_k = \arctan\left(\frac{l_\mathrm{r}}{l_\mathrm{f} + l_\mathrm{r}} \tan\left(\delta_k\right)\right), \tag{4e}$$

with the discrete states $\boldsymbol{\xi}_k^{\mathrm{EV}} = [x_k, v_k, y_k, \psi_k]^\top$ and inputs $\boldsymbol{u}_k = [\delta_k, a_k]^\top$ at prediction step k, and the distances l_r and l_f from the center of gravity of the vehicle to the rear and front axles, respectively. The discretized EV prediction model in (4) is encapsulated in $\boldsymbol{\xi}_{k+1}^{\mathrm{EV}} = \boldsymbol{f}\left(\boldsymbol{\xi}_k^{\mathrm{EV}}, \boldsymbol{u}_k\right)$.

At all prediction steps k, the input vector is subject to constraints

$$\boldsymbol{u}_{\min} \leq \boldsymbol{u}_k \leq \boldsymbol{u}_{\max} \tag{5}$$

with $\boldsymbol{u}_{\min} = [\delta_{\min}, a_{\min}]^\top$ and $\boldsymbol{u}_{\max} = [\delta_{\max}, a_{\max}]^\top$. These boundaries on the steering angle and the acceleration are determined by vehicle parameters. Similarly, the lateral position and the velocity of the EV are also restricted by the road and speed limits

$$y_{\min} \leq y_k \leq y_{\max}, \tag{6a}$$

$$v_{\min} \leq v_k \leq v_{\max}. \tag{6b}$$

Finally, the input and state constraints (5) and (6) are summarized in the set of admissible inputs $\mathcal{U}$ and the set of admissible states Ξ at prediction step k.

C. Grid-based Stochastic Model Predictive Control

The SMPC OCP considers a chance constraint to allow a certain probability level of constraint violation. Since the traffic environment around the EV is highly uncertain, it has proven effective to model this stochastic uncertainty with SMPC chance constraints [1]. Nevertheless, chance constraints introduce an additional computational complexity that makes the OCP intractable. Therefore, grid-based SMPC with OGs was proposed in [3] to transform chance constraints into a linear, deterministic constraint that reduces the computational cost of the method. Effectively, the OCP solved in grid-based SMPC is thus a standard MPC OCP.

We initialize the OCP with the estimated EV state $\hat{\boldsymbol{\xi}}_t^{\mathrm{EV}}$ at sampling step t, which is subject to sensor noise in the measurements

$$\hat{\boldsymbol{\xi}}_t^{\mathrm{EV}} = \boldsymbol{\xi}_t^{\mathrm{EV}} + \boldsymbol{w}^{\mathrm{EV}}, \tag{7}$$

where $\boldsymbol{w}^{\mathrm{EV}} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_w^{\mathrm{EV}})$ is a normally distributed, zero-mean random variable with covariance matrix $\boldsymbol{\Sigma}_w^{\mathrm{EV}}$. For simplicity, in the remainder of this work, only the index k, corresponding to the prediction step, is given unless explicitly indicated otherwise.

With all the above considerations, the grid-based SMPC OCP is given by

$$\boldsymbol{U}^* = \arg\min_{\boldsymbol{U}} \sum_{k=0}^{N-1} l\left(\boldsymbol{\xi}_k^{\mathrm{EV}}, \boldsymbol{u}_{k-1}, \boldsymbol{u}_k\right) + J_\mathrm{f}\left(\boldsymbol{\xi}_k^{\mathrm{EV}}\right) \tag{8a}$$

$$\text{s.t.} \quad \boldsymbol{\xi}_0^{\mathrm{EV}} = \hat{\boldsymbol{\xi}}_t^{\mathrm{EV}}, \tag{8b}$$

$$\boldsymbol{\xi}_{k+1}^{\mathrm{EV}} = \boldsymbol{f}\left(\boldsymbol{\xi}_k^{\mathrm{EV}}, \boldsymbol{u}_k\right), \quad k \in \mathbb{N}, \tag{8c}$$

$$\boldsymbol{\xi}_k^{\mathrm{EV}} \in \Xi_k, \qquad k = 1, \ldots, N, \tag{8d}$$

$$\boldsymbol{u}_k \in \mathcal{U}_k, \qquad k = 0, \ldots, N-1, \tag{8e}$$

$$\boldsymbol{A}_k^{\text{safe}} \boldsymbol{\xi}_k^{\mathrm{EV}} \leq \boldsymbol{b}_k^{\text{safe}}, \qquad h = 1, \ldots, N, \tag{8f}$$

with the optimal control sequence $\boldsymbol{U}^* = \left[\boldsymbol{u}_0^*, \ldots, \boldsymbol{u}_{N-1}^*\right]$, which minimizes the cost function over the control input $\boldsymbol{U} = [\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{N-1}]$, the prediction step k, the prediction horizon $N \in \mathbb{N}$, the estimated EV state $\hat{\boldsymbol{\xi}}_t^{\mathrm{EV}}$ at sampling step t, the system dynamics $\boldsymbol{f}$, the state and input constraint sets

