

# Collision Avoidance in Tightly-Constrained Environments without Coordination: a Hierarchical Control Approach

Xu Shen<sup>†</sup>, Edward L. Zhu<sup>†</sup>, Yvonne R. Stürz, and Francesco Borrelli

**Abstract**—We present a hierarchical control approach for maneuvering an autonomous vehicle (AV) in tightly-constrained environments where other moving AVs and/or human driven vehicles are present. A two-level hierarchy is proposed: a high-level data-driven strategy predictor and a lower-level model-based feedback controller. The strategy predictor maps an encoding of a dynamic environment to a set of high-level strategies via a neural network. Depending on the selected strategy, a set of time-varying hyperplanes in the AV’s position space is generated online and the corresponding halfspace constraints are included in a lower-level model-based receding horizon controller. These strategy-dependent constraints drive the vehicle towards areas where it is likely to remain feasible. Moreover, the predicted strategy also informs switching between a discrete set of policies, which allows for more conservative behavior when prediction confidence is low. We demonstrate the effectiveness of the proposed data-driven hierarchical control framework in a two-car collision avoidance scenario through simulations and experiments on a 1/10 scale autonomous car platform where the strategy-guided approach outperforms a model predictive control baseline in both cases.

## I. INTRODUCTION

Autonomous vehicles (AVs) have been the focus of significant research efforts in both academia and industry, where they have demonstrated notable potential in reducing the number of traffic incidents due to human-driver error. While autonomous driving tasks in well-structured environments, such as highway driving and lane changing [1] have largely been solved, more challenging tasks, such as driving in tightly-constrained environments, still poses open questions especially when other moving AVs and human driven vehicles are present.

The problem of collision avoidance, which is central to AV navigation, is nonconvex and NP-hard in general [2]. In tightly-constrained dynamic environments, additional challenges arise from these aspects: 1) nonlinear and non-holonomic vehicle dynamics, 2) non-convexity of environments and close proximity, and 3) change in obstacle configuration over time (motion of other human-driven or AVs).

Conventional search or sampling-based path planning methods, such as RRT, lattice planners, A\* and their variants [3], [4], [5] have been applied to the collision avoidance problem. However, these can suffer greatly from the

curse of dimensionality and offer no guarantees of feasibility in dynamic environments. Reachability analysis [6], [7] provides not-at-fault behavior with a parameterized set of trajectories, though these can be overly conservative in tightly-constrained spaces.

With the surge of machine learning and reinforcement learning (RL), there has been a strong interest in applying data-driven approaches in control. Among them, end-to-end planning [8], [9] constructs a direct map from perception to control but lacks interpretability and safety guarantees. Deep RL has also been used for collision avoidance [10], [11], but typically requires discretization of the action space [12] or relaxation of the vehicle dynamics constraints.

Recently, optimization-based algorithms such as Model Predictive Control (MPC) [13] have received special attention due to their ability to include the exact dynamics and safety constraints in the formulation of the control problem. In [14], a novel optimization-based collision avoidance (OBCA) approach obtains a smooth reformulation of the collision avoidance constraints with exact vehicle geometry.

In order to combine the advantages of different approaches, hierarchical frameworks for autonomous driving have been proposed in the literature. Such methods separate the problem into coarse path planning, maneuver choice, or parameter optimization at the higher strategic level, and trajectory planning, tracking, and control at the lower execution level [8], [3], [4], [15], [16]. As demonstrated in [17], MPC also displays great flexibility in leveraging data in a hierarchical manner, for example, by incorporating a learned “strategy set” to guide the receding horizon controller when navigating in an unknown environment.

In this work, we propose a hierarchical, data-driven, and strategy-guided control scheme to tackle the problem of motion planning in tight, dynamic environments. While our hierarchical framework is formulated for generic control of AVs in such environments, for simplicity, our presentation focuses on the specific scenario of collision avoidance between an ego vehicle (EV) and a target vehicle (TV). The more general case of pairwise collision avoidance between an EV and multiple TVs will be the subject of future investigation. We assume that the vehicles do not use any form of motion coordination such as in [18]. This scenario is highly relevant, e.g., AV parking in a mixed autonomy setting, where collision avoidance and minimizing task duration are critical, and greatly depends on the selection of a good high-level strategy while maneuvering around other vehicles. In fact, a simple “stop and wait” strategy can be highly inefficient and might never complete the task in a crowded space.

