Differentiable Constrained Imitation Learning for Robot Motion Planning and Control

Christopher Diehl, Janis Adamek, Martin Krüger, Frank Hoffmann and Torsten Bertram

***Abstract*— Motion planning and control are crucial components of robotics applications like automated driving. Here, spatio-temporal hard constraints like system dynamics and safety boundaries (e.g., obstacles) restrict the robot's motions. Direct methods from optimal control solve a constrained optimization problem. However, in many applications finding a proper cost function is inherently difficult because of the weighting of partially conflicting objectives. On the other hand, Imitation Learning (IL) methods such as Behavior Cloning (BC) provide an intuitive framework for learning decision-making from offline demonstrations and constitute a promising avenue for planning and control in complex robot applications. Prior work primarily relied on soft constraint approaches, which use additional auxiliary loss terms describing the constraints. However, catastrophic safety-critical failures might occur in out-of-distribution (OOD) scenarios. This work integrates the flexibility of IL with hard constraint handling in optimal control. Our approach constitutes a general framework for constraint robotic motion planning and control, as well as traffic agent simulation, whereas we focus on mobile robot and automated driving applications. Hard constraints are integrated into the learning problem in a differentiable manner, via explicit completion and gradient-based correction. Simulated experiments of mobile robot navigation and automated driving provide evidence for the performance of the proposed method.**

I. Introduction

The motion of robots in the real world is constrained by the kinematics and dynamics of the robot as well as the geometric structure of the environment. For example, to navigate safely and smoothly, a self-driving vehicle (SDV) must consider various factors such as its control limits, stop signs, and obstacles building a driving corridor. A core challenge is incorporating these constraints into robot planning and control. That is also essential for automated driving traffic simulation to enhance the realism of the simulated agents. For instance, traffic agents must follow common road rules. On the one side, optimal control approaches solve a finite horizon optimal control problem by optimizing a cost function under *explicitly* defined constraints. A common approach, like in direct methods [1], is to derive a nonlinear program from a continuous optimal control formulation [2], [3], [4] and then solve the problem with numerical optimization. However, designing a general cost function remains an unsolved problem for inherently complex tasks such as automated driving [5], [6], [7]. Here, aspects like comfort and safety must be weighed against each other. On the other side, robot behavior can be learned from demonstrations, which is the task of IL. One example is BC, a simple *offline* learning method, requiring no *on-policy* environment interactions. Here, constraints are *implicitly* learned from data. Further, constraints can be integrated by auxiliary loss functions. However, there are no guarantees for constraint satisfaction, and robot policies fail under distribution shifts [8], causing unexpected unsafe actions.

This research was funded by the Federal Ministry for Economic Affairs and Climate Actions on the basis of a decision by the German Bundestag and the European Union in the project "KISSaF - AI-based Situation Interpretation for Automated Driving".

The authors are with the Institute of Control Theory and Systems Engineering, TU Dortmund University, D-44227, Germany.

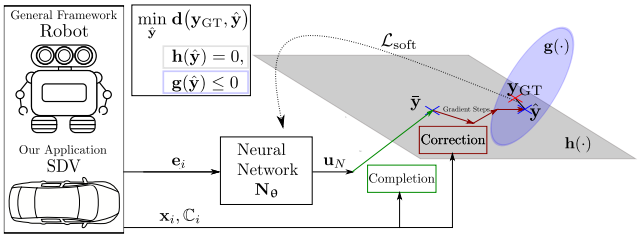


Fig. 1: A schematic overview of the proposed framework: A robot, like an SDV, perceives its environment and builds a high-dimensional environment model $\mathbf{e}_i$ and a low-dimensional state representation $\mathbf{x}_i$. Constraints $\mathbb{C}_i$ (grey rectangle: equality constraints, blue ellipse: inequality constraints) further bound the robots motion. A neural network $\mathbf{N}_{\boldsymbol{\theta}}$ processes $\mathbf{e}_i$ and outputs an initial sequence of control values $\mathbf{u}_N$. These are *completed* to the initial solution $\bar{\mathbf{y}}$, also containing the predicted states, by unrolling a robot dynamics model. Afterward, $\bar{\mathbf{y}}$ is *corrected* with gradient steps (red arrows), such that the estimated solution $\hat{\mathbf{y}}$ lies in the space defined by equality (grey) and the inequality constraints (blue) of $\mathbb{C}_i$. During training, the framework computes a distance measure between the $\hat{\mathbf{y}}$ and the ground truth $\mathbf{y}_{\mathrm{GT}}$ and backpropagates the softloss $\mathcal{L}_{\mathrm{soft}}$. During testing, the approach delivers a solution that imitates the expert behavior, while obeying a set of nonlinear constraints.

That raises the question: *Can we combine offline IL methods like BC with the constraint incorporation of optimal control methods?*

Donti et al. [9] present a method for incorporating hard constraints into the training of neural networks. The problem is formulated as a nonlinear program, and evaluated with a simple network architecture. Our approach extends their previous work to the robotic IL setting. The nonlinear program is constructed via direct transcription. Our proposed approach, summarized in Fig. 1, leverages two differentiable procedures to account for equality and inequality constraints and is agnostic to the used network architecture. First, the network predicts a sequence of control vectors, which

are explicitly completed to a sequence of states w.r.t. the system dynamics represented as equality constraints. Then, a gradient-based correction accounts for inequality constraints while satisfying the equality constraints.