Ξ and $\mathcal{U}$ respectively, and the safe space matrix $\boldsymbol{A}^{\text{safe}}$ and vector $\boldsymbol{b}^{\text{safe}}$. The cost function is made up of the stage cost $l\left(\boldsymbol{\xi}_k^{\text{EV}}, \boldsymbol{u}_{k-1}, \boldsymbol{u}_k\right)$ and the terminal cost $J_{\text{f}}\left(\boldsymbol{\xi}_k^{\text{EV}}\right)$, which are given by

$$l\left(\boldsymbol{\xi}_k^{\text{EV}}, \boldsymbol{u}_{k-1}, \boldsymbol{u}_k\right) = \left\|\boldsymbol{\Delta}\boldsymbol{\xi}_k^{\text{EV}}\right\|_{\boldsymbol{Q}}^2 + \left\|\boldsymbol{u}_k\right\|_{\boldsymbol{R}}^2 + \left\|\boldsymbol{\Delta}\boldsymbol{u}_k\right\|_{\boldsymbol{S}}^2, \quad (9a)$$

$$J_{\text{f}}\left(\boldsymbol{\xi}_k^{\text{EV}}\right) = \left\|\boldsymbol{\Delta}\boldsymbol{\xi}_N^{\text{EV}}\right\|_{\boldsymbol{Q}}^2, \quad (9b)$$

with the operator $\|\boldsymbol{z}\|_{\boldsymbol{W}}^2 = \boldsymbol{z}^\top \boldsymbol{W} \boldsymbol{z}$ as the squared norm of $\boldsymbol{z}$ w.r.t. metric $\boldsymbol{W}$, the deviation from the reference state $\boldsymbol{\Delta}\boldsymbol{\xi}_k^{\text{EV}} = \boldsymbol{\xi}_k^{\text{EV}} - \boldsymbol{\xi}_{k,\text{ref}}^{\text{EV}}$ with the EV reference $\boldsymbol{\xi}_{k,\text{ref}}^{\text{EV}}$, the input difference $\boldsymbol{\Delta}\boldsymbol{u}_k = \boldsymbol{u}_k - \boldsymbol{u}_{k-1}$, and the weighting matrices $\boldsymbol{Q} \in \mathbb{R}^{4\times4}$ and $\boldsymbol{R}, \boldsymbol{S} \in \mathbb{R}^{2\times2}$. For the input difference $\boldsymbol{\Delta}\boldsymbol{u}_k$, the input $\boldsymbol{u}_0$ for prediction step $k = 0$ is initialized as $\boldsymbol{u}_0 = [0, 0]^\top$.

Constraint (8f) describes the maximal convex subset around the EV within the space of admissible grid cells, i.e., the grid cells that are believed to be free of obstacles for a given prediction step k. Consequently, all information about external road obstacles and dynamic objects considered in the OCP is included in (8f). Hence, our goal is to develop multi-step and long-term prediction models to derive the admissible space at every prediction step k as the linear inequality constraint (8f). Given the OCP (8), a total of N predictions with a sampling time T are needed. Because of their previous success in OG prediction tasks, we explore RNN-based forecasting models with the same prediction horizon N and sampling time T as the grid-based SMPC framework.

III. Method

This section consists of three parts. First, we present the training data set and the data acquisition method from microscopic traffic simulations. Second, we build two different RNN-based multi-step forecasting models and train them to predict the sequence of N future OGs, which describes the future motion of all TVs in the surroundings of the EV. Third, we regard the stochastic prediction uncertainty to derive the admissible space as the linear inequality constraint (8f).

All prediction models treat the forecasting problem as a sequence-to-sequence prediction task. The models are initialized with the sequence of previous M observations, containing information of the TVs in the local environment of the EV in the M last measurement steps. We choose the initialization length M to be equal to the prediction horizon N, which is set to 20 (i.e., $M = N = 20$). Similarly, we also use the same sampling time T for the observation sequence as for the prediction sequence, which is established as $T = 0.2\,\text{s}$.

A. Training Data

We generate training data for the deep learning prediction models by sampling traffic simulations in the microscopic traffic simulator SUMO (Simulation of Urban MObility) [10], which models individual vehicles and their interactions at a microscopic level in a space-continuous, time-discrete setting. The simulation scenarios are highly customizable and different dynamic, behavioral, and probabilistic models are available to configure the environment and traffic.

We record 62 independent traffic scenarios that take place in the same two-lane highway setting as the later simulation setup for the evaluation in Section IV, where the parameters of the road layout can be found. The registered scenarios vary in the amount, type, and initial conditions of traffic participants. Overall, the scenarios are intended to imitate and cover most of the situations that can be observed in real driving circumstances, such as different ranges of road congestion, partial obstruction of roadways, diverse uses of lanes, individual and group driving behavior patterns, and varying TV characteristics.

Each driving scenario is scanned in an egocentric view, where all the entities within the local environment of the EV are sampled. The local environment is bounded by the detection range of $100\,\text{m}$ from the center of gravity of the EV, which results in the environment length $l_{\text{env}} = 2 \times 100\,\text{m} = 200\,\text{m}$. As the width of the two lanes on the highway is $3.5\,\text{m}$, the width of the sampled environment is $w_{\text{env}} = 2 \times 3.5\,\text{m} = 7\,\text{m}$. The size of the grid cells is set to $l_{\text{cell}} = 0.5\,\text{m}$ and $w_{\text{cell}} = 0.25\,\text{m}$, yielding a rectangular grid with the shape $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{400\times28}$. During the simulation, objects on the road may enter or leave the detection zone freely. Tracking the local environment of a particular vehicle instead of simulating a stationary sensor is advantageous considering that the EV can only measure in an egocentric reference frame as well. We process the raw position, velocity, and orientation measurements from the simulation into an OG of binary values, describing minimum bounding boxes as the output of the object tracking algorithm in [4], in MATLAB®. However, we do not consider any sensor noise in the measurements used to construct the training data sets, as we aim to train prediction models with a high accuracy.

