

Improving Deep Dynamics Models for Autonomous Vehicles with Multimodal Latent Mapping of Surfaces

Johan Vertens* Nicolai Dorka* Tim Welschehold Michael Thompson Wolfram Burgard

Abstract—The safe deployment of autonomous vehicles relies on their ability to effectively react to environmental changes. This can require maneuvering on varying surfaces which is still a difficult problem, especially for slippery terrains. To address this issue we propose a new approach that learns a surface-aware dynamics model by conditioning it on a latent variable storing surface information about the current location. A latent mapper is trained to update these latent variables during inference from multiple modalities on every traversal of the corresponding locations and stores them in a map. By training everything end-to-end with the loss of the dynamics model, we enforce the latent mapper to learn an update rule for the latent map that is useful for the subsequent dynamics model. We implement and evaluate our approach on a real miniature electric car. The results show that the latent map is updated to allow more accurate predictions of the dynamics model compared to a model without this information. We further show that by using this model, the driving performance can be improved on varying and challenging surfaces.

I. INTRODUCTION

In recent years autonomous cars have become reliable enough to be deployed in the real world [1], [2]. Nonetheless, they can currently operate safely only under limited conditions such as a mapped environment, good weather, or specific types of roads. In the unstructuredness of the real world with unforeseeable situations, potentially caused by mistakes of other road users, the autonomous car might be required to drive at the edge of its ability under all environmental conditions in order to avoid accidents. One such scenario is driving on varying surfaces which becomes especially challenging if their corresponding friction values differ a lot. Maneuvers that are safe to execute on one surface can be dangerous or even impossible on a different surface. For autonomous vehicles, this poses a safety-critical problem as the driving has to be adjusted according to the current surface and a wrong estimate about possible future trajectories can result in disastrous outcomes. How to autonomously learn to predict features of a surface and the corresponding impact on the future trajectory without supervision by ground truth friction values for all surfaces is still an open research question.

Previous works allowing stable handling at the edge of controllability considered a single surface [3], [4], [5] or achieved steady state drifting but no cornering [6]. A typical approach in the case of varying surfaces is to use a separate

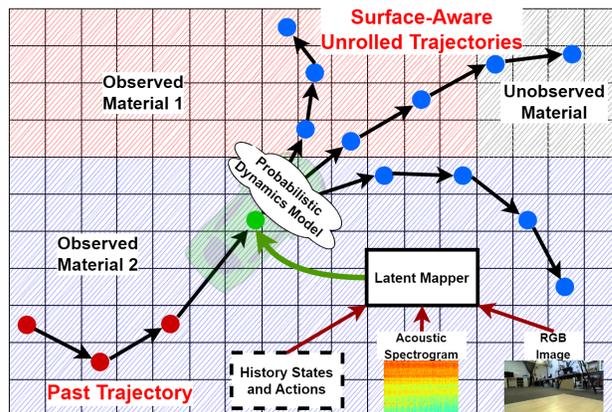


Fig. 1: Overview of our approach. During driving the latent mapper uses different modalities like sound and vision to update grid cell variables that represent information about the road and its properties at that location. The learned dynamics model receives the latent variable corresponding to its current location as additional input to allow for trajectory predictions that are aware of the road material.

terrain classifier [7], [8], [9], which, however, assumes a fixed number of prespecified terrain classes. This problem can be overcome by using conditional dynamics models [10] but the inferred information is not stored in a map and can hence not be used for later traversals of the same location. Other works estimate the explicit coefficient of friction inside a physics model [11], [12]. This, however, relies on the assumption that an accurate model can be identified, which is difficult for slippery surfaces.

We tackle this problem with a new approach that trains a dynamics model to include information from a learned and automatically updated latent map. During inference, a latent mapper updates the map, such that the latent variable at a specific location stores valuable information about the surface material at that position. The dynamics model uses the latent variable for its current location as additional input to make accurate surface-dependent predictions of the next state. Several traversals through a single location are given sequentially to the latent mapper to update the corresponding latent variable in an autoregressive fashion. Since all model parts are differentiable, we can train the dynamics model and the latent mapper simultaneously from the same loss in a way that the latter learns to give useful surface information to the former. Motivated by the fact that humans automatically use multiple cues to infer what driving style is appropriate for the surface they are driving on, the latent mapper receives multiple modalities like images and sound allowing it to learn a general representation of different surface patterns.

We implement and evaluate our method on a real miniature

*These authors contributed equally. Johan Vertens, Nicolai Dorka, and Tim Welschehold are with the University of Freiburg, Germany. Michael Thompson is with Toyota Research Institute, Los Altos, USA. Wolfram Burgard is with the University of Technology, Nuremberg, Germany. Corresponding author: vertensj@informatik.uni-freiburg.de

electric car. The results show our approach is able to generate complex driving maneuvers on unknown and varying surfaces, showing the benefit of implicit terrain maps. To summarize, our contributions are

- a latent mapper trained to update a map with surface information from multiple modalities
- a surface-aware dynamics model using the latent map
- the implementation on a real electric car
- an extensive evaluation of the single components of our method and their combinations
- demonstrating the benefit of our latent mapping relative to a baseline without surface awareness in the real world.

II. RELATED WORK

A. Surface-Awareness

Terrain classification can be done in an automated way [13] or from different modalities [14]. Several approaches use an explicit terrain classifier trained with human-specified labels and deploy a terrain-specific policy accordingly [7], [8], [9]. More similar to our approach is the work of Nagabandi *et al.* [10] where additionally to state and action an image is given as input to the dynamics model such that it can infer the next state conditioned on the surface the robot is currently moving on. Furthermore, meta-learning has been used for learning to adapt the dynamics model parameters during inference to a given environment including different terrains [15]. Our approach is different in that it updates the surface embedding during inference allowing it in principle to learn the adaption to any new surface and to further store the information in a map.