<sup>†</sup>Denotes equal contribution. The authors are with the Department of Mechanical Engineering, University of California at Berkeley, Berkeley, CA 94701 USA {xu\_shen, edward.zhu}@berkeley.edu

This was supported by the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 846421, the National Science Foundation under award No. 1931853, and the Office of Naval Research under grant No. N00014-18-1-2833

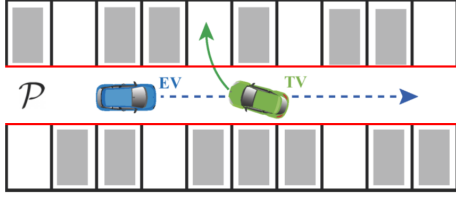


Fig. 1: The EV tracks a reference (dashed blue) in the local region  $\mathcal{P}$  while the TV executes a parking maneuver (solid green). The lane boundaries are marked in red.

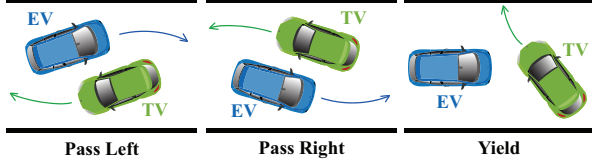


Fig. 2: Available strategies in  $\mathcal{S}$  for the EV navigation task. Our contributions are twofold:

- We propose a data-driven framework to construct a mapping from a high-dimensional environment encoding to a given set of high-level strategies. The latter is chosen such that it encodes prior knowledge of behavior in the corresponding scenario. Specifically, a neural network is trained offline with optimal collision-free trajectory rollouts, which are collected in a simulated environment. This mapping is used as a strategy predictor during online control.
- A strategy-guided time-varying MPC policy is formulated with exact vehicle and obstacle geometry to navigate a tightly-constrained dynamic environment. We refer to this policy as Strategy-Guided Optimization-Based Collision Avoidance (SG-OBCA). In addition to SG-OBCA, we also design a set of control policies which are selected based on the predicted strategy to maintain safety. The effectiveness of this control scheme is demonstrated through extensive numerical simulations and hardware experiments.

**Notation:**  $\mathcal{B}_n(c, r)$  denotes the  $l_2$ -norm ball centered at  $c \in \mathbb{R}^n$  with radius  $r > 0$ . The Minkowski sum of two sets  $\mathcal{A}$  and  $\mathcal{B}$  is denoted as  $\mathcal{A} \oplus \mathcal{B} = \{a + b : a \in \mathcal{A}, b \in \mathcal{B}\}$ . The minimum translation distance between two sets  $\mathcal{A}$  and  $\mathcal{B}$  is defined as  $\text{dist}(\mathcal{A}, \mathcal{B}) = \min_t \{\|t\| : (\mathcal{A} + t) \cap \mathcal{B} \neq \emptyset\}$  [19].  $\text{vec}(\cdot)$  and  $\text{vec}(\cdot, \cdot)$  denote vectorization and concatenation operators to flatten a sequence of matrices into a single vector.

## II. PROBLEM FORMULATION: STRATEGY-GUIDED COLLISION AVOIDANCE

We consider the problem of two-vehicle collision avoidance in a parking lot between a human-driven TV and an autonomous EV. In particular, the scenario of interest occurs in a local region of the vehicles' position space  $\mathcal{P} \subset \mathbb{R}^2$  and involves the TV executing a parking maneuver, while the EV seeks to navigate safely and efficiently through the narrow parking lot lane (Fig. 1).