Every scenario is recorded for $72\,\text{s}$ at a sampling rate of $\frac{1}{T} = 5\,\text{Hz}$, which yields a total of $62 \times 72\,\text{s} = 4464\,\text{s}$ of collected data. The 62 sequences are then partitioned into 9 segments of $(M + N)\,T = 8\,\text{s}$ each, where the first $M \cdot T = 4\,\text{s}$ are used to initialize the prediction model and the remaining $N \cdot T = 4\,\text{s}$ are employed as the ground truth labels during training. The total $62 \times 9 = 558$ sequence samples are further split randomly into a training data set, which contains 80% of the sequence samples (447 samples), a validation data set, composed by 10% of the sequence samples (56 samples), and a test data set with the remaining 10% (55 samples).

B. Multi-step Grid Prediction Models

The input to the SMPC controller is a series of N OGs that indicate the locations of vehicles for N time steps into the future. One approach is to train a neural network to directly output these OGs based on a time series of OGs as an input. At the core this is a computer vision problem. The network needs to detect the vehicles in the input sequence purely based on visual data, such that it can subsequently predict their behavior in the future. As explained in II-A, RNNs using LSTM cells are a prominent method for time

series understanding. However, the standard LSTM cell is not well suited for image data, as it does not take advantage of the spatial dependencies of an image. In [11], Shi et al. propose an extension to the LSTM, the convolutional Long Short Term memory (ConvLSTM). ConvLSTMs conduct convolution operations at the input and hidden state terminals. This enables understanding of local spatio-temporal correlations in the input image, which is crucial for identifying objects and their movement in a sequence of images. In this work we present two neural network architectures based on ConvLSTMs, similar to [6]. We now present a brief overview of the implemented NN architectures.

1) The OGs derived from the perception system have a high degree of regularity, the vehicles, although of different sizes, all have a rectangular shape. Furthermore, in all scenarios, a significant amount of the image is black and there is always a high contrast between the vehicles and the street. Therefore, at the input of the network we embed the OGs into a different feature space. In computer vision this is normally done using convolutional layers. As described by LeCun et al. in [12], they can filter out different aspects of the image, while retaining spatial relations. In the proposed network, there are two convolutional layers that project the images into a latent feature space. The input of the network is a tensor with shape $\mathbb{R}^{M\times 400\times 28\times 1}$. The two convolutional layers project the input images from the shape $\mathbb{R}^{400\times 28\times 1}$ to $\mathbb{R}^{200\times 28\times 4}$. Convolutions are applied to each time-step separately so that the resulting output is $\mathbb{R}^{M\times 400\times 28\times 4}$. In the next stage there are two layers of bidirectional ConvLSTMs. The first layer takes an input tensor of size $\mathbb{R}^{M\times 200\times 28\times 4}$ and outputs a tensor of size $\mathbb{R}^{M\times 200\times 28\times 10}$, which serves as an input to the second ConvLSTM. The second ConvLSTM layer not only takes the output and hidden states of the first ConvLSTM layer, but also its internal state. Therefore, the second layer has access to an encoded representation of the input sequence and, based on that, can now predict the next N time steps into the future. The output tensor of the second layer is again a tensor of the size $\mathbb{R}^{N\times 200\times 28\times 10}$, which then feeds into a transposed convolutional layer, which reconstructs the initial shape of the OGs and therefore outputs a tensor of size $\mathbb{R}^{N\times 400\times 28\times 1}$. This output represents a prediction for the next N time steps into the future.

2) The second network has a very similar structure to the first. Again, the input is a tensor of shape $\mathbb{R}^{M\times 400\times 28\times 1}$, which is projected onto $\mathbb{R}^{M\times 100\times 28\times 8}$ by the first two convolutional layers. The output is then fed into a bidirectional ConvLSTM that encodes the input sequence. Importantly, the first LSTM layer does not share its internal state with the second layer. The output of the second layer is a tensor of size $\mathbb{R}^{M\times 100\times 28\times 10}$, which then again is fed into the transposed convolution layer to get an output of size $\mathbb{R}^{M\times 400\times 28\times 1}$. In contrast to the first network only the last element of this tensor is treated as the output of the network. Therefore, inference has to be repeated N times, with an updated input sequence, in every iteration.

For regularization and Monte Carlo Dropout (explained in the next section), we introduce a 2D dropout layer after the first and second layer as well as after the second ConvLSTM layer.

Since the data format is an OG of binary values, the network output has to be restricted to the interval $[0,1]^R$, where R is the number of responses. Therefore, we apply a clipped Rectified Linear Unit (ReLU) $\sigma_{\text{cReLU}}(\boldsymbol{x}) : \mathbb{R}^R \to [0,1]^R$ to the vector of activations $\boldsymbol{x} = [x_1, \dots, x_R]^\top \in \mathbb{R}^R$ of the last layer, where we define the element-wise clipped ReLU as

$$\sigma_{\text{cReLU}}(\boldsymbol{x})_r = \begin{cases} 0, & \text{if } x_r < 0, \\ x_r, & \text{if } 0 \leq x_r < 1, \\ 1, & \text{if } x_r \geq 1, \end{cases} \tag{10}$$

where $r = 1, \dots, R$ is the r-th response.