B. Explicit Estimation of the Coefficient of Friction

One can compute possible trajectories using the coefficient of friction between the tires and the road. However, the coefficient depends on all kinds of conditions and has to be estimated from real data. For example, Huang *et al.* [12] estimate the coefficient with a limited-memory adaptive extended Kalman Filter to reduce the effect of outdated measurements on filtering. Multiple surveys [16], [17] cover these topics. Deep neural networks can use ultrasound as input to classify the road surface from which the coefficient of friction is derived [18]. For autonomous race cars driving several laps on a track with an inhomogeneous surface covering has been proposed to estimate the coefficient of friction and store it in a map [11]. All of these approaches have in common that they use a physical dynamics model for which they estimate the corresponding parameters. Our approach is more flexible in that it can learn any dynamics model without the need to model the underlying physics. Thus, our work is most comparable to other methods that learn dynamics models solely from data. Our focus is to improve ML-based methods by introducing surface-awareness.

C. Dynamics Models with Latent Variables

Some works train a network for system identification from the most recent history to output a latent variable which is then given as additional input to a model-free RL policy [19] or the dynamics model in model-based RL [20], [21]. In the meta-RL setting Rakelly *et al.* [22] train a network to

produce a probabilistic context variable given as additional input to an off-policy RL algorithm to adapt to a given task.

Our approach differs in that it builds a map over the state space and learns to learn different context embeddings at different locations in the state space during inference.

III. TECHNICAL APPROACH

In our work, we employ a dynamics model that predicts a potential future state given an input action. Subsequently, the dynamics model is used to control a vehicle along pre-defined trajectories following a model predictive scheme. Our approach does not model explicit parameters of a physical model nor does it take factors such as tire pressure or temperature into account. One could combine these aspects with our method but this is beyond the scope of this work.

In contrast to previous approaches, we propose a method that considers potential changes of the road material along the track, which may influence the dynamic behavior of the vehicle. Therefore, we first create a grid map that is defined in global coordinates and equally divides the space into quadratic cells. Once the vehicle traverses a cell of the grid map, a mapping neural network infers a latent vector that describes the local road surface. Thereby, the mapping network leverages various modalities that were recorded during the traversal of the cell, such as RGB images, acoustic spectrograms, history states, and history actions. Our dynamics model takes, aside from low-level state and action information, the latent vector corresponding to the current vehicle location as an input, such that it can leverage the additional mapped cues from multiple modalities to optimize the predictions of future states, making it surface-aware. As the training of the mapping model is guided by the loss of the dynamics model predictions, we do not require any labels that associate the input modalities with specific surface characteristics. Note, that in contrast to previous approaches, our work models the whole surface-aware dynamics as a neural network, which avoids assumptions or inductive biases for the created surface map. Next, we unroll multiple trajectories using the dynamics model and sampling from the action space. Our system then employs a reward function and the cross entropy [23] method to score and select the trajectories that follow a reference path as fast and close as possible. Following a typical model predictive scheme, we execute the first action of the resulting plan and restart the process. In the following we describe each component of our system in more detail, followed by an explanation of the loss functions and training procedure.

A. Surface-Aware Probabilistic Dynamics Model Ensemble

Our dynamics model predicts the next output state s_{t+1}^{out} , given the current input state s_t^{in} and the current action a_t . To this end, we distinguish between input states, which are the representation of the input to the dynamics model, and output states, which are the prediction of the dynamics model. We define the input state as the concatenation of 3D linear velocities vel_l , angular velocities vel_a , linear accelerations acc_l , angular accelerations acc_a , and motor rpm : $s_t^{\text{in}} = [\text{vel}_l^T, \text{vel}_a^T, \text{acc}_l^T, \text{acc}_a^T, \text{rpm}]^T$. Additionally, we define the predicted output state of the dynamics model as estimated

local changes in the x-position Δp_x , y-position Δp_y and yaw angle $\Delta\gamma$, all velocities, and all accelerations as $s^{\text{out}} = [\Delta p_x, \Delta p_y, \Delta\gamma, \text{vel}_l^T, \text{vel}_a^T, \text{acc}_l^T, \text{acc}_a^T, \text{rpm}]^T$. The action is composed of the throttle a_{th} and steering command a_{st} , such that $a = [a_{\text{th}}, a_{\text{st}}]$. We make our model surface-aware by using additional latent vectors as input to the dynamics model. These latent vectors describe the local learned properties of the road. Therefore, we propose to learn a latent map L that is represented as a grid map where each quadratic cell c holds a distribution over the latent vector. Here, we assume each entry of the map to be a k_l -dimensional multivariate normal distribution that is parametrized by l_θ^c , corresponding to cell c . As the vector is learned implicitly, the number of dimensions k_l represents a hyperparameter. To predict the next output state, we first sample a latent vector l^c from the latent distribution $\mathcal{N}(l_{\theta_\mu}^c, l_{\theta_{\sigma^2}}^c)$ at the cell corresponding to the current vehicle position. Following, we employ an ensemble of probabilistic dynamics models [24] with parameters ψ to predict the next state, while capturing model and data uncertainties. The input of this ensemble comprises the current input state, the current action, and the sampled latent vector, which we feed by simple concatenation. Thus, we model the Gaussian distribution of the next state as:

$$f_\psi(s_{t+1} | s_t, a_t) = \Pr(s_{t+1} | s_t, a_t, l^c; \psi). \quad (1)$$

For clarity, we omitted the differentiation between the input and the output state.