Contributions. To summarize, the paper makes the following contributions: (i) It proposes a general Differentiable Constraint Imitation Learning (DCIL) framework for incorporating constraints, which is agnostic to the particular neural network architecture. (ii) It demonstrates the approach's effectiveness in one mobile robot and one automated driving environment during closed-loop evaluation. The approach outperforms multiple state-of-the art baselines considering a variety of metrics.

II. Related Work

The proposed approach is situated within the broader scope integrating constraints into learning-based approaches and IL in the robotics and automated driving literature. This section classifies related work into two major categories.

Modification of the Training Loss. The first class of approaches incorporates constraints by modifying the training loss. A simple approach adds the constraints as weighted penalties to the imitation loss. [10] proposes an application for automated driving. The work shows that additional loss functions penalizing constraint violations improve the closed-loop performance. [11] modifies the training process with a primal-dual formulation and converts the constrained optimization problem into an alternating min-max optimization with Lagrangian variables. [12] uses an energy-based formulation. During training, the loss pushes down the energy of positive samples (close to the expert demonstration) and pulls up the energy-values on negative samples, which violate constraints (e.g., colliding trajectories). While these methods are more robust to errors in constraint-specifications, they often fail in OOD scenarios as errors made by the learned model still compound over time. That can lead to unexpected behavior like leaving the driving corridor [8].

Projection onto Feasible Sets. The second group of approaches projects the neural network's output onto a solution that is compliant with the constraints. Instead of predicting a future sequence of states, a neural network predicts a sequence of controls [13]. Unrolling a dynamics model generates a feasible state trajectory consistent with the robot system dynamics. However, the approach does not account for general nonlinear inequality constraints. [14] presents an inverse reinforcement learning approach. First, a set of safe trajectories is sampled, and learning is only performed on the safe samples. SafetyNet [15] trains an IL planner and proposes a sampling-based fallback layer performing sanity checks. [16] proposes a similar approach using quadratic optimization. Other works incorporate quadratic programs [17] or convex optimization programs [18] as an implicit layer into neural network architectures. These approaches constitute the last layer to project the output to a set of feasible solutions. [19] directly modifies the network architecture by encoding convex polytopes. Sampling, quadratic optimization and convexity severely restrict the solution space.

Most closely related to our approach is the work of [9]. The authors present a hybrid approach, which accounts for nonconvex, nonlinear constraints. Experiments deal with numerical examples with simple network architectures. We extend this work to the real-world-oriented robot IL setting with more complex architectures for high-dimensional feature spaces. Further, we use an explicit completion by unrolling a robot dynamics model.

Just recently, concurrent works propose approaches which also incorporate nonlinear constraints using Signal Temporal Logic [20] and differentiable control barrier functions [21], which emphasizes the importance of using nonlinearities. In contrast, our approach relies on a differentiable completion, and gradient-based correction procedure, and the training is guided by auxiliary losses. [20] evaluates on simple toy examples, whereas our analysis considers a more realistic environment. [21] evaluates in real-world experiments but only use a circular robot footprint and object representation, whereas this work evaluates using different constraints. Moreover, our approach is able to resolve incorrect constraints that render the problem infeasible.

III. Problem Formulation

Assume robots dynamics described by nonlinear, time-invariant differential equations with time $t \in \mathbb{R}$, state $\mathbf{x} \in \mathcal{X}$ and controls $\mathbf{u} \in \mathcal{U} \subset \mathbb{R}^{n_\mathrm{u}}$:

$$\dot{\mathbf{x}}(t) = \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t)\big). \tag{1}$$

The state space size $\mathcal{X}$ of dimension n_x is the union of an arbitrary number of real spaces and non-Euclidean rotation groups $SO(2)$. In addition to the low-dimensional state representation $\mathbf{x}$, assume access to a high-dimensional environment representation $\mathbf{e} \in E \subset \mathbb{R}^{n_\mathrm{e}}$ (e.g., a birds-eye-view (BEV) image of the scene). Further, the system is bounded by a set of nonlinear constraints $\mathbb{C}$ (e.g., by control bounds, rules, or safety constraints).

A (sub-)optimal expert, pursuing a policy $\boldsymbol{\pi}_\mathrm{exp}$, controls the robot and generates a dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{u}_i, \mathbf{e}_i, \mathbb{C}_i)\}_{i=0}^{I}$ with $I \in \mathbb{N}^+$ samples. A future trajectory of length $H \in \mathbb{N}^+$ containing states and controls belonging to sample i is given by $\mathbf{y}_\mathrm{GT} = \left[\mathbf{x}_i^\mathrm{T}, \mathbf{u}_i^\mathrm{T} \ldots, \mathbf{x}_{i+H}^\mathrm{T}, \mathbf{u}_{i+H-1}^\mathrm{T}\right]^\mathrm{T}$. During training, the objective is to find the optimal parameters $\boldsymbol{\theta} \in \mathbb{R}^{n_\boldsymbol{\theta}}$ under a maximum likelihood estimation:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}\left[\mathbf{d}\big(\mathbf{y}_\mathrm{GT}, \hat{\mathbf{y}}\big)\right], \tag{2}$$

subject to equation (1) and the constraints $\mathbb{C}$. The function $\mathbf{d}$ denotes a distance measure and $\hat{\mathbf{y}} = \boldsymbol{\pi}_\boldsymbol{\theta}(\mathbf{x}_i, \mathbf{e}_i)$ is the output of the function $\boldsymbol{\pi}_\boldsymbol{\theta}$ parameterized by $\boldsymbol{\theta}$. Function $\boldsymbol{\pi}_\boldsymbol{\theta}$ is described by a neural network $\mathbf{N}_\boldsymbol{\theta}$ and the completion $\mathbf{f}_\mathrm{compl}$ and correction $\mathbf{f}_\mathrm{corr}$ procedure. During inference, given the environment representation, the robot's goal is to predict a sequence of states and controls compliant with the constraints. In the spirit of an model predictive control (MPC) framework, the first control vector is applied or an underlying tracking controller regulates the robot along the reference.