Network 1 (III-B) is trained with an input sequence of length M, the target sequence are the N subsequent time steps. Model 2 (III-B) is trained with an input sequence of length M to predict a target sequence that is shifted by one time step into the future. Both networks are trained using the Adam [13] solver. Adam performs a stochastic optimization of the network parameters based on an error measure given by the cost function $\mathcal{L}$. Model hyperparameters are tuned using the iterative Hyperband tuning algorithm [14], which is based on random search of the parameter space. Compared to random search, Hyperband tuning provides a speedup, as it employs early stopping and first trains a large set of models on a small amount of epochs and then iteratively trains the best performing models on an increasing amount of epochs. Net 1 and Net 2 are then trained a final time with the best performing set of hyperparameters. The training is performed on 80 epochs, with a mini-batch size of 5 and an initial learning rate of 0.00294 for Net 1 and 0.00467 for Net 2. We use early stopping to terminate the training whenever the generalization performance does not further improve, which we consider if the validation loss does not decrease for 7 consecutive epochs. As loss function we use the mean-squared-error loss $\mathcal{L}_{\text{regr}}$, given by

$$\mathcal{L}_{\text{regr}} = \frac{1}{MR} \sum_{m=1}^{M} \sum_{r=1}^{R} (y_{m,r} - y^*_{m,r})^2 \tag{11}$$

where R is the number of cells in an OG, in our case $R = 11200$. Therefore, $y_{m,r}$ is the predicted, $y^*_{m,r}$ the ground truth, r-th pixel in time step m.

C. Chance Constraint Reformulation

In grid-based SMPC, we consider the environment predictions to be uncertain. However, the forecasted OGs are deterministic and do not include any information about the model uncertainty. Therefore, a stochastic description of the local environment that additionally models prediction uncertainty is required. We denominate such a data structure as a Probability Grid (PG). The PG consists of a grid $\boldsymbol{\mathcal{P}} \in \mathbb{R}^{\frac{l_{\text{env}}}{l_{\text{cell}}} \times \frac{w_{\text{env}}}{w_{\text{cell}}}}$ of occupancy values $p_{i,j} \in \boldsymbol{\mathcal{P}}$ that describe the estimated likelihood of the cell being occupied. To find the values of the PG, we need to process the raw outputs of the neural networks to model the uncertainty of the grid predictions at

every prediction step k. There are only few options to obtain a measure of prediction uncertainty from neural networks.

Bayesian neural networks (BNNs) were specifically invented for estimating prediction uncertainty [15]. However, the implementation of a BNN is significant overhead as compared to a standard neural network. Another method for estimating model uncertainty is the so called Monte Carlo Dropout (MCD) [16]. The idea behind this technique is to use dropout layers, not only during training, but also in inference to randomly change the connections in the network. Therefore, the MCD introduces a stochastic disturbance into the predictions. By sampling multiple predictions for the same input, an estimate for prediction accuracy, as well as an average prediction can be computed. In [16], Gal et al. demonstrate that this approach for inference approximates Bayesian inference in deep Gaussian processes and support their thesis with a multitude of examples where uncertainty estimations improve model performance. A major benefit of this approach is that by averaging model predictions, we can directly derive predictions that model uncertainty and can thus be used as a PG in grid-based SMPC. For instance, regions where the grid prediction model consistently predicts the same deterministic value (i.e., 0 or 1), are unaffected by the averaging process. Conversely, areas in which the prediction model assigns different values each time, are noisy after the averaging, signaling prediction uncertainty. Hence, in this work we opted, for implementing MCD in the presented neural network architectures.

After the computation of the PG, to stochastically model the territory around the EV at prediction step k, the SMPC chance constraint can be reformulated as a linear constraint. We contemplate the following chance constraint on the set of admissible EV states Ξ_k:

$$\Pr\left(\boldsymbol{\xi}_k^{\mathrm{EV}} \in \Xi_k\right) \geq \beta. \tag{12}$$

First, a threshold p_{th} is applied to the grid $\boldsymbol{\mathcal{P}}$ to turn it into a Binary Grid (BG), represented by $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{\frac{l_{\mathrm{env}}}{l_{\mathrm{cell}}} \times \frac{w_{\mathrm{env}}}{w_{\mathrm{cell}}}}$, of binary values

$$b_{i,j} = \begin{cases} 1, & \text{if } p_{i,j} \geq p_{\mathrm{th}}, \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

The threshold parameter p_{th} is tunable and allows to adjust the risk aversion of the controller: similarly to the probability level β, the control behavior becomes more conservative the higher the risk factor p_{th}. In essence, the BG divides the space into two mutually exclusive and collectively exhaustive subsets of inadmissible or occupied cells, where $b_{i,j} = 1$, and admissible or free cells, which hold the value 0. This way, the chance constraint (12) can be transformed into a hard state constraint

$$\boldsymbol{\xi}_k^{\mathrm{EV}} \in \Xi_k^{\mathrm{adm}}, \tag{14}$$

where Ξ_k^{adm} contains all admissible cells in $\boldsymbol{\mathcal{B}}$ and symbolizes the admissible space.

Next, a linear subset of (14) can be found to further improve the performance of the SMPC scheme. In this sense, a method that relies on Bresenham's line algorithm is proposed in [3] to find the maximal convex subset within the admissible space. Consequently, the convex space extracted from Ξ_k^{adm} can be expressed with a linear inequality, given by

$$\boldsymbol{A}_k^{\mathrm{safe}} \boldsymbol{\xi}_k^{\mathrm{EV}} \leq \boldsymbol{b}_k^{\mathrm{safe}}, \tag{15}$$

with $\boldsymbol{A}_k^{\mathrm{safe}} \in \mathbb{R}^{4\times 4}$ and $\boldsymbol{b}_k^{\mathrm{safe}} \in \mathbb{R}^4$. The resulting constraint is tractable and reduces the computational cost of the SMPC trajectory planning method significantly, as this procedure is repeated at every prediction step k. Finally, (15) is used as constraint (8f) in the grid-based SMPC OCP, thus completing the proposed method

IV. Results

In this section, we evaluate the proposed control framework. First, we present the general simulation setup and the driving scenario where the controller is tested. Then, we analyze the performance of the different grid prediction models and explore the performance of the closed SMPC control loop in the driving scenario. Lastly, the results are discussed at the end of the section.