B. Mapping Network

To estimate the latent map, we propose a novel neural network architecture that takes a variety of modalities as input. In more detail, when the car traverses a cell c , we leverage an RGB image I^c , an acoustic spectrogram S^c , the history of states H_s^c , the history of actions H_a^c , and the previous estimate of the mean and variance of the latent vector l_θ^c that were recorded in the same cell c . The cues are then encoded into high-level features using respective encoders. These features are then concatenated and passed to another MLP with two output heads, which predicts the mean and variance of the latent vector respectively. As mentioned in Sec. III-A, the means and variances are then aggregated to a latent map L . More formally, let ϕ be the parameters of the mapping model and \tilde{l}_θ^c the updated parametrization of the Gaussian distribution of the latent vector in cell c . We define a latent update for the cell c as:

$$\tilde{l}_\theta^c = M_\phi(I^c, S^c, H_s^c, H_a^c, l_\theta^c), \quad (2)$$

Note, that since we input previous latent estimates l_θ^c , the latent representation of the road surface is updated iteratively.

While the RGB image may entail visual information of the road, the acoustic spectrograms capture direct tire-road interactions that are characteristic for specific materials. Additionally, from learning about the history of states and actions, our mapping model can infer ground patterns that lead to specific state sequences given the respective actions.

Particularly at inference time, estimating the road characteristics from only the low-level state and action history would most likely fail when the vehicle is at slow speeds or stands still. To argue about the road surface, the dynamic

behavior needs to differ across different road materials due to distinct friction characteristics. This, however, is only given in cases where the maximum frictional force [25] that is achieved is smaller than the force that is needed to sustain the vehicle track. Thus, to enable arguing about the road material, situations are required in which the car starts slipping. While, for training purposes of the dynamics model, this data can be collected by an expert driver, uncontrolled slipping should be avoided during inference. To this end, our multimodal approach allows learning a mapping that associates visual or acoustic cues to a latent representation of the road without the requirement of slipping. As an example, our network can learn to associate a slippery surface with the visually shiny appearance of the road. Further, acoustic spectrograms can contain information about the road even under low velocities.

Consequently, we require examples of aggressive driving only during training, while during inference the road representation can be estimated under all conditions. Furthermore, the employed modalities can complement each other if one modality lacks information due to visual occlusions, low lighting, or when external acoustic events drown out important acoustic tire-road interactions. We show in Sec. VI that leveraging multimodal data yields high gains in state prediction performance.

C. Training of the models

We first collect a training dataset with dynamic examples of random driving in environments with spatially changing road materials. These driving examples include situations where the car slips. To train our networks we propose a loss function that fulfills two requirements:

- Unrolling of future states may lead to querying cells that have not been observed yet. In these cases, it should be possible to inform the dynamics model of zero knowledge of the surface material.
- The outputs of the mapping model should represent a spatial property of the road surface that is valid for any state prediction in the respective cell.

To accomplish these requirements, we first group the individual recorded ground-truth state transitions according to the cell c in which the transition was captured. Here, we denote the n -th state-transition in our dataset that occurred in the cell c as $s_t^{c^n} \rightarrow s_{t+1}^{c^n}$. Now, having a list of all state transitions that occurred in the same cell, we select N random state transitions within a cell and define the loss for training the dynamics model as:

$$\mathcal{L}_d = \sum_{n=\{0,1,\dots,N\}} (\mathcal{L}_g(f_\psi(s_{t+1}^{c^n} | s_t^{c^n}, a_t^{c^n}, l^{c^n}), s_{t+1}^{c^n})), \quad (3)$$

where $l^{c^0} = 0$ and for $n > 0$:

$$l^{c^{n+1}} = M_\phi(I^{c^n}, S^{c^n}, H_s^{c^n}, H_a^{c^n}, l_\theta^{c^n}), \quad (4)$$

and where $\mathcal{L}_g(\theta, \text{target}) = \frac{1}{2}(\log \theta_\sigma^2 + \frac{(\theta_\mu - \text{target})^2}{\theta_\sigma^2})$ is the Gaussian negative log-likelihood loss, and $s_{t+1}^{c^n}$ is the ground truth target state. The order of the traversals is irrelevant to our loss. Our loss represents the update scheme of the latent vectors, in which in the first iteration no knowledge of

the surface is assumed. Thus, in the first iteration, a latent vector for the dynamics model is defined as a zero-vector, which forces the dynamics model to predict a future state distribution that is broad enough to cover all road materials properties that appear during training. This is particularly useful when unrolling state sequences over cells that have not been observed yet. In these cases, a conservative estimate of the state distribution is required as unobserved cells may entail any material or surface condition. In the following iterations of our loss function ($n > 0$), the latent vector fed to the dynamics model is updated using our mapping model. In our loss function, the latent update is calculated based on the observed data of the previous traversal $n - 1$, while the dynamics model predicts the state transition for the current traversal n in the same cell c . Thus, we ensure that the latent vectors are independent of the currently predicted state transition and represent a joint representation that improves prediction accuracy for all transitions in the respective cell. Note that if the dynamics model would receive a latent vector generated by the mapping network using data recorded at the same time as the inputs of the dynamics model, the mapping model could directly contribute to the prediction of the next state rather than representing a spatial property of the track. For our experiments, we set the number of selected state transitions to $N = 3$.