IV. Constrained Imitation Learning Synthesis

This section introduces the constrained IL framework. We first show how to construct a nonlinear program (NLP) per sample used for training the network. Afterwards, the solution process is detailed. A general description of the approach is visualized in Fig. 1.

A. Nonlinear Program Formulation

Direct transcription (see for example [1]) transforms the time-continuous formulation of the constraints $\mathbb{C}$ and Equ. (1) into a nonlinear program per sample. We discretize the time interval of the future with length H with $t_0 \leq t_1 \leq \cdots \leq t_k \leq \cdots \leq t_H$ and $k = 0, 1, \ldots, H$. We assume a piecewise constant control $\mathbf{u}(t) := \mathbf{u}_k =$ constant for $t \in [t_k, t_k + \Delta t)$, where $\Delta t = t_{k+1} - t_k$ for $k = 0, 1, \ldots, H-1$ denotes the time interval. The states at grid points t_k are described by $\mathbf{x}(t_k) := \mathbf{x}_k$ for $k = 0, 1, \ldots, H$. The forward differences

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \tag{3}$$

impose a set of equality constraints $\mathbf{h}(\mathbf{x}_{k+1}, \mathbf{x}_k, \mathbf{u}_k) = 0$. With a slight abuse of notation, we set $\mathbf{x}_0 = \mathbf{x}_i$, and at $t_0 = t_i$. Note that index i denotes the measured variables in the dataset, whereas index k describes the variables of the constrained optimization problem (4). Further, inequalities constraints $\mathbf{g}(\mathbf{x}_k, \mathbf{u}_k) \leq 0$ are constructed based on $\mathbb{C}_i$ and are only evaluated at the discrete time steps for $\mathbf{x}_k$ and $\mathbf{u}_k$.

The resulting NLP per sample is given:

$$\begin{aligned} \min_{\hat{\mathbf{y}}} \quad & \mathbf{d}\big(\mathbf{y}_{\text{GT}}, \hat{\mathbf{y}}\big) \\ \text{subject to} \quad & \\ & \mathbf{h}(\mathbf{x}_{k+1}, \mathbf{x}_k, \mathbf{u}_k) = 0, \quad k = 0, 1, \ldots, H-1 \\ & \mathbf{g}(\mathbf{x}_k, \mathbf{u}_k) \leq 0, \qquad\quad k = 0, 1, \ldots, H-1 \\ & \mathbf{g}(\mathbf{x}_N) \leq 0, \end{aligned} \tag{4}$$

with optimization vector $\hat{\mathbf{y}} = \left[\mathbf{x}_0^{\mathrm{T}}, \mathbf{u}_0^{\mathrm{T}} \ldots, \mathbf{x}_H^{\mathrm{T}}, \mathbf{u}_{H-1}^{\mathrm{T}}\right]^{\mathrm{T}}$.

Remember that $\hat{\mathbf{y}}$ is a function of the parameters $\boldsymbol{\theta}$. Hence, the complete procedure must be differentiable in order to backpropagate the gradients. The next section will describe such an approach using a modified version of [9].

B. Explicit Equality Completion

Instead of directly regressing a trajectory of future states, it is a common practice [13] to output a sequence of control vectors and unroll a differentiable dynamics model. That is similar to the explicit completion procedure described by [9]. To be precise, the neural network $\mathbf{N}_{\boldsymbol{\theta}}$ predicts a sequence of control vectors $\mathbf{u}_N$. The sequence of states $\mathbf{x}_N$ is then computed by iteratively applying Equ. (3), starting from the measured state $\mathbf{x}_i$, described by function $\mathbf{x}_N = \mathbf{f}_{\text{compl}}(\mathbf{u}_N, \mathbf{x}_i)$. The concatenation of both vectors results in $\bar{\mathbf{y}} = \left[\mathbf{u}_N^{\mathrm{T}}, \mathbf{x}_N^{\mathrm{T}}\right]^T$

C. Inequality Correction

The completion process accounts for the equality constraints derived from the discretized robots system dynamics. To further consider the inequality constraints, a differentiable gradient-based correction procedure is applied [9]. Here, we take gradient steps along the manifold of states and controls satisfying the equality constraints towards a feasible region.

The gradient-based correction, described by function $\mathbf{f}_{\text{corr}}(\bar{\mathbf{y}})$, is initialized by $\bar{\mathbf{y}} = \left[\mathbf{u}_N^{\mathrm{T}}, \mathbf{x}_N^{\mathrm{T}}\right]^T$. The approach then calculates the gradients of the inequality constraints w.r.t. the sequence of control vectors $\mathbf{u}_N$ and takes n_{grad} steps along the gradients. With the learning rate $\gamma \in \mathbb{R}^+$ and abbreviating $\mathbf{f}_{\text{compl}}(\cdot) = \mathbf{f}_{\text{compl}}(\mathbf{u}_N, \mathbf{x}_i)$ formally the function is given by:

$$\mathbf{f}_{\text{corr}}\left(\begin{bmatrix} \mathbf{u}_N \\ \mathbf{f}_{\text{compl}}(\cdot) \end{bmatrix}\right) = \begin{bmatrix} \mathbf{u}_N - \gamma \Delta \mathbf{u}_N \\ \mathbf{f}_{\text{compl}}(\cdot) - \gamma \Delta \mathbf{f}_{\text{compl}}(\cdot) \end{bmatrix}, \tag{5}$$

with gradients

$$\Delta \mathbf{u}_N = \nabla_{\mathbf{u}_N} \left\| \mathrm{ReLU}\left(\boldsymbol{\alpha} \odot \mathbf{g}\left(\begin{bmatrix} \mathbf{u}_N \\ \mathbf{f}_{\text{compl}}(\cdot) \end{bmatrix}\right)\right) \right\|_2^2, \tag{6}$$

and

$$\Delta \mathbf{f}_{\text{compl}}(\cdot) = \frac{\partial \mathbf{f}_{\text{compl}}(\cdot)}{\partial \mathbf{u}_N} \Delta \mathbf{u}_N. \tag{7}$$