In our scenario, the vehicles operate at low-speeds where tire slip and inertial effects can be ignored. We therefore

model the vehicle dynamics using the kinematic bicycle model [20] given by

$$\dot{z} := \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \cos(\psi + \beta) \\ v \sin(\psi + \beta) \\ \frac{v}{l_r} \sin(\beta) \\ a \end{bmatrix}, \quad u = \begin{bmatrix} \delta_f \\ a \end{bmatrix}, \quad (1)$$

where  $\beta = \tan^{-1}(l_r \tan(\delta_f)/(l_f + l_r))$  and the following nonlinear, time-invariant, discrete-time system with state  $z_k \in \mathcal{Z} \subseteq \mathbb{R}^4$  and input  $u_k \in \mathcal{U} \subseteq \mathbb{R}^2$  is obtained by discretizing (1) with the 4-th order Runge–Kutta method

$$z_{k+1} = f(z_k, u_k), \quad (2)$$

The geometry of the EV is modelled as a rotated and translated rectangle with length  $L$  and width  $W$ , given as  $\mathbb{B}(z_k) = R(\psi_k)\mathbb{B}_0 + [x_k, y_k]^\top$  with  $\mathbb{B}_0 := \{p \in \mathbb{R}^2 : Gp \leq g\}$ , where  $G = [I_2, -I_2]^\top$ ,  $g = [L/2, W/2, L/2, W/2]^\top$ ,  $p = (x, y) \in \mathbb{R}^2$  denotes the position states, and  $R(\psi_k) \in SO(2)$  is the rotation matrix corresponding to the angle  $\psi_k$ .

The TV and lane boundaries comprise the dynamic environment and are parameterized by  $\eta_k \in \mathbb{R}^o$ . Specifically, we consider  $M$  time-varying obstacles in the position space which can be each described as compact polyhedrons, i.e.

$$\mathbb{O}_k^{(m)} := \{p \in \mathbb{R}^2 : A_k^{(m)} p \leq b_k^{(m)}\}, \quad m = 1, \dots, M,$$

where  $A_k^{(m)} \in \mathbb{R}^{q_m \times 2}$  and  $b_k^{(m)} \in \mathbb{R}^{q_m}$  are known matrices at time  $k$ , and  $q_m$  is the number of faces of the  $m$ -th polyhedron. The environment encoding  $\eta_k$  can then be constructed as  $\eta_k = \text{vec}(\mathbb{O}_k^{(1:M)}) = \text{vec}(A_k^{(1:M)}, b_k^{(1:M)})$ . In the examples presented in this paper, we have  $M = 3$ .

At every discrete time step  $k$ , we require that the EV remain collision-free with all  $M$  obstacles. This is expressed as the constraint

$$\text{dist}(\mathbb{B}(z_k), \mathbb{O}_k^{(m)}) \geq d_{\min}, \quad \forall m = 1, \dots, M, \quad (3)$$

which enforces a minimum safety distance  $d_{\min} > 0$  between the EV and obstacles. We assume that measurements and predictions of the time-varying obstacles are available over an  $N$ -step time horizon, i.e., at time step  $k$ , we have access to  $\eta_k = \{\eta_k, \eta_{k+1}, \dots, \eta_{k+N}\}$ , and that there is no mismatch between the predictions and realized trajectories of the time-varying obstacles. The navigation task is then defined as a constrained tracking problem for the reference  $z_k^{\text{ref}}$ .

Central to our approach is the selection a high-level strategy  $s_k$  from a finite and discrete set  $\mathcal{S}$ , which encode prior knowledge of the behavioral modes available to the EV. In the case of our two-vehicle scenario, we use the following high-level strategies, which are illustrated in Fig. 2.

- (i) **Pass Left:** The EV passes the TV from the left side;
- (ii) **Pass Right:** The EV passes the TV from the right side;
- (iii) **Yield:** The EV yields to the TV for safety.

*Remark 1.* The strategy set  $\mathcal{S}$  is provided by the designer and will require domain expertise in its construction. The problem of identifying salient strategies given a task is outside the scope of this work.

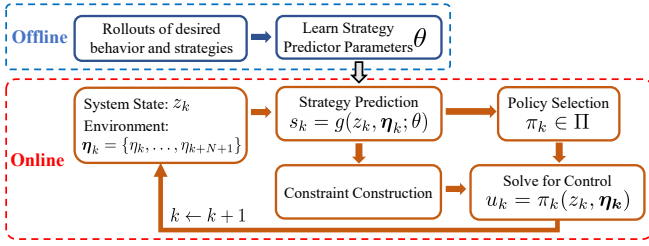


Fig. 3: The proposed strategy-guided control scheme



(a) Parking Lot and Recorded Maneuver (b) Subject Driving

Fig. 4: Data Collection in CARLA

### Summary of Approach

The constrained tracking task of interest is well-suited for MPC [13]. However, control horizon lengths, for which real-time MPC is possible, may result in myopic behavior leading to collision in tightly-constrained environments.