1) *Three Stage Training*: We optimize our model using a three-stage approach. In the first stage, we optimize the dynamics model as well as the latent vectors but without training the mapping network. Instead, we optimize the latent vectors l^c directly by backpropagating into them, treating the map as model parameters. We denote the directly optimized latent parameters as $\bar{l}^c \in \bar{L}$. In contrast to the limited information of the input of the mapping network at inference time, this has the advantage that the latent vectors can be thoroughly optimized over all batches of the dataset. We denote the resulting loss as:

$$\mathcal{L}_{d_{s1}} = \sum_{n=[0,1,\dots,N]} (\mathcal{L}_g(f_\psi(s_{t+1}^{c^n} | s_t^{c^n}, a_t^{c^n}, \bar{l}^c), s_{t+1}^{c^n})) \quad (5)$$

In contrast to the later training stages, we do not inject zero-vectors, while optimizing the latent vectors directly as we presented in Eq. 3. Experiments have shown that the training becomes unstable otherwise.

To avoid agitated latent maps, we add a smoothness term minimizing the local gradient of the latent maps:

$$\mathcal{L}_s = \|\nabla \bar{L}\|^2 \quad (6)$$

The overall loss being optimized in the first stage is simply a weighted sum of both loss functions:

$$\mathcal{L}_{s1} = \mathcal{L}_{d_{s1}} + \lambda \mathcal{L}_s, \quad (7)$$

where λ denotes a weighting hyper-parameter that defines the strength of the smoothness term.

However, as the parameter map \bar{L} is optimized offline, it can not be employed in practical applications as the vehicle should be capable of driving through previously unseen environments. By leveraging our multimodal mapper, new environments should be observed on-the-fly avoiding this limitation. Thus, in the second stage, we freeze the

learned latent parameters \bar{L} and the dynamics model while optimizing the mapping network. In this stage, we guide the latent predictions from our mapping model, by optimizing the negative log-likelihood of the predicted latent vector distribution l_θ^c given the learned parameter corresponding to the same cell l^c . Overall, the loss for the second stage of our training scheme is defined as:

$$\mathcal{L}_{s2} = \sum_{n=[1,\dots,N]} \mathcal{L}_g(l_\theta^{c^n}, \bar{l}^{c^n}). \quad (8)$$

In the last stage, we then freeze the mapping model and refine the dynamics model by optimizing Eq. 3 and feed the estimate l^c from the mapping network into the dynamics model instead of the previously used learned parameter \bar{l}^c .

D. Planning and Control

In order to follow a reference trajectory T_r , we follow a model predictive control approach. In detail, we use iCEM [26], which generates multiple future state sequences by sampling over the actions. The best sequences are selected using a pre-defined reward function and are refined for a specific amount of iterations. In contrast to the vanilla CEM [23], iCEM provides significantly better sample efficiency and generates smoother trajectories due to enforced temporal consistency along the state sequences. These properties make the sampling-based planning real-time capable, which is a crucial requirement for high-speed autonomous driving.

1) *Trajectory Unrolling and Latent Sampling*: To generate candidate trajectories for the planning module, we start from the initial current state of the vehicle and unroll future state sequences by successively applying our dynamics model given a sequence of actions. We then accumulate all predicted local changes of the x-position Δp_x , y-position Δp_y , and yaw angle $\Delta \gamma$ to convert the state sequences into trajectories that are defined in the coordinate system of the initial state. Finally, we add the initial position of the vehicle to convert them into the global coordinate system. To sample multiple trajectories from our ensemble of dynamics models, we employ the *TSI*-strategy [24]. In each iteration of iCEM we generate trajectories for N_a distinct action sequences. As our ensemble of dynamics models predicts a single state transition at a time, we apply our dynamics model h times for a trajectory with the length of h time steps and the same number of actions. Further, we sample k different state hypotheses for each individual action, effectively resulting in $N_a * k * h$ inferences passes of our dynamics model and $k * N_a$ trajectories. During trajectory generation, the latent vectors are sampled from the respective latent distribution for each individual state of the trajectories and leveraged for the next update using the dynamics model. Thus, intra-trajectory changes of the road materials are considered.

2) *Map Update*: We start with a map in which the parametrization of each cell is set to zero. As discussed in Sec. III-C this indicates zero knowledge about the map. As the latent distribution in each cell can be updated asynchronously with respect to the prediction of the dynamics model, we run the mapping model and the controller including the dynamics model in separate threads. This ensures that