A. Simulation Setup

The proposed grid-based SMPC method with an integrated deep learning grid prediction module, which can either be executed as Model 1 or Model 2 from Section III-B, is evaluated in a simulation framework developed in MATLAB ®, Python, and SUMO. Figure 2 depicts the outline of the general simulation framework.

A simulation of a road scenario is conducted in SUMO, where the interactions between the traffic participants can be evaluated step-wise at a microscopic level, meaning that the dynamics of every single vehicle are modeled individually [10]. The EV is controlled externally from a MATLAB ® client, where SUMO is run as a server. The computed optimal inputs $\boldsymbol{u}_0^*$ are applied to the simulation with the discretized EV system model presented in Section 4 as $\boldsymbol{\xi}_{t+1}^{\mathrm{EV}} = \boldsymbol{f}\left(\boldsymbol{\xi}_t^{\mathrm{EV}}, \boldsymbol{u}_0^*\right)$. Then, the EV state $\boldsymbol{\xi}_{t+1}^{\mathrm{EV}}$ for simulation step $t+1$ is updated in SUMO and the traffic simulation advances one step to simulation instance $t+1$. There, SUMO produces the states $\boldsymbol{\xi}_{t+1}^{\mathrm{TV},\tau}$ of all $\tau = 1, \ldots \mathcal{T}$ TVs in the simulation scenario, simulating real-life driving behaviors and interactions.

Next, the states of the TVs are measured and an OG is constructed. While we directly obtain the data from the simulation, we simulate the process that would occur in the dynamic object tracking approach from [4]. Then, the obtained OG is used in the evidence sequence containing M observations used to initialize the grid prediction models.

Subsequently, one of the two presented multi-step grid prediction methods is used to predict the future motion of the dynamic objects in the surroundings of the EV for prediction steps $k = 1, \ldots, N$. The PG sequence $\{\boldsymbol{\mathcal{P}}_1, \ldots, \boldsymbol{\mathcal{P}}_N\}$ is derived from the forecasts to be used in the grid-based SMPC trajectory planning scheme. The predictors are implemented,

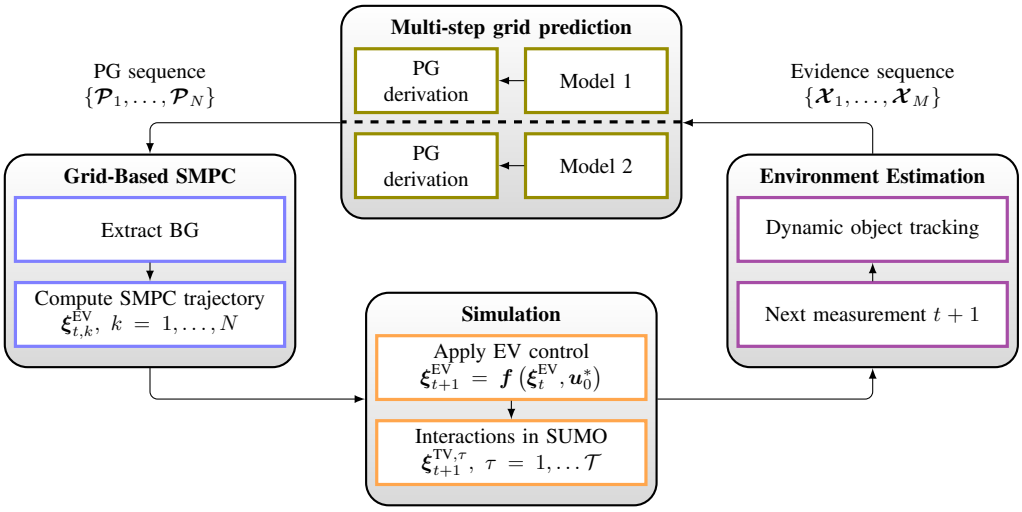


Fig. 2: Simulation framework outline. Starting at the container at the bottom, a simulation of a traffic scenario is conducted in SUMO, where the EV is controlled externally. At each simulation step, a sequence of past observations is constructed in the environment estimation module (container to the right) from the information in the simulation. In the upper container, the multi-step grid prediction module produces a sequence of PGs either with a classification or a regression network. The PGs are then used in the grid-based SMPC (container to the left) to calculate the optimal trajectory.

TABLE I: Simulation parameters.

scalars	vectors	matrices
w_{lane}=3.5	$\boldsymbol{u}_{\min} = [-3, -5]^\top$	$\boldsymbol{Q} = \text{diag}(0, 0.25, 0.2, 10)$
$l_{\text{EV}} = 4.3$	$\boldsymbol{u}_{\max} = [3, 5]^\top$	$\boldsymbol{R} = \text{diag}(5, 0.33)$
$w_{\text{EV}} = 1.8$		$\boldsymbol{S} = \text{diag}(15, 0.33)$
$l_{\text{f}} = l_{\text{r}} = 1.8$		
$v_{\min} = 0$		
$v_{\max} = 30.55$		
$y_{\min} = 1$		
$y_{\max} = 6$		

TABLE II: Vehicle states at the start of the simulation. We consider the longitudinal position x, the lateral position y, the longitudinal velocity v_x, and the lateral velocity v_y.