Our proposed strategy-guided control scheme for collision avoidance (illustrated in Fig. 3) attempts to address this issue by leveraging the high-level strategy as a surrogate for describing long-term behavior in the presence of other vehicles. In particular, we design a strategy predictor

$$s_k = g(z_k, \eta_k; \theta), \quad (4)$$

and train the parameters  $\theta$  in (4) using a database of human-driven parking maneuvers in a simulated environment, and locally optimal solutions to the EV navigation tasks corresponding to each of the TV maneuvers. We leverage the expressiveness of data-driven methods for strategy selection, as opposed to an end-to-end architecture directly for control as such grey-box approaches typically exhibit poor sample efficiency and require large amounts of data.

For the lower-level online strategy-guided control scheme, we construct three policies:

$$\Pi^{\text{SG}} = \{\pi^{\text{SG-OBCA}}, \pi^{\text{SC}}, \pi^{\text{EB}}\}, \quad (5)$$

where  $\pi^{\text{SG-OBCA}}$  is a nominal MPC controller based on OBCA [14], [21], which uses the selected strategy to generate hyperplane constraints which help reduce myopic behavior and improve the ability of the EV to complete its navigation task.  $\pi^{\text{SC}}$  is a safety controller which can be activated due to infeasibility of SG-OBCA or ambiguity in the behavior of the TV.  $\pi^{\text{EB}}$  is an emergency brake controller, to be triggered in the event of impending collision. Conditions for switching between the three policies are discussed in Section IV-C.

## III. OFFLINE LEARNING OF THE STRATEGY PREDICTOR

### A. Data Collection

We begin with raw data in the form of a set of human-driven parking maneuvers (Fig. 4a). These were collected us-

ing the CARLA simulator [22] in a custom parking lot [23], where the subject (driver) controls the brake, throttle, and steering of the vehicle (Fig. 4b). For each parking trial, the parking lot is randomly populated with static vehicles and the subject was asked to park into different spots either in forward or reverse.

For the  $i$ -th recording of length  $T^{[i]}$ , the environment encoding is constructed as  $\eta_{0:T^{[i]}}^{[i]} = \left\{ \text{vec} \left( \bigcirc_k^{(1:M), [i]} \right) \right\}_{k=0}^{T^{[i]}}$ , which contains the lane boundaries and the time-varying TV obstacles over the entire task. After obtaining  $\eta_{0:T^{[i]}}^{[i]}$ , a finite-time optimal control problem, formulated using the OBCA algorithm [14], was used to solve for the locally optimal, collision-free EV trajectory  $\{z^{[i],*}, u^{[i],*}\}$ .

Lastly, given the strategy set  $\mathcal{S} = \{\text{“Pass Left”}, \text{“Pass Right”}, \text{“Yield”}\}$ , the strategy label  $s^{[i]} \in \mathcal{S}$  for the  $i$ -th task can be generated automatically by looking at the relative configurations of the EV and TV over the EV solution trajectory, as illustrated in Fig. 2.

### B. Strategy Predictor Training

Using the method described in Sec. III-A, we collect  $K$  locally optimal TV-EV rollouts and their corresponding strategy labels offline. Now, given the horizon length  $N$ , the training dataset is constructed as  $\mathcal{D} = \{(X_0^{[1]}, s^{[1]}), \dots, (X_{T^{[1]}-N}^{[1]}, s^{[1]}), (X_0^{[2]}, s^{[2]}), \dots, (X_{T^{[K]}-N}^{[K]}, s^{[K]})\}$ . We note that the strategy labels remain constant within the  $i$ -th recording since the behavioral mode for each TV-EV rollout is fixed. Each data point  $X_k^{[i]}$  consists of the current state of the EV and the environment encoding along the  $N$ -step horizon, i.e.,  $X_k^{[i]} = \text{vec} \left( z_k^{[i],*}, \eta_k^{[i]} \right)$ ,  $k \in \{0, \dots, T^{[i]} - N\}$ . Using this dataset, we train a neural network (NN) for the predictor  $g(\cdot)$  in (4) to predict the strategy  $s_k$  from  $z_k$  and  $\eta_k$  at every time step. The network architecture is composed of a fully connected hidden layer with 40 nodes, tanh activation function, and a softmax output layer for 3 strategies in  $\mathcal{S}$ . The objective function we minimize is the cross-entropy loss for classification tasks.