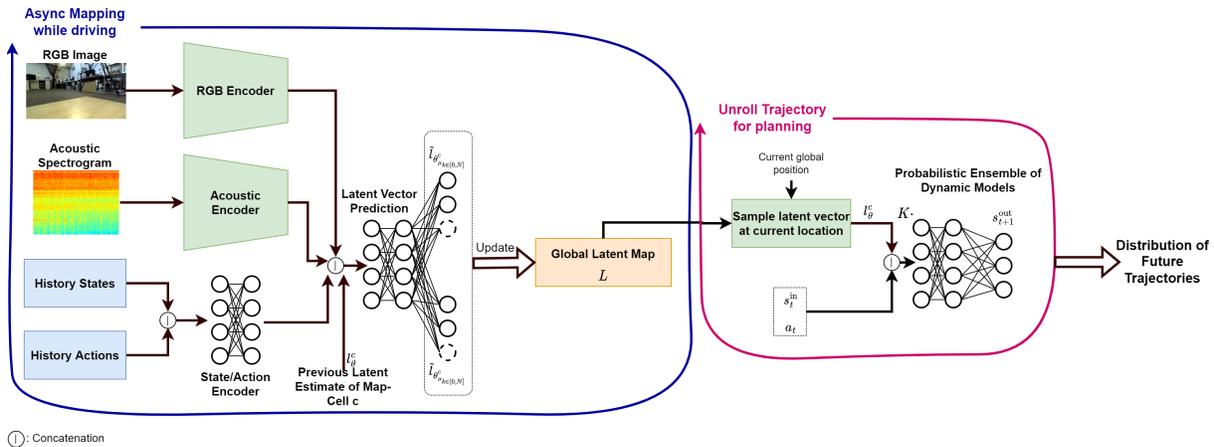


Fig. 2: In our approach the vehicle estimates a distribution of a latent representation of the road surface using different modalities, such as history state/action information, RGB images, and acoustic spectrograms. Thereby, all local latent distributions are collected in a global grid map L . Once the vehicle traverses a cell c of the grid map, the underlying latent distribution l_c^θ gets updated. For planning purposes, an ensemble of probabilistic dynamics models estimates the distribution of the future state s_{t+1} given the actions a_t , the current input state s_t and a sampled latent vector from the latent distribution of the grid-map-cell that corresponds to the input state s_t . As the dynamics model can be successively applied, a distribution of future trajectories can be obtained. As both mechanisms, latent mapping and state prediction can run independently from each other, the mapping is conducted asynchronously.

the dynamics model does not need to wait for the mapping inference to finish and allows for parallelization as we run the dynamics model on CPU and the mapping model on GPU.

3) *Reward Function*: To score all trajectory candidates, we propose a reward function that evaluates these in terms of different metrics. The target trajectory is given by $x - y$ coordinate waypoints. We measure the deviation of a given trajectory to the target with the cross-track error R_{cte} . As we want the car to move as fast as possible we define a progress reward R_p by adding up the length of all the line segments between waypoints the trajectory passes. Further, to punish risky maneuvers and prevent shortcuts we define a binary boundary violation reward R_b which is equal to 1 if a specified boundary around the waypoints is exceeded and 0 otherwise. Lastly, to encourage smooth driving we define the reward R_a as the absolute difference between the last executed throttle command and that of the first action of the trajectory. The final reward for a trajectory is defined as

$$R = w_p \cdot R_p - w_{cte} \cdot R_{cte} - w_a \cdot R_a - w_b \cdot R_b, \quad (9)$$

where we set the weighting parameters to $w_p = 40$, $w_{cte} = 10$, $w_a = 20$, $w_b = 20000$.

IV. DATASET

To train and evaluate our approach we propose a novel dataset which we refer to as *Dynamic FreiCar*. As an experimental vehicle, we employ a rear-driven miniature 1:8 scale car that is equipped with a computer, various sensors, and a high-torque electric motor. To capture RGB images, we leverage a *ZED* camera, while we use a *Rode* compact microphone to record audio data. We further use *Valve Lighthouses* to track the position of the car and gather ground-truth velocity and acceleration data. Our dataset contains 70 minutes of expert driving along random trajectories that include drifting scenarios. All data is recorded at 100hz. We use two types of wood laminate and gym rubber mats for the different surface materials. To avoid bumps at the transition



Fig. 3: Left: Our experimental vehicle. The background shows the various surface materials over which the driving is conducted. Center and right: the driven path of the vehicle with and without map information.

we level out the different materials. To ensure fair training and evaluation splits we create two maps for training and one map for evaluation by spatially rearranging the materials. Thus, as we train on multiple maps, we avoid overfitting to a specific map, which we validate in our experiments section. Figure 3 shows our experimental vehicle and the driving environment. Note that our approach does not model explicit friction values and we do not have the ground truth friction values for the different surfaces.

V. IMPLEMENTATION DETAILS

We implement our approach on top of MBRL [27], a pytorch-based [28] general framework for model-based reinforcement learning. We employ a ResNet-18 architecture for the image and spectrogram encoders respectively. For the base architecture of the dynamics model, we leverage the implementation of the Gaussian ensemble from MBRL.

We use a learning rate of 0.001 for optimizing the parameters of the dynamics model and the map parameters, and a different learning rate of 0.0001 for the mapping model. We train all models with a batch size of 96 across four *RTX 2080 ti* GPUs for 500 epochs. For inference and running the model predictive controller, we use a Ryzen 5950X processor.

To extract the spectrograms from the acoustic data, we use an FFT with 257 bins and a hop length of 128. We extract the spectrogram over the last second of data and convert it to

decibels afterward. The images and spectrograms are resized to 336x188 before passing them to their respective encoders.

We further define the time span of a single state transition as 0.1s and set the map resolution to 50cm. As for the iCEM optimizer, we set $\beta = 4$ and $\gamma = 1.3$. Furthermore, we use a planning horizon of $h = 8$ steps, 32 samples in each iteration, and set the number of iCEM iterations to 2.

VI. EXPERIMENTAL RESULTS

To assess the efficacy of our method we train on the training set of *Dynamic Freicar*.