Vehicle	x	y	v_x	v_y
EV	82.28	1.75	26.3	0
TV1	112.50	5.25	27	0
TV2	139.30	1.75	24	0
TV3	155.82	5.25	28	0

trained, and evaluated in Python, using the popular model-level library Keras [17] with the TensorFlow [18] backend.

The relevant simulation parameters are shown in Table I.

B. Driving Scenario

We design a two-lane highway scenario with an initial configuration that is not found in the training data. Since we first need to generate a sequence of $M = 20$ OGs to initialize the grid prediction models, we let the simulation run controlled by SUMO for 20 steps before we start the control framework. At this time instant, the initial setup of the the vehicles on the grid can be found in Table II.

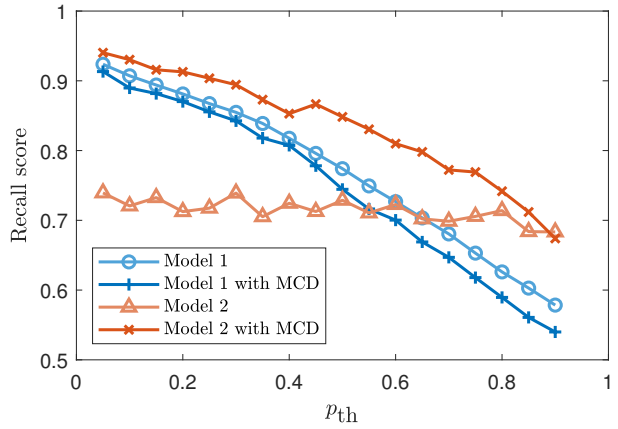
C. Evaluation

We first evaluate the presented grid prediction models with and without MCD on the test data set. Then, we tune the SMPC probability threshold p_{th}. Finally, the overall grid-based SMPC framework is assessed in the previously described simulation of a driving scenario. For a comprehensive assessment of predictor performance we will consider multiple metrics. As a general metric of prediction accuracy we could use the Mean Squared Error (MSE), however, for our problem MSE only illustrates part of the picture. There are different implications for errors in controller predictions, depending on whether the prediction failed to indicate that a cell is occupied (false negative) or it indicated a cell that is not occupied as occupied (false positive). A false negative poses a potential safety risk, as the SMPC controller might drive more aggressively because of too weak constraints. False positives on the other hand, pose less of a safety risk, but might decrease controller performance because they indicate tighter constraints than necessary. We therefore borrow metrics from classification

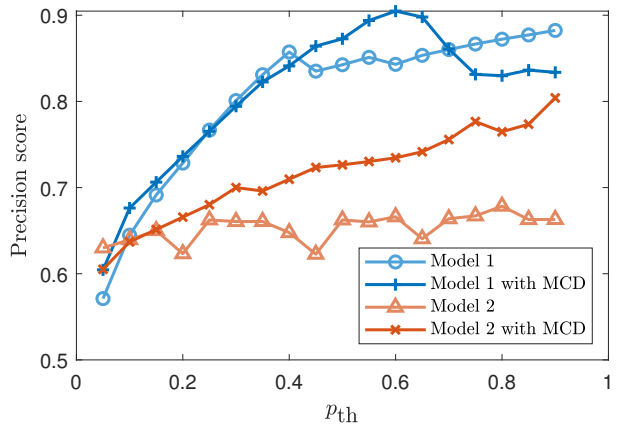
$$\text{Precision}_k = \frac{1}{n}\sum_{i=0}^{n}\frac{\text{TP}_{k,i}}{\text{TP}_{k,i} + \text{FP}_{k,i}} \tag{16}$$

$$\text{Recall}_k = \frac{1}{n}\sum_{i=0}^{n}\frac{\text{TP}_{k,i}}{\text{TP}_{k,i} + \text{FN}_{k,i}} \tag{17}$$

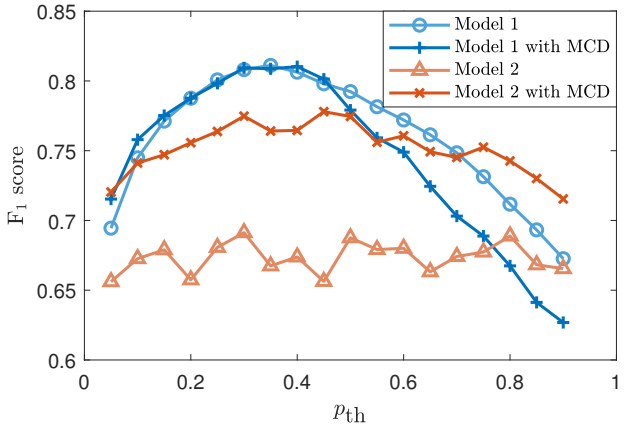
$$\text{F}_{1_k} = \frac{2}{n}\sum_{i=0}^{n}\frac{\text{Precision}_{k,i} \cdot \text{Recall}_{k,i}}{\text{Precision}_{k,i} + \text{Recall}_{k,i}} \tag{18}$$



(a) Recall scores at prediction step $k = 20$.