## IV. ONLINE STRATEGY PREDICTION AND CONTROL

At time step  $k$  of the online task execution, the trained network with parameters  $\theta^*$  returns confidence scores over the strategy set  $\mathcal{S}$ , i.e.  $\hat{Y}_k = \text{NN}(z_k, \eta_k; \theta^*) \in \mathbb{R}^3$ . The predicted strategy is chosen to be the one with the highest score, i.e.,  $\hat{s}_k = g(z_k, \eta_k; \theta^*) = \arg \max_{s \in \mathcal{S}} \hat{Y}_k$ . By leveraging the predicted scores  $\hat{Y}_k$  and selected strategy  $\hat{s}_k$ , we proceed to construct time-varying hyperplanes and select the control policy  $\pi_k$  from the set  $\Pi^{\text{SG}}$ .

### A. Constructing Constraints from Strategies

Similar to [17], in this work, we view strategies as lower dimensional encodings of long term relative behavior between the controlled system and the dynamic environment. As such, we now describe how the chosen strategy at each timestep can be used to generate constraints for online control which help guide the EV into regions of space where it is more likely to successfully navigate around the TV.

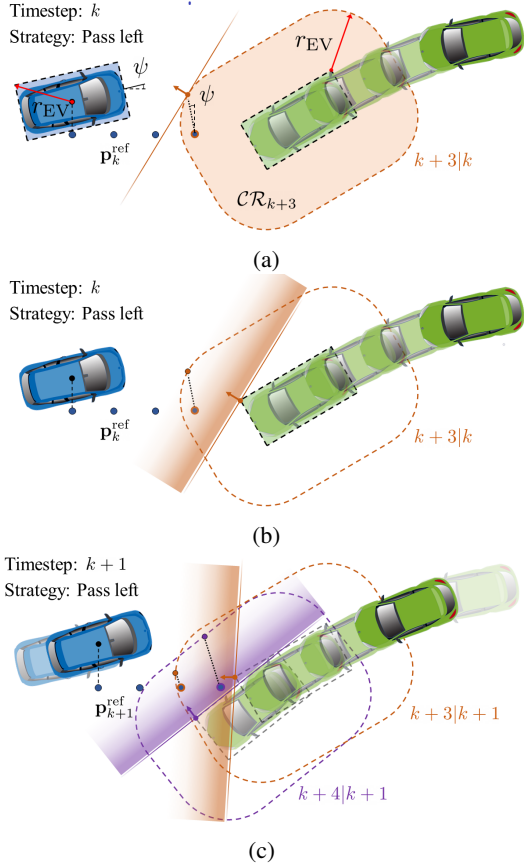


Fig. 5: Illustration of the strategy-guided constraint generation procedure with  $N = 3$  and strategy “Pass Left”. a) The position reference of the EV at timestep  $k+3$  falls within the critical region  $\mathcal{CR}_{k+3}$ . The reference point is projected to the boundary of the critical region in the positive body lateral direction and the tangent plane is found. b) The tangent plane is translated to be coincident with the TV boundary. c) Result of the constraint generation procedure at timestep  $k+1$ .

For the TV occupying the region  $\mathbb{O}_k^{\text{TV}}$  at timestep  $k$ , we define the critical region as:

$$\mathcal{CR}_k = \mathbb{O}_k^{\text{TV}} \oplus \mathcal{B}_2(0, r_{\text{EV}}),$$

where  $r_{\text{EV}} = \sqrt{L_{\text{EV}}^2 + W_{\text{EV}}^2}/2$  is the radius of the smallest disk which covers the extents of the EV as shown in Figure 5a. The critical region represents the area containing the TV where the collision avoidance constraints are close to or have become active.