A. Evaluation of the Dynamics Model

We evaluate the prediction accuracy of the dynamics model on the test set of our dataset. To generate a test set that represents practical use-cases, we leverage our method to drive 10 laps autonomously on track 3 (see Sec. VI-C). We record all state transitions during this run and employ them as the test set for the following evaluation. As unrolling long trajectories are practically important for planning and control, we introduce a metric that describes how well the predicted states align with the ground truth while considering uncertainty. Thus, for all time steps in our dataset, we unroll a sequence of N_s states using the future actions that have been carried out following the current state. To capture multiple hypotheses we repeat the unrolling process 100 times for the same action sequence. Now, given 100 hypotheses of the future trajectory, we compute the euclidean distance between each point of every trajectory hypothesis and the observed ground-truth trajectory that was driven. Finally, we compute the mean euclidean error. More formally, let D_s be the set of all observed states in the dataset, $H(s)$ all unrolled hypotheses starting in state s , and $\overline{T(s)}_n$ the n -th point of the observed ground-truth trajectory that starts in s . Then we define the metric as:

$$L2_{N_s} = \frac{1}{N_D N_H N_s} \sum_{s \in D_s} \sum_{h \in H(s)} \sum_{n=0}^{N_s} \| h_n - \overline{T(s)}_n \|_2 . \quad (10)$$

where N_D is the number of starting states in our dataset and $N_H = 100$ is the number of rolled-out hypotheses. We compute our evaluation metrics in the chronological order of the test dataset by iteratively updating the latent cues of the grid map as more data about the surface can be observed over time. This evaluation scheme strictly represents the practical deployment of the model, since the full map can not be leveraged at the beginning but builds up progressively.

We compare our approach against a baseline that does not employ any latent mapping (Ours-w/o map) similar to PETS [24] and additionally present the performance of our approach when only a subset of the modalities are leveraged for the mapping model. We denote a model that uses only images, spectrograms or history state/action information for latent mapping as *Ours (I)*, *Ours (A)* or *Ours (S)* respectively. Furthermore, we show the results of a dynamics model that takes the ground-truth terrain types instead of the estimated latent vectors of the map as input. To encode the ground-truth surface we simply provide a scalar value to the model indicating on which terrain type it is operating. We denote

this model as *Ours (GT)*. We argue that this model should provide the highest accuracy as the terrain is known at all times. As described in Sec. II, the settings tackled in previous works deviate considerably from ours in terms of surface map representation and hence we refrain from comparing to them. In our work, we set the focus on evaluating the benefits of latent surface maps for dynamics models.

The results in Table I show that our multimodal latent mapper (*Ours (AIS)*) significantly boosts prediction performance. By learning and mapping multimodal cues about the surface material, we reduce the error for $N_s = 30$ from 0.462 to 0.374 corresponding to a reduction of 19% over a model without the mapping model. In comparison to the model with the ground-truth terrain types as input (*Ours (GT)*), our model is on par with respect to the $L2_{10}$ metric and only 1.9% and 4.2% worse for the $L2_{20}$ and $L2_{30}$ metrics respectively. As the performance of the model with ground-truth terrain types is expected to be an upper limit, one can argue that our model effectively predicts the surface information needed to improve prediction performance.

Further, one can observe that the dynamics model performs best when using a mapping model that takes all modalities as input. Leveraging only the history of state/action information yields an $L2_{30}$ error of 0.422, only acoustic yields 0.397, and only visual cues yields 0.383. An interesting observation is that the combination of the history of states/actions and acoustic cues reaches a very low $L2_{30}$ error of 0.393, even though images are not employed in this case. In contrast to RGB images, acoustic spectrograms do not contain information on the spatial location of the car. Thus, the image-free version of our mapping model is less prone to overfitting. Although we could not observe overfitting of the mapping model during our experiments, this could ease training in more large-scale scenarios. Further, some of the improvement of using all modalities over just using the image might come from the fact that the front-facing camera only captures the surface farther in front of the current position, which might result in poor image-terrain associations at surface transitions.

While we used a ten-dimensional latent vector ($k_l = 10$) for the previously explained models, we additionally evaluate models that leverage all modalities and employ 1 and 5 elements. These models are denoted as *Ours (AIS, $k_l = 1$)* and *Ours (AIS, $k_l = 5$)* respectively. Here, one can note that estimating ten-dimensional latent vectors additionally improves the $L2_{30}$ metric by 10% over a model that employs a single dimension. We explain this effect by the thesis that overparametrization of the surface information eases the training procedure and helps avoiding local minima. Estimating more than ten latent dimensions did not show further performance gains during our experiments.

B. Qualitative Latent Maps

To investigate the learned latent maps in more detail, we visualize the learned map using a color-coding and $k_l = 10$. A principal component analysis (PCA) projects the multi-dimensional latent vector to a single scalar. We create the latent vectors using all available data of the test-split (10

Model	$L2_{10} \downarrow$	$L2_{20} \downarrow$	$L2_{30} \downarrow$
Ours-w/o map	0.094	0.262	0.462
Ours (S)	0.090	0.242	0.422
Ours (A)	0.085	0.228	0.397
Ours (I)	0.081	0.218	0.383
Ours (AS)	0.082	0.225	0.393
Ours (AIS- $k_l=1$)	0.087	0.237	0.418
Ours (AIS- $k_l=5$)	0.080	0.223	0.400
Ours (AIS)	0.079	0.212	0.374
Ours (GT)	0.079	0.208	0.359

TABLE I: Quantitative evaluation of the prediction accuracy of our proposed dynamics network. We present comparisons to a model that is not using a latent map as input and to a model that leverages a map that is optimized offline as parameters.