Equ. (6) calculates the gradients of the inequality constraints $\mathbf{g}$ (depended on $\mathbf{u}_N$ and $\mathbf{x}_N$). $\mathbf{g}$ is weighted by $\boldsymbol{\alpha} \in \mathbb{R}^{\text{a}}$, with $\odot$ as the element-wise product. The norm is squared, leading to a quadratic penalty for inequality violations [2]. The ReLU only activates the penalty when the inequality is violated. For instance, the trajectory of an SDV not violating the lane bounds should not be corrected. The solution of the procedure[1] is $\hat{\mathbf{y}}$. The intuition is that the network provides a good initialization that, if at all, violates the constraints slightly. Afterward, $\mathbf{f}_{\text{corr}}(\bar{\mathbf{y}})$ corrects those initialization to satisfy all inequality constraints, such as safety constraints, e.g., lane boundaries. That procedure is similar to [22], which produces an initial trajectory using sampling-based optimization and fine-tunes it with gradient-based optimization. In contrast, our initialization is learned.

D. Training and Inference

As already noticed by [9], the convergence of gradient-based methods is not guaranteed and depends on initialization. However, if initialized closed to an optimum these methods are highly effective. The softloss for training[2],

$$\mathcal{L}_{\text{soft}} = \mathbf{d}\big(\mathbf{y}_{\text{GT}}, \hat{\mathbf{y}}\big) + \lambda_g \left\| \mathrm{ReLU}\left(\boldsymbol{\alpha} \odot \mathbf{g}(\hat{\mathbf{y}})\right) \right\|_2 + \lambda_h \left\| \mathbf{h}(\hat{\mathbf{y}}) \right\|_2, \tag{8}$$

enables a feasible or at least nearly feasible initial solution, such that the inequality correction converges

[1]While $\mathbf{f}_{\text{corr}}$ respects the equality constraints $\mathbf{h}$, it could lead to violations of them. However, empirically, we found that penalizing $\mathbf{h}$ in Equ. (8) led to mean equality violations in the order of $1\text{e}{-5}$, which seems neglectable in our application.

[2]The loss in Equ. (8) described in the paper of [9] squares the norms of constraint violations. However, the official implementation (https://github.com/locuslab/DC3) uses the same loss as in this work. The authors of [9] verified that the mentioned implementation was used to generate the results of their paper. As later discussed in Section V-E, loss (8) also produced better results in our experiments.

during test time. $\lambda_g \in \mathbb{R}$ and $\lambda_h \in \mathbb{R}$ are weighting factors. Algorithm 1 summarizes the approach.

Algorithm 1 Deep Constraint Imitation Learning

procedure DCIL($\mathbf{e}_i, \mathbf{x}_i$)
  compute initial sequence of controls $\mathbf{u}_N = \mathbf{N}_\theta(\mathbf{e}_i)$
  complete to $\bar{\mathbf{y}} = \left[\mathbf{u}_N^{\mathrm{T}}, \mathbf{x}_N^{\mathrm{T}}\right]^T$ with $\mathbf{f}_{\text{compl}}(\mathbf{u}_N, \mathbf{x}_i)$
  correct to estimated solution $\hat{\mathbf{y}} = \mathbf{f}_{\text{corr}}(\bar{\mathbf{y}})$ (function applied n_{grad} times)
  if train **then**
    compute loss (8) and **update** θ
  else
    return $\hat{\mathbf{y}}$
  end if
end procedure

V. Experimental Evaluation

This section evaluates the proposed approach in one mobile robot and one automated driving environment. This section addresses the following research questions: *Q1*: Does the approach improve the closed-loop performance of IL methods? *Q2*: How does the approach deal with incorrect constrain specifications?

Environments. The environments used for evaluation are visualized in Fig. 2a) and b).

Mobile Robot Environment (MRE): In the first environment data is collected by controlling a mobile robot with radius $r_{\text{robot}} = 1\text{m}$ using the Dynamic Window Approach [23]. The task during demonstrations is to navigate from a random start to a random goal location in the shortest time possible while avoiding collisions with circular shaped obstacles. Objects are randomly located with varying radii $r_{\text{c}} \in [0.1, 3]\,\text{m}$. The dataset contains 838 episodes (69638 samples), which were spitted in 83.3% training, and 8.3% validation and test each samples. This work evaluates closed-loop on another 76 *unseen* test episodes.

Self-Driving Environment (SDE): The second environment uses CARLA, a realistic automated driving simulator [24]. The CARLA Roach agent [25] collects training data. Further additive noise is applied to make the demonstrations more diverse, but sub-optimal. The dataset contains 120 episodes (174275 samples) from Town01, using the same ratios as in the MRE. We test on 25 random routes from a *different* environment (Town02) using the scenarios of the CARLA NoCrash-Challenge (Empty) [26] following the standardized evaluation protocol of the CALRA leaderboard. The SDV's task it to follow the routes while avoiding collisions and obeying traffic lights.