Recall that we are interested in the tracking problem where we are given the  $N$ -step reference trajectory  $\mathbf{z}_k^{\text{ref}} = \{z_{k|k}^{\text{ref}}, \dots, z_{k+N|k}^{\text{ref}}\}$  at time step  $k$ . In the scenario presented in Section II, this is simply a constant velocity trajectory which follows the centerline of the lane. Denote the position states of the reference trajectory as  $\mathbf{p}_k^{\text{ref}}$ .

The main idea behind our constraint generation procedure is to project positions along the reference trajectory, which fall within the critical region (i.e.,  $p_{k+t|k}^{\text{ref}} \in \mathcal{CR}_{k+t}$  for some  $t \in \{0, \dots, N\}$ ), to the boundary of the critical region  $\partial\mathcal{CR}_{k+t}$ . The direction of projection is determined by the chosen strategy  $\hat{s}_k$  of either “Pass Left” or “Pass Right”.

In particular, the two aforementioned strategies correspond with the directions  $(-\sin \psi, \cos \psi)$  and  $(\sin \psi, -\cos \psi)$  respectively, i.e., the positive and negative body lateral axis directions in the inertial frame. Since the boundary of the critical region is smooth by definition, we can find a unique tangent plane at the projection point with an outward facing normal vector  $w_{k+t|k}(\eta_{k+t}, \hat{s}_k)$  as can be seen in Figure 5a. The final step involves translating the tangent plane such that it is coincident with the boundary of  $\mathbb{O}_k^{\text{TV}}$  as is shown in Figure 5b. This procedure results in a sequence of additional hyperplane constraints for time steps  $k+t$  where  $p_{k+t|k}^{\text{ref}} \in \mathcal{CR}_{k+t}$ ,  $t \in \{0, \dots, N\}$ . Denote the hyperplane parameters as  $\phi(\eta_{k+t}, \hat{s}_k) = -\text{vec}(w_{k+t|k}(\eta_{k+t}, \hat{s}_k), b_{k+t|k}(\eta_{k+t}, \hat{s}_k))$  and the augmented state  $\bar{z}_{k+t|k} = \text{vec}(z_{k+t|k}, 1)$ . The constraints can then be written as

$$\phi(\eta_{k+t}, \hat{s}_k)^\top \bar{z}_{k+t|k} \leq 0. \quad (6)$$

*Remark 2.* While we may also apply the constraint generation procedure to the “Yield” strategy, we choose to instead deal with it directly by activating a safety controller, which will be described in Sec. IV-C.

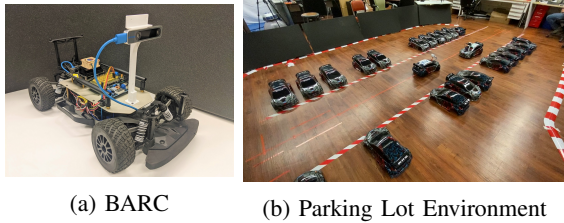
### B. Strategy-Guided Optimization-Based Collision Avoidance

For the nominal MPC policy  $\pi^{\text{SG-OBCA}}$  in  $\Pi^{\text{SG}}$ , we leverage the generated constraints to extend the OBCA algorithm [14] which formulates the collision avoidance constraints using exact vehicle geometry to enable tight maneuvers in narrow spaces. By introducing the dual variables  $\lambda^{(m)} \in \mathbb{R}^q$ ,  $\mu^{(m)} \in \mathbb{R}^4$  associated with the  $m$ -th obstacle, we obtain a smooth reformulation of the collision avoidance constraint in (3). The resulting NLP is written as follows:

$$\begin{aligned} \min_{\mathbf{z}, \mathbf{u}, \boldsymbol{\lambda}, \boldsymbol{\mu}} \quad & \sum_{t=0}^N c(z_{k+t|k}, u_{k+t|k}, z_{k+t|k}^{\text{ref}}) \\ \text{s.t.} \quad & z_{k+t+1|k} = f(z_{k+t|k}, u_{k+t|k}), \quad t = 0, \dots, N-1 \quad (7a) \\ & z_{k|k} = z_k, \quad (7b) \\ & \left( A_{k+t|k}^{(m)} t(z_{k+t|k}) - b_{k+t|k}^{(m)} \right)^\top \lambda_{k+t|k}^{(m)} - g^\top \mu_{k+t|k}^{(m)} > d_{\min} \quad (7c) \\ & G^\top \mu_{k+t|k}^{(m)} + R(z_{k+t|k})^\top A_{k+t|k}^{(m)\top} \lambda_{k+t|k}^{(m)} = 0 \quad (7d) \\ & \left\| A_{k+t|k}^{(m)\top} \lambda_{k+t|k}^{(m)} \right\| \leq 1, \quad (7e) \\ & \lambda_{k+t|k}^{(m)} \geq 0, \mu_{k+t|k}^{(m)} \geq 0, \quad (7f) \\ & \phi(\eta_{k+t}, \hat{s}_k)^\top \bar{z}_{k+t|k} \leq 0, \quad (7g) \\ & \forall t = 0, \dots, N, \quad m = 1, \dots, M, \end{aligned}$$

where  $c(z_{k+t|k}, u_{k+t|k}, z_{k+t|k}^{\text{ref}}) = \|z_{k+t|k} - z_{k+t|k}^{\text{ref}}\|_{Q_z}^2 + \|u_{k+t|k}\|_{Q_u}^2 + \|\Delta u_{k+t|k}\|_{Q_d}^2$  penalizes the deviation from the reference trajectory, input magnitude and rate, with  $Q_z \succeq 0$ ,  $Q_u \succ 0$ , and  $Q_d \succeq 0$ , respectively. (7a) are the dynamics constraints over the control horizon, (7b) sets the initial condition, (7c)-(7f) are the smooth and exact reformulations of (3) using Theorem 2 in [14], and (7g) are the strategy-guided hyperplane constraints. We call the NLP in (7): SG-OBCA.

*Remark 3.* Problem (7) is non-convex, therefore providing a good initial guess to warm-start the NLP solver is crucial to



(a) BARC (b) Parking Lot Environment

Fig. 6: Experimental Setup

Control Method	Failure Rate	Avg. # Iter	Median # Iter
Baseline	30%	213	220
<b>Strategy-Guided</b>	<b>6%</b>	<b>270</b>	<b>227</b>

TABLE I: Failure Rate and Iterations to Task Completion

maintain feasibility. For  $\mathbf{z}$  and  $\mathbf{u}$ , it is straightforward to use the solution from the previous iteration as the initial guess. The initial guess for the dual variables  $\lambda$ , and  $\mu$  are obtained with a similar method as described in [21].

### C. Policy Selection

In our proposed strategy-guided control scheme we defined the set (5), which contains three classes of policies<sup>1</sup>: SG-OBCA Control  $\pi^{\text{SG-OBCA}}$ , Safety Control  $\pi^{\text{SC}}$ , and Emergency Brake  $\pi^{\text{EB}}$ . The formulation of each policy and the selection criteria are as follows.

**1) SG-OBCA Control:** The SG-OBCA Control policy solves the problem (7) at every time step  $k$ .

*Selection Criteria:*

- (i) Problem (7) is solved successfully, **AND**
- (ii)  $\hat{s}_k \in \{\text{“Pass Left”}, \text{“Pass Right”}\}$  with high confidence, i.e.  $\max_s \hat{Y}_k \geq \xi$ , where  $\xi$  is a user-defined threshold.

**2) Safety Control:** The Safety Control policy regulates the relative states between the EV and the TV into a safe control invariant set with respect to the two vehicles. In this work, we define this set to be the states with zero relative speed, such that the distance between EV and TV is preserved.

*Selection Criteria:*

- (i) Problem (7) is infeasible, **OR**
- (ii)  $\hat{s}_k = \text{“Yield”}$  with  $\max_s \hat{Y}_k \geq \xi$ , **OR**
- (iii)  $\max_s \hat{Y}_k < \xi$

**3) Emergency Brake:** We define the Emergency Brake policy in the context of impending system failure. When the Emergency Break policy is selected, it means that the autonomous agents cannot resolve the situation by either  $\pi^{\text{SG-OBCA}}$  or  $\pi^{\text{SC}}$  and human intervention is necessary.