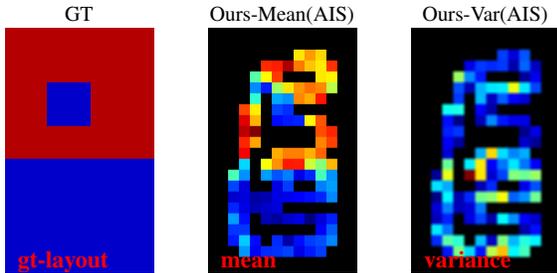


Fig. 4: Qualitative results of aggregated predicted latent vectors. It can be observed that the predicted maps are representing different materials well.

laps) of our dataset. The updates of the latent vectors are conducted in chronological order of the split. Fig. 4 shows the learned map, the predicted variance of the latent estimate, and the corresponding ground-truth layout. The experiment clearly reveals that our latent embedding correlates with the real-world layout of the test environment. Thus, we can validate that our mapping model predicts spatial cues of the track surface. Further, we observe that the predicted variance of the latent cues correlates with the locations of transitions between different materials. This follows our intuition as the observed data corresponding to a single cell of the grid map can contain information about multiple materials, since the cell can potentially stretch over material transitions. As the test split entails a distinct arrangement of the surface materials not seen during training, we can confirm that our mapping model generalizes over the map structure.

C. Real-World Planning and Control

To evaluate our surface-aware dynamics model for practical applications, we employ our model predictive controller for autonomous racing. The experiment is designed to investigate the effect of the dynamics model being surface-aware or not under otherwise same conditions. The dynamics model could be further improved by taking other features such as tire pressure and temperature into account. However, the car does not have sensors for this and these are orthogonal improvements that are beyond the scope of this work. We conduct the experiments on three diverse single-lane maps varying significantly in curvature. Track 1 and 3 have three different surfaces with highly varying friction values while Track 2 is completely on a slippery surface. The accompanying video shows all maps as well as the resulting driving of our approach. We quantify the driving performance in terms

Model	Lap Time \downarrow	CTE \downarrow	Bd. Violations \downarrow
Track 1			
Ours w/o map	8.17 ± 0.59	27.63 ± 7.95	4.82 ± 8.64
Ours (AIS)	7.35 ± 0.42	26.34 ± 6.12	2.79 ± 5.98
Track 2			
Ours w/o map	5.47 ± 0.34	14.57 ± 4.92	1.39 ± 4.89
Ours (AIS)	4.89 ± 0.26	17.33 ± 3.48	0.30 ± 1.29
Track 3			
Ours w/o map	20.47 ± 0.91	61.32 ± 14.04	10.09 ± 14.06
Ours (AIS)	19.08 ± 0.89	56.14 ± 8.17	8.78 ± 10.82

TABLE II: Quantitative evaluation of the driving performance of our approach. The results show the efficiency of our approach, which significantly reduces the cross-track error while achieving faster lap times in comparison to a model without a mapping network.

of three metrics computed over 30 laps. In more detail, we present the average lap time, the average cross-track error (CTE), and the average lane-boundary violation score. The lap time is a useful performance indicator as wrong dynamics predictions either lead to overly careful driving or to overly aggressive maneuvers on slippery surfaces which results in deviations from the racing line or emergency braking. The other two performance indicators focus on measuring wrong predictions in the context of overly aggressive driving. The results in Tab. II suggest, that our surface-aware dynamics model significantly improves all stated metrics. Our surface-aware model achieves significant lower lap times. Furthermore, it yields reduced cross-track error and significantly reduces the violation of lane boundaries. Thus, our approach increases driving safety in challenging environments.

We observe that latent mapping helps in particular to improve the violation of lane boundaries as our model prevents unexpected drifting that leads off the racing track. We illustrate such a scenario in Fig. 3. Further, actions obtained from optimizing erroneous trajectory estimates sometimes led to failure cases on slippery surfaces. In these cases, we had to manually intervene to prevent damage to the car. This happened three times for the model without map updates, while we never observed it for the model with map updates. This behavior indicates that on slippery surfaces the model with latent surface information predicts the dynamics more accurately while the model without overestimates the speed at which certain corners can be taken as it is trained to predict well averaged over all surfaces including those for which such maneuvers are possible.

D. Progressive Map Building

We further investigate the quality of the latent map with respect to the number of driven laps. The mapping model can improve its estimate of the underlying latent vector and variance with every new lap. This experiment is conducted

Lap number	1	2	3	4	5	10
Lap time [s] ↓	20.89	20.62	20.08	19.44	19.32	19.10
L2 ₁₀ ↓	0.0857	0.0816	0.0807	0.0805	0.0804	0.0801

TABLE III: Quantitative evaluation of the influence of completed laps with respect to the dynamics prediction accuracy and lap time. In this experiment, we only consider map updates during a specified number of consecutive laps. The respective L2 error is calculated over the state transitions of all laps.

on Track 3 while driving ten laps. Tab. III shows the lap times and the L2 error that is computed by considering the state transitions of all laps but only using the observations of the first ten laps to build the latent map respectively. The results show that with each additional lap our mapping model is consistently improving the latent estimates to improve the state predictions as well as the driving performance.

VII. CONCLUSION

In this paper, we present a novel approach to estimate latent representations of the road surface. The latent representation is aggregated to a global grid map, which is then leveraged to improve future state predictions from a dynamics model. To this end, an ensemble of probabilistic dynamics models estimates the distribution of the next state given the current state, current action, and the latent representation of the road that corresponds to the current position of the vehicle. Thus, our dynamics model can effectively leverage the latent cues making it surface-aware. We show the efficacy of our method on a newly proposed dataset, presenting improved prediction accuracy of future states. Furthermore, we employ a model predictive controller and a novel reward function. In real-world driving experiments we demonstrate that including the latent mapping provides a significantly improved driving performance, reducing lap times and enhancing vehicle safety on surfaces with varying friction. While this work considers the use of surface information to improve the dynamics model, there are also other changing factors influencing the dynamics such as tire pressure and temperature. A promising avenue of future work is to integrate our approach in a more complete system with additional sensors to also take these factors into account.