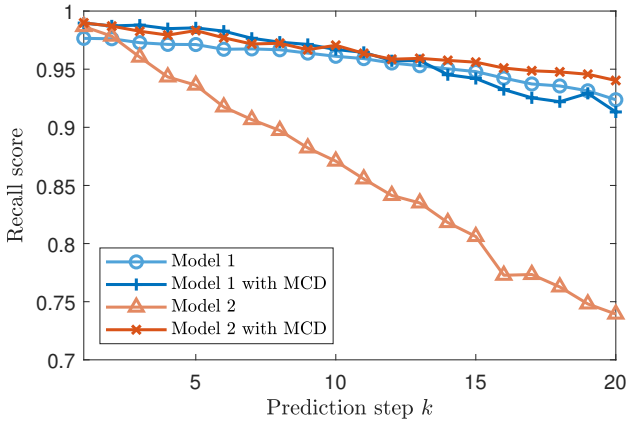


(b) Precision scores at prediction step $k = 20$.

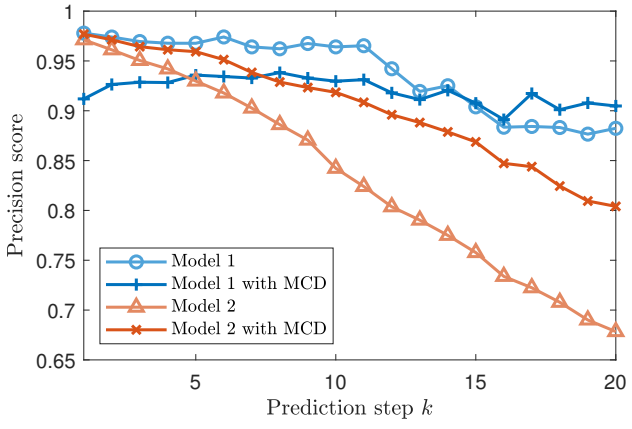


(c) F_1 scores at prediction step $k = 20$.

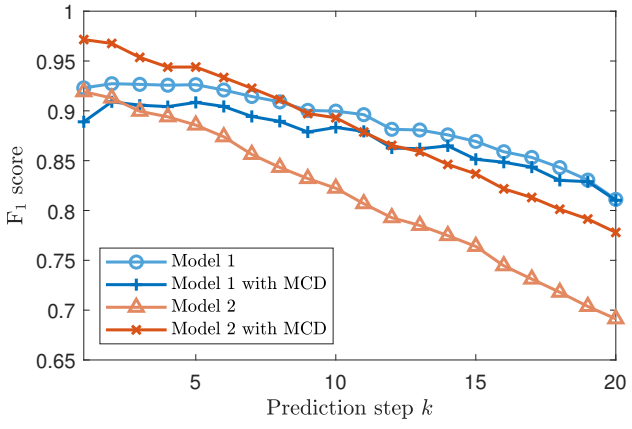
Fig. 3: Accuracy scores of all grid prediction models at the fixed prediction step $k = 20$ for different values of p_{th}.



(a) Optimal recall scores over time.



(b) Optimal precision scores over time.



(c) Optimal F_1 scores over time.

Fig. 4: Accuracy scores of all grid prediction models over the prediction horizon. The values of p_{th} are selected for each prediction model individually, according to the optimal values in Table III.

where i is the index of the test example and n is the total number of samples in the test data set. This means, $\text{TP}_{k,i}$ is the number of true positives in time step k of test example i. Respectively, $\text{FP}_{k,i}$ is the number of false positives and $\text{FN}_{k,i}$ is the number of false negatives. These metrics are computed on the predicted frames after they have been converted to BGs, as described in III-C, using the tunable threshold p_{th}. Here positives refer to occupied cells and negatives refer to

unoccupied cells. It is conceivable that there might be frames where $(\mathrm{TP}_{k,i} + \mathrm{FP}_{k,i}) = 0$ or $(\mathrm{TP}_{k,i} + \mathrm{FN}_{k,i}) = 0$. In these cases we assign $\mathrm{Recall}_{k,i} = 0$ if $\mathrm{FP}_{k,i} \neq 0$, $\mathrm{Precision}_{k,i} = 0$ if $\mathrm{FN}_{k,i} \neq 0$ and $\mathrm{Recall}_{k,i} = 1$ if $\mathrm{FP}_{k,i} = 0$, $\mathrm{Precision}_{k,i} = 1$ if $\mathrm{FN}_{k,i} = 0$.

Based on these evaluation metrics, we first tune the probability threshold p_{th} for all four predictor models at $k = 20$, as we want to have robust long-term predictions. With this parameter set, we can the evaluate the performance of the different predictors over time.

Figure 3 shows the scores of the four predictors plotted over p_{th}. In these figures we can identify the expected trend of recall decreasing with increasing probability threshold, as the criterion for a cell to be accepted as positive increases. Conversely, precision increases, as the criterion for a cell to be accepted as unoccupied decreases leading to less false positives. Striking is the curve of Model 2 which does not show a trend in any of the three plots. This seems to indicate that Model 2 is relatively unaffected by p_{th}. This behavior can be explained by the fact that Model 2 (compared to the other predictors), generates very crisp predictions with values either very close to 0 or to 1, therefore, it is reasonable to assume that the probability threshold would have a negligible effect here. Other than that we can see that the models generally produce good results, especially considering that this is the twentieth time step. These diagrams can now be used to select a p_{th} that gives a good compromise between specificity and safety. An optimal choice for the threshold based on the different scores can be seen in Table III.