Baselines. This work benchmarks against the following baselines. *IL*: The traditional imitation learning directly regresses a future state trajectory. *DKM*: An IL approach [13] predicting a sequence of control vectors bounded by a sigmoid layer. The controls and a dynamics model are then used to unroll a future state trajectory. *DKM*$\leq$: DKM with an additional gradient-based correction procedure only applied during test time, similar to a fallback layer as in [15]. *SL*: The same approach as IL trained using the softloss $\mathcal{L}_{\text{soft}}$. That is similar to [10], but here the soft constraints are not computed in image space.

Metrics. The MRE uses the following metrics: *Goal Reaching Rate* (GRR): Rate of reached goals. *Collision Rate* (CR): Rate of collision-prone episodes. *Time*: Percentage of the agents completion time relative to the expert trajectory. This metric measures efficiency. *Kinematic Constraint Violations* (KCV): Summed number of constraint violations (tolerance: $1\text{e}{-4}$) of velocity, angular velocity, acceleration, and angular acceleration.

The SDE uses the metrics of the official CARLA Leaderboard Benchmark as described in [25]. We focus on closed-loop metrics as open-loop metrics can be a poor indicator to the actual task performance of robot policies [21].

A. Implementation

The method is agnostic and not restricted to the specific design choices made here.

State and Controls. A robot-centric coordinate system describes the state. *MRE*: The state $\mathbf{x}$ is defined by a 2-D position with $x, y \in \mathbb{R}$ and $\phi \in SO(2)$. The robot controls $\mathbf{u}$ are described by a forward $v \in \mathbb{R}$ and rotational velocity $\omega \in \mathbb{R}$. During testing, the flatness property of the unicycle model is used to compute the control values based on the predicted state trajectory. One could also directly use the predicted control values in a MPC formulation. However, the IL baseline only predicts a state sequence. Therefore, for a fair comparison, DCIL uses the same control strategy.

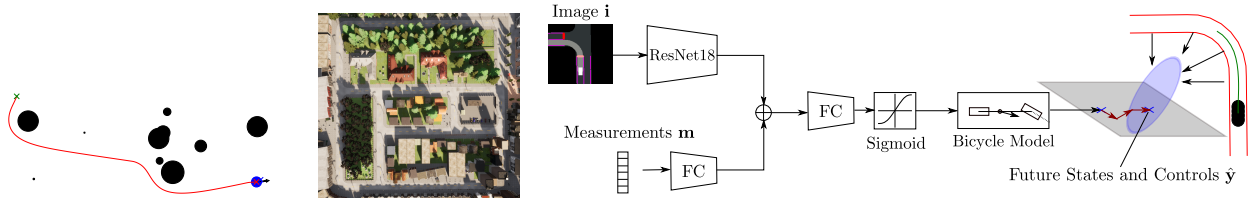
SDE: State $\mathbf{x}$ and controls $\mathbf{u}$ are the same as in [13]. During the evaluation, two PID controllers track the predicted state sequence of all methods.

Inputs. In both environments the input of the neural network $\mathbf{e}_i$ constitutes of an image $\mathbf{i} \in \mathbb{R}^{a \times b \times c}$ with resolution $\text{res} \in \mathbb{R}$ and a measurement vector $\mathbf{m} \in \mathbb{R}^{n_{\text{m}}}$, which is a common representation in automated driving applications [13], [7].

MRE: The robot centric image has dimensions $a = b = 128\,\text{px}$ with $\text{res} = 10\,\frac{\text{px}}{\text{m}}$. One channel $c = 1$ describes the binary occupancy information. $\mathbf{m}$ contains the current v and ω. It is further described by the distance $d_{\text{goal}} \in \mathbb{R}$ and heading $\theta_{\text{goal}} \in SO(2)$ w.r.t. the goal point. The dimension of the estimated control and state sequence $\hat{\mathbf{y}}$, is defined by $H = 10$ with time interval $\Delta t = 0.3\,\text{s}$.

SDE:: The image has dimensions $a = b = 192\,\text{px}$ with $\text{res} = 5\,\frac{\text{px}}{\text{m}}$. The SDV is centered in all images at $40\,\text{px}$ above the bottom. Different semantic classes from the work of [25] are color coded using the RGB channels with $c = 3$ as visualized in 2c). $\mathbf{m}$ contains the current 2-D velocity $\mathbf{v} \in \mathbb{R}^2$, acceleration $\mathbf{a} \in \mathbb{R}^2$, and current speed limit v_{max}. We set $H = 20$ and $\Delta t = 0.2\,\text{s}$.

Constraints. *MRE:* For the dynamics, which constitute the equality constraints, we use a unicycle model as in [27]. This work applies box constraints such that $v \in [-0.5, 1]\frac{\text{m}}{\text{s}}$, $\omega \in [-0.70, 0.70]\frac{\text{rad}}{\text{s}}$, $a \in [-0.2, 0.2]\frac{\text{m}}{\text{s}^2}$,$\dot{\omega} \in [-0.70, 0.70]\frac{\text{rad}}{\text{s}^2}$. The acceleration a and angular acceleration $\dot{\omega}$ are computed from finite differences. For collision avoidance, we compute euclidean distances d_{obst} between the robot and the obstacles, as



(a) (b) (c)

Fig. 2: (a) Mobile robot environment. Red visualizes a demonstration trajectory navigating from the green start point to the blue goal region, avoiding random obstacles (black). (b) Self-driving environment. (c) Network architecture in the SDE.

both footprints are circles. Then the state of every predicted time step is constrained by $d_{\text{obst}} > r_{\text{robot}} + r_{\text{c}} + 0.1\,\text{m}$. As the algorithm requires a fixed number of constraints, it uses the three closest obstacles in the front half level of the robot.