REFERENCES

- [1] “Tesla autopilot,” <https://www.tesla.com/autopilot>, accessed: 2022-09-08.
- [2] “Waymo one,” <https://waymo.com/waymo-one>, accessed: 2022-09-08.
- [3] M. Cutler and J. P. How, “Autonomous drifting using simulation-aided reinforcement learning,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 5442–5448.
- [4] E. Velenis and P. Tsiotras, “Minimum time vs maximum exit velocity path optimization during cornering,” in *Proceedings of the IEEE International Symposium on Industrial Electronics, 2005. ISIE 2005.*, vol. 1, 2005, pp. 355–360.
- [5] I. Zubov, I. Afanasyev, A. Gabdullin, R. Mustafin, and I. Shimchik, “Autonomous drifting control in 3d car racing simulator,” in *International Conference on Intelligent Systems (IS)*, 2018, pp. 235–241.
- [6] M. Acosta and S. Kanarachos, “Teaching a vehicle to autonomously drift: A data-based approach using neural networks,” *Knowl. Based Syst.*, vol. 153, pp. 12–28, 2018.
- [7] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, “Fast, robust quadruped locomotion over challenging terrain,” in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 2665–2670.
- [8] J. Kolter, P. Abbeel, and A. Ng, “Hierarchical apprenticeship learning with application to quadruped locomotion,” *Advances in Neural Information Processing Systems*, vol. 20, 2007.
- [9] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, et al., “Stanley: The robot that won the darpa grand challenge,” *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.
- [10] A. Nagabandi, G. Yang, T. Asmar, R. Pandya, G. Kahn, S. Levine, and R. S. Fearing, “Learning image-conditioned dynamics models for control of underactuated legged millirobots,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4606–4613.
- [11] L. Hermansdorfer, J. Betz, and M. Lienkamp, “A concept for estimation and prediction of the tire-road friction potential for an autonomous racecar,” in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 1490–1495.
- [12] B. Huang, X. Fu, S. Wu, and S. Huang, “Calculation algorithm of tire-road friction coefficient based on limited-memory adaptive extended kalman filter,” *Mathematical Problems in Engineering*, 2019.
- [13] J. Zürn, W. Burgard, and A. Valada, “Self-supervised visual terrain classification from unsupervised acoustic feature learning,” *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 466–481, 2020.
- [14] A. Valada, L. Spinello, and W. Burgard, “Deep feature learning for acoustics-based terrain classification,” in *Robotics research*. Springer, 2018, pp. 21–37.
- [15] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, “Learning to adapt in dynamic, real-world environments through meta-reinforcement learning,” in *International Conference on Learning Representations*, 2018.
- [16] S. Khaleghian, A. Emami, and S. Taheri, “A technical survey on tire-road friction estimation,” *Friction*, vol. 5, no. 2, pp. 123–146, 2017.
- [17] Y. Wang, J. Hu, F. Wang, H. Dong, Y. Yan, Y. Ren, C. Zhou, and G. Yin, “Tire road friction coefficient estimation: review and research perspectives,” *Chinese Journal of Mechanical Engineering*, vol. 35, no. 1, pp. 1–11, 2022.
- [18] M.-H. Kim, J. Park, and S. Choi, “Road type identification ahead of the tire using d-cnn and reflected ultrasonic signals,” *International journal of automotive technology*, vol. 22, no. 1, pp. 47–54, 2021.
- [19] W. Yu, J. Tan, C. K. Liu, and G. Turk, “Preparing for the unknown: Learning a universal policy with online system identification,” in *Proceedings of Robotics: Science and Systems*, July 2017.
- [20] K. Lee, Y. Seo, S. Lee, H. Lee, and J. Shin, “Context-aware dynamics model for generalization in model-based reinforcement learning,” in *International Conference on Machine Learning*, 2020, pp. 5757–5766.
- [21] W. Zhou, L. Pinto, and A. Gupta, “Environment probing interaction policies,” in *International Conference on Learning Representations*, 2018.
- [22] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, “Efficient off-policy meta-reinforcement learning via probabilistic context variables,” in *International conference on machine learning*, 2019.
- [23] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, “A tutorial on the cross-entropy method,” *Annals of operations research*, vol. 134, no. 1, pp. 19–67, 2005.
- [24] K. Chua, R. Calandra, R. McAllister, and S. Levine, “Deep reinforcement learning in a handful of trials using probabilistic dynamics models,” *Advances in neural information processing systems*, 2018.
- [25] S. Müller, M. Uchanski, and K. Hedrick, “Estimation of the maximum tire-road friction coefficient,” *J. Dyn. Sys., Meas., Control*, vol. 125, no. 4, pp. 607–617, 2003.
- [26] C. Pinneri, S. Sawant, S. Blaes, J. Achterhold, J. Stueckler, M. Rolinek, and G. Martius, “Sample-efficient cross-entropy method for real-time planning,” in *Proceedings of the 2020 Conference on Robot Learning*, 2021, pp. 1049–1065.
- [27] L. Pineda, B. Amos, A. Zhang, N. O. Lambert, and R. Calandra, “Mbrl-lib: A modular library for model-based reinforcement learning,” *Arxiv*, 2021.
- [28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.