In Figure 4 we show the scores of the different models for $N = 20$ with p_{th} chosen according to the best value for each metric. We can see that all scores stay generally high over the entire prediction horizon, with an exception of model 2 which shows a significant decrease in both precision and recall with increasing k. However, we see that MCD can significantly improve the performance of this model accross all metrics. Conversely, we can see that MCD apparently does not have a strong effect on the performance of Model 1. We suspect this might be because, upon closer inspection, the predictions by Model 1 are already somewhat blurry with increasing time step k, therefore MCD presumably does not exhibit such a big effect. By either choosing p_{th} according to precision, recall or F_1 score, we could now tune the model to be more aggressive, safe or a balance between both.

We test the overall control method exemplarily on Model 2 in the previously described driving scenario with three TVs. The experiment is run on a laptop with an Intel® i5 processor (1.60 GHz) and 8 GB of RAM. The mean computation time per iteration of the algorithm (including all interfaces between Python and MATLAB® and the simulation in SUMO) is $\mu = 1.3867\,\mathrm{s}$, and the standard deviation $\sigma = 0.3821\,\mathrm{s}$. The EV remains faster than the TV that is located in front on the same lane (TV2). Thus, the gap reduces until EV changes to the left lane and proceeds to perform a successful overtaking maneuver.

TABLE III: Optimal values for the probability threshold p_{th} for each prediction model, based on different scores (recall, precision, and F_1).

Prediction model	Optimal p_{th} based on score		
	Recall	Precision	F_1
Model 1	0.05	0.90	0.35
Model 1 with MCD	0.05	0.60	0.40
Model 2	0.05	0.80	0.30
Model 2 with MCD	0.05	0.90	0.45

D. Discussion

The risk parameter p_{th} plays an essential role in SMPC. A higher value of p_{th} causes less grid cells to be regarded as occupied. Only the cells that the prediction model considers to be occupied with a high degree of certainty can remain above p_{th}. Thus, the admissible space becomes larger with a rising p_{th} and the behavior of the controller proves more aggressive. In the same way, low values of p_{th} result in a more conservative driving behavior.

Tuning the parameter p_{th} during control design requires a compromise between aggressiveness and conservatism. When aiming for a conservative control behavior, we need to ensure that the recall score of the PG is as high as possible to minimize false negatives. Likewise, the precision score of the PG needs to remain as high as possible to avoid false positives and achieve an aggressive control behavior. If, on the other hand, a balanced driving behavior is desired, the F_1 score is a good trade-off. Therefore, we find the values for the probability level p_{th} that are optimal for each case and each grid prediction model (see Table III) and suggest their use, depending on the pursued vehicle behavior.

The high computational cost of the developed approach is mainly due to the nonlinear EV model used for the OCP and the followed MCD approach, which requires to repeat the inference procedure multiple times. Hence, the computational complexity of the control method can be further reduced by two strategies. First, the EV prediction model can be linearized for a faster OCP. Second, we can either decrease the number of iterations of the MCD procedure at the cost of a worse uncertainty estimation and reduced recall score, or implement BNNs.

V. Conclusion

In this work, we have developed an approach to couple the grid-based SMPC framework for trajectory planning in automated driving with RNN-based multi-step grid prediction models. We make use of Monte Carlo dropout during inference to model the stochastic uncertainty, which is introduced by the dynamic objects around the controlled vehicle. The method handles complex driving scenarios with different types of vehicles and maneuvers effectively. It is still of interest to evaluate the scalability of the method in urban and real-life

situations, as well as applying bayesian neural networks for faster forecasts with uncertainty estimation.

REFERENCES

[1] A. Mesbah, "Stochastic model predictive control: An overview and perspectives for future research," *IEEE Control Systems*, vol. 36, no. 6, pp. 30–44, 2016.

[2] A. M. Carvalho, "Predictive control under uncertainty for safe autonomous driving: Integrating data-driven forecasts with control design," 2016.

[3] T. Brüdigam, F. Di Luzio, L. Pallottino, D. Wollherr, and M. Leibold, "Grid-based stochastic model predictive control for trajectory planning in uncertain environments," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–8, IEEE, 2020.

[4] S. Steyer, G. Tanzmeister, and D. Wollherr, "Object tracking based on evidential dynamic occupancy grids in urban environments," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1064–1070, IEEE, 2017.

[5] S. H. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi, "Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1672–1678, 2018.

[6] M. Schreiber, S. Hörmann, and K. Dietmayer, "Long-term occupancy grid prediction using recurrent neural networks," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9299–9305, IEEE, 2019.

[7] N. Mohajerin and M. Rohani, "Multi-step prediction of occupancy grid maps with recurrent neural networks," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Los Alamitos, CA, USA), pp. 10592–10600, IEEE Computer Society, 2019.

[8] S. Hörmann, M. Bach, and K. Dietmayer, "Dynamic occupancy grid prediction for urban autonomous driving: A deep learning approach with fully automatic labeling," *Computing Research Repository (CoRR)*, 2017.

[9] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *Physica D: Nonlinear Phenomena*, vol. 404, no. 8, p. 132306, 2020.

[10] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*, pp. 2575–2582, IEEE, November 2018.

[11] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," *Computing Research Repository (CoRR)*, 2015.

[12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[13] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, vol. 3, 2014.

[14] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *Journal of Machine Learning Research*, vol. 18, no. 185, pp. 1–52, 2018.

[15] L. V. Jospin, W. L. Buntine, F. Boussaïd, H. Laga, and M. Bennamoun, "Hands-on bayesian neural networks - a tutorial for deep learning users," *CoRR*, vol. abs/2007.06823, 2020.

[16] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," *Proceedings of The 33rd International Conference on Machine Learning*, 06 2015.

[17] F. Chollet *et al.*, "Keras," 2015.

[18] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.