SDE: In the CARLA experiments, this work uses the extended bicycle model for the dynamics (3) as [13] with vehicle sizes of a Lincoln MKZ. We bound the velocity by $v_{\max} = 8.33\,\frac{\text{m}}{\text{s}}$. Further the control accelerations are constraint by $a \in [-8, 4]\,\frac{\text{m}}{\text{s}^2}$, and the control steering angles by $\delta \in [-1, 1]\,\text{rad}$. For collision avoidance, this work constructs a polyline-based driving corridor as in [28] using the high-level route. Four circles approximate the vehicle footprint. At every gradient step, the algorithm estimates the shortest distance to the left and right lane boundary for every predicted time step k and every circle. Further logical constraints for traffic lights are imposed. If a traffic light is yellow or red, it constructs a stop line in front of the vehicle. Otherwise, due to the required fixed number of constraints, this line is created far away not affecting the correction step. For the stop line and the driving corridor, the inequality constraints are described as point-line distances. For a visual example of the constraints refer to Fig. 2c).

Loss. *MRE*: Let subscript $\hat{}$ define the estimated state and control of $\mathbf{y}$. Based on related work [13], losses are:

$$\mathbf{d}(\mathbf{y}_{\text{GT}}, \hat{\mathbf{y}}) = \sum_{k=1}^{H} (\hat{x}_k - x_{k,\text{GT}})^2 + (\hat{y}_k - y_{k,\text{GT}})^2 + \Big(\cos(\hat{\phi}_k) - \cos(\phi_{k,\text{GT}})\Big)^2 + \Big(\sin(\hat{\phi}_k) - \sin(\phi_{k,\text{GT}})\Big)^2 . \tag{9}$$

SDE: [29] showed that it its also beneficial to use an regularization term in the form of a inverse dynamics model. We follow this approach by adding the term to Equ. (9).

Network Architecture and Parameters. *MRE*: The binary image $\mathbf{i}$ is encoded by a LeNet [30] outputting a latent vector, which is concatenated with the measurement vector $\mathbf{m}$. Afterwards, the result is processed by the same 2-layer fully connected network (FCN) of [9]. The predicted control sequence is bounded by a sigmoid layer and passed to the completion and correction step. In both environments we choose the hyperparameters by grid searches. In MRE we choose: $\lambda_g = \lambda_h = 0.5$, $\boldsymbol{\alpha} = \mathbb{1}$.

SDE: A ResNet18 [31] first encodes the RGB image $\mathbf{i}$ and the measurement vector $\mathbf{m}$ is encoded by a 1-layer FCN . The concatenation of both encodings is passed to the previously described 2-layer FCN. Fig. 2c) visualizes the network architecture. In SDE we choose: $\lambda_g = 5$, $\lambda_h = 5$, $\lambda_u = 1$. Vector $\boldsymbol{\alpha}$ is defined by weighting factors for the different inequality constraints. Collision is weighted by factor $\alpha_{\text{c}} = 1$, stopping line violations by $\alpha_{\text{s}} = 2$ and bounds on kinematic values by $\alpha_{\text{k}} = 1$. Both experiments use $\gamma = 1\text{e}-3$ and $n_{\text{grad}} = 5$. For a fair comparison, we ran grid searches for all baselines.

TABLE I: Closed-loop performance of all methods using unseen test scenarios the in mobile robot environment.

Methode	GRR	CR	Time	KCV
	[%], ↑	[%], ↓	[%], ↓	[% (#)], ↓
IL	**100**	3.94	106	7.20 (4600)
SL	92	6.57	117	2.06 (1317)
DCIL	**100**	**0.00**	**105**	**0.12** (**89**)

B. MRE Results

To answer *Q1*, DCIL is compared against the described baselines. Table I visualizes the results. DCIL outperforms all baselines in all metrics and it is the only one, which reaches all goals and without any collisions. Moreover, compared to the normal IL baseline, the number of kinematic constraint violations is reduced by a factor of 51.69. A qualitative comparison in an exemplary scenario is visualized in Fig. 3a). DCIL is the only method planning a collision-free trajectory. This can be attributed to the correction procedure acting as a safety layer.

C. SDE Results

Again considering question *Q1*, refer to the quantitative comparison of Table II. Note that IL++ describes the same approach as IL, but uses a PID controller which takes more time to tune, such that the evaluation favors IL++. However, DCIL performs best in the different metrics of the CARLA leaderboard. We observed that the other baselines (IL, DKM, SL) often plan trajectories, that divert from the route or onto the opposite. That is explained by their behavior under distribution shifts. Fig. 3c) illustrates such an qualitative result using the IL method. The closed-loop metrics[3] in Table II also underline the described failures of the baselines.

Let us consider question *Q2*. In the CARLA experiments, the bicycle model [13] is an approximation of the real vehicle dynamics. However, it serves as an *inductive bias*,

[3]The closed-loop performance also depends on the underlying tracking controller. Even if the planned trajectory obeys all constraints, an inadequate PID controller could lead to lane boundary violations or red light infractions. For instance, DCIL violates one red traffic light.

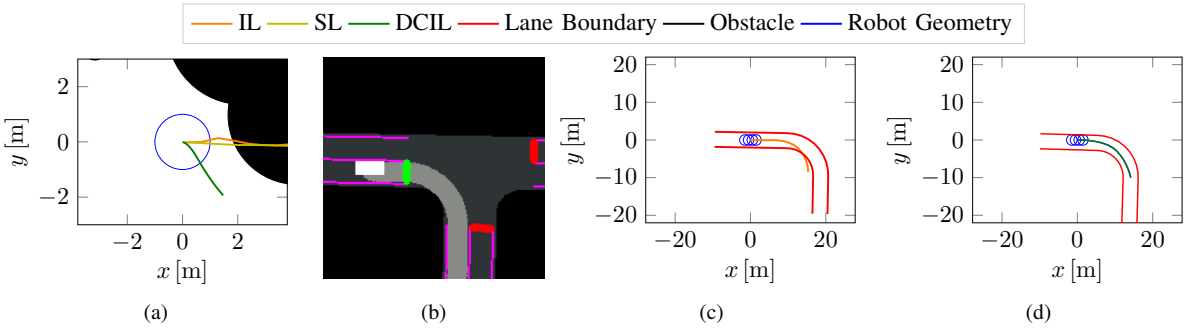


Fig. 3: Qualitative comparison. (a) Open-loop prediction results of the different methods in the MRE. (b) BEV input image of the neural network representing in CARLA. White pixels denote the SDV, black static obstacles, dark grey the road, light grey the route, and violet lane markings. Traffic lights are visualized by green or red color. The image is rotated by 90 degree (c) Constraint plot of the IL agent during closed-loop control. (d) DCIL during closed-loop control.

TABLE II: Closed-loop performance of all methods using unseen test routes in Town02 from CARLA-NoCrash. ↓ indicates a lower number is better and ↑ vice versa. Bold numbers indicate the best results.

Methode	Sucess rate	Driving score	Route completion	Infraction penalty	Collision layout	Red light infraction	Agent blocked	Outside of lane	Wrong lane
	[%], ↑	[%], ↑	[%], ↑	[%], ↑	[#/Km], ↓	[#/Km], ↓	[#/Km], ↓	[#/Km], ↓	[#/Km], ↓
IL	36	50.18	44	95.65	1.44	0.42	525.00	0.06	5.73
IL++	52	62.33	80	98.84	5.70	0.27	94.28	3.39	9.41
DKM	76	76.69	92	98.94	1.40	0.11	6.90	**0.00**	8.85
DKM $\leq$	76	87,66	96	98.94	0.94	0.13	3.26	0.15	0.09
SL	92	96.09	**100**	**100.00**	0.35	**0.00**	**0.00**	**0.00**	0.21
DCIL	**96**	**97.40**	**100**	98.94	**0.22**	0.18	**0.00**	**0.00**	**0.00**

simplifying the learning process, and enhancing generalization capabilities, as shown by the closed-loop results in Table II. To further answer *Q2*, this work conducts another experiment, in which the SDV is spawned onto the wrong lane (Fig. 4b). Note that the SDV never encountered such a situation during training, and hence this initial state is entirely outside the manifold of the training data. That situation could occur due to disturbances during driving or because a parked vehicle blocks the lane. Here, some hard constraint methods as [3] provide no solution at all, as the initial state is already infeasible. However, DCIL is robust w.r.t. incorrect specifications, softens the constraints and leads the vehicle back onto the right lane (Fig. 4c).

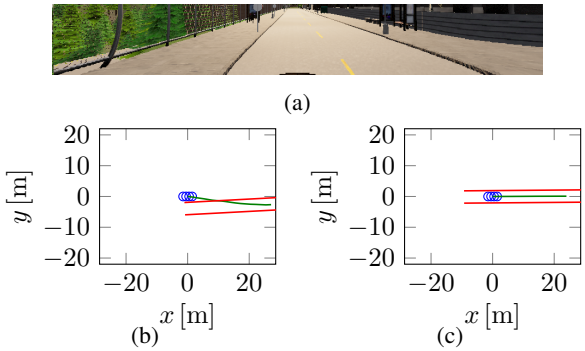


Fig. 4: Experiment, in which the SDV is spawned in an infeasible OOD state. (a) Camera image of the scene. (b) Initial infeasible configuration at $t = 0$s. (c) DCIL successfully leads the vehicle back to the right lane at $t = 4$s.

D. Runtime

The experiments use a AMD Ryzen 9 5900X and a Nvidia RTX 3090. Our non-optimized python implementation takes on average 20.02 ms in the MRE and 119.22 ms in the SDE. The runtime for the pure IL (SDE) is 37.20 ms.

E. Discussion, Limitations and Future Work

This section discusses the limitations of the presented work and gives an outlook on possible future directions. First, many active constraints make the loss landscape challenging to optimize and the procedure can be trapped in local minima. We observed that results of SL and DCIL get worse (SDE: both methods' driving score drops by about 15 %) using the described squared loss of [9]. That can be explained by the fact that the squared loss is more sensitive to outliers and harder to optimize during training, especially in the SDE. Hence we decided to use the non-squared loss of the official implementation [9] as mentioned in Section IV-D. Second, we marginalize over agents and plan the constrained unimodal motion of a single vehicle. Future work should extend the approach to constrained joint planning of multi-modal futures with multiple agents, similar to [32], and evaluate the resulting traffic simulation with real-world data [33].

VI. Conclusion

This work combined ideas from IL and optimal control for motion planning and control. It accounts for constraints using a differentiable completion and correction procedure. The experiments revealed that our approach outperforms multiple baselines in one mobile robot and one automated driving environment, and can deal with infeasible initial states.

References

[1] J. Betts, "Practical methods for optimal control and estimation using nonlinear programming. 2nd ed," 01 2010.

[2] C. Rösmann, F. Hoffmann, and T. Bertram, "Kinodynamic trajectory optimization and control for car-like robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 5681–5686.

[3] C. Rösmann, A. Makarow, and T. Bertram, "Online motion planning based on nonlinear model predictive control with non-euclidean rotation groups," in *2021 European Control Conference (ECC)*, 2021, pp. 1583–1590.

[4] C. Diehl, A. Makarow, C. Rösmann, and T. Bertram, "Time-optimal nonlinear model predictive control for radar-based automated parking," *IFAC-PapersOnLine*, vol. 55, no. 14, pp. 34–39, 2022, 11th IFAC Symposium on Intelligent Autonomous Vehicles IAV 2022.

[5] W. B. Knox, A. Allievi, H. Banzhaf, F. Schmitt, and P. Stone, "Reward (mis)design for autonomous driving," *Artificial Intelligence*, vol. 316, p. 103829, 2023.

[6] C. Diehl, T. Sievernich, M. Krüger, F. Hoffmann, and T. Bertram, "Umbrella: Uncertainty-aware model-based offline reinforcement learning leveraging planning," in *Advances in Neural Information Processing Systems, Machine Learning for Autonomous Driving Workshop*, 2021.

[7] C. Diehl, T. S. Sievernich, M. Krüger, F. Hoffmann, and T. Bertram, "Uncertainty-aware model-based offline reinforcement learning for automated driving," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 1167–1174, 2023.

[8] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 15. Fort Lauderdale, FL, USA: PMLR, 11–13 Apr 2011, pp. 627–635.

[9] P. L. Donti, D. Rolnick, and J. Z. Kolter, "DC3: A learning method for optimization with hard constraints," in *International Conference on Learning Representations (ICLR)*, 2021.

[10] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," in *Proceedings of Robotics: Science and Systems*, June 2019.

[11] Y. Nandwani, A. Pathak, Mausam, and P. Singla, "A primal dual formulation for deep learning with constraints," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[12] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun, "End-to-end interpretable neural motion planner," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[13] H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, J. Schneider, D. Bradley, and N. Djuric, "Deep kinematic models for kinematically feasible vehicle trajectory predictions," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 10 563–10 569.

[14] T. Phan-Minh, F. Howington, T.-S. Chu, M. S. Tomov, R. E. Beaudoin, S. U. Lee, N. Li, C. Dicle, S. Findler, F. Suarez-Ruiz, B. Yang, S. Omari, and E. M. Wolff, "Driveirl: Drive in real life with inverse reinforcement learning," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 1544–1550.

[15] M. Vitelli *et al.*, "Safetynet: Safe planning for real-world self-driving vehicles using machine-learned policies," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 897–904.

[16] J. Zhou, R. Wang, X. Liu, Y. Jiang, S. Jiang, J. Tao, J. Miao, and S. Song, "Exploring imitation learning for autonomous driving with feedback synthesizer and differentiable rasterization," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 1450–1457.

[17] B. Amos and J. Z. Kolter, "OptNet: Differentiable optimization as a layer in neural networks," in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70, 2017, pp. 136–145.

[18] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter, "Differentiable convex optimization layers," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[19] M. Brosowsky, F. Keck, O. Dünkel, and M. Zöllner, "Sample-specific output constraints for neural networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 8, pp. 6812–6821, May 2021.

[20] K. Leung and M. Pavone, "Semi-supervised trajectory-feedback controller synthesis for signal temporal logic specifications," in *2022 American Control Conference (ACC)*, 2022, pp. 178–185.

[21] W. Xiao, T.-H. Wang, R. Hasani, M. Chahine, A. Amini, X. Li, and D. Rus, "Barriernet: Differentiable control barrier functions for learning of safe robot control," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 2289–2307, 2023.

[22] A. Sadat, M. Ren, A. Pokrovsky, Y.-C. Lin, E. Yumer, and R. Urtasun, "Jointly learnable behavior and trajectory planning for self-driving vehicles," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 3949–3956.

[23] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[24] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the Conference on Robot Learning*, 2017, pp. 1–16.

[25] Z. Zhang, A. Liniger, D. Dai, F. Yu, and L. Van Gool, "End-to-end urban driving by imitating a reinforcement learning coach," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 15 222–15 232.

[26] F. Codevilla, S. Eder, A. M. Lopez, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," in *IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9328–9337.

[27] T.-C. Lee, K.-T. Song, C.-H. Lee, and C.-C. Teng, "Tracking control of unicycle-modeled mobile robots using a saturation feedback controller," *IEEE Transactions on Control Systems Technology*, vol. 9, no. 2, pp. 305–318, 2001.

[28] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for bertha — a local, continuous method," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, 2014, pp. 450–457.

[29] F. Janjoš, M. Dolgov, and J. M. Zöllner, "Self-supervised action-space prediction for automated driving," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, 2021, pp. 200–207.

[30] 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.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2016, pp. 770–778.

[32] C. Diehl, T. Klosek, M. Krüger, M. Murzyn, and T. Bertram, "On a connection between differential games, optimal control, and energy-based models for multi-agent interactions," in *International Conference on Machine Learning, New Frontiers in Learning, Control, and Dynamical Systems*, 2023.

[33] N. Montali, J. Lambert, P. Mougin, A. Kuefler, N. Rhinehart, M. Li, C. Gulino, T. Emrich, Z. Yang, S. Whiteson, B. White, and D. Anguelov, "The waymo open sim agents challenge," 2023, arxiv:2305.12032.