*Selection Criteria:* A collision is anticipated and cannot be resolved by either  $\pi^{\text{SG-OBCA}}$  or  $\pi^{\text{SC}}$ .

## V. SIMULATION AND EXPERIMENTAL RESULTS

In this section, we report results from simulations and experiments with the proposed strategy-guided control approach (SG) as described in Sec. IV. We compare the outcomes with a baseline control scheme (BL), where the  $\pi^{\text{SG-OBCA}}$  in (5) is replaced with  $\pi^{\text{BL-OBCA}}$  which solves

<sup>1</sup>Due to space constraints, the presented policy classes and their corresponding selection criteria are an abstraction of the actual implementation. Please visit [bit.ly/data-sg-control](http://bit.ly/data-sg-control) for full details.

(7) without the additional constraints (7g). Moreover, for the baseline control scheme,  $\pi^{\text{SC}}$  is selected only when  $\pi^{\text{BL-OBCA}}$  is infeasible. In both simulation and hardware experiments, we chose a horizon length of  $N = 20$  with a time step of  $dt = 0.1s$ . The safety distance  $d_{\min}$  in (7c) is set to 0.01m and we set the confidence threshold to be 0.75. As described in Section III, we recorded a total of 486 navigation tasks with TV maneuvers from which we generate 103, 553 labeled examples for the training dataset  $\mathcal{D}$ . Details can be found at: [bit.ly/data-sg-control](http://bit.ly/data-sg-control).

All tasks are attempted by both SG and BL control schemes in a MATLAB simulation environment. Table I reports the failure rate, where a collision occurs or the  $\pi^{\text{EB}}$  policy is selected, and the number of iterations required to complete the task. We can see that in simulation, the SG control scheme reduces the failure rate significantly, without sacrificing much in terms of task performance.

Both BL and SG approaches were implemented with the same cost matrices  $Q_z = \text{diag}([1, 1, 1, 10])$ ,  $Q_u = \text{diag}([1, 1])$ , and  $Q_d = \text{diag}([50, 50])$ , on the 1/10 scale open source Berkeley Autonomous Race Car (BARC) platform in a laboratory parking lot environment (Fig. 6). The cars are equipped with an Intel RealSense T265 tracking camera for localization and an NVIDIA Jetson Nano for onboard computation. We use FORCESPRO [24], [25] to compile the NLP solver, which achieves an average solution time of 30ms for problem (7). 99.8% of feasible solutions are returned in less than 100ms. The strategy and policy selection, constraint generation, and NLP solver all run on a host laptop with an Intel i9 processor and 32 GB of RAM. Communication with the cars is done through the Robot Operating System (ROS). The human driven TV was represented by another car which tracked the recorded trajectories using nonlinear MPC. The MPC predictions at each time step are used to construct  $\mathbb{O}_k^{\text{TV}}$  over the EV control horizon.

Results from the first navigation task are presented in Figs. 7a and 7b where the TV parks into an empty spot in the top row. The predicted strategy and selected policy over the task are visualized in Fig. 7b. It is clear that at the beginning of the task (before  $t = 4s$ ), the intention of the TV is ambiguous and the confidence in the “Pass Left” strategy is slightly higher because the TV veers to the right hand side of the EV. However, once the TV begins its parking maneuver, the “Pass Right” strategy is identified with high confidence score. Under SG, the EV is able to leverage the hyperplane constraints generated via the identified strategy to preemptively begin its maneuver around the TV, thereby resulting in a smoother speed and input profile compared to BL as can be seen in Fig. 7b. In contrast, BL performs a more aggressive and dangerous maneuver as it is only constrained by the safety distance  $d_{\min}$ . This results in a fragile policy which can easily lead to constraint violation when the EV is in the critical region. Indeed, in Fig. 7b, we observe that the BL triggers the Safety Control policy for several time steps due to infeasibility caused by the aggressive behavior.

Results from a second navigation task are presented in Figs. 7c and 7d where the TV parks in reverse into an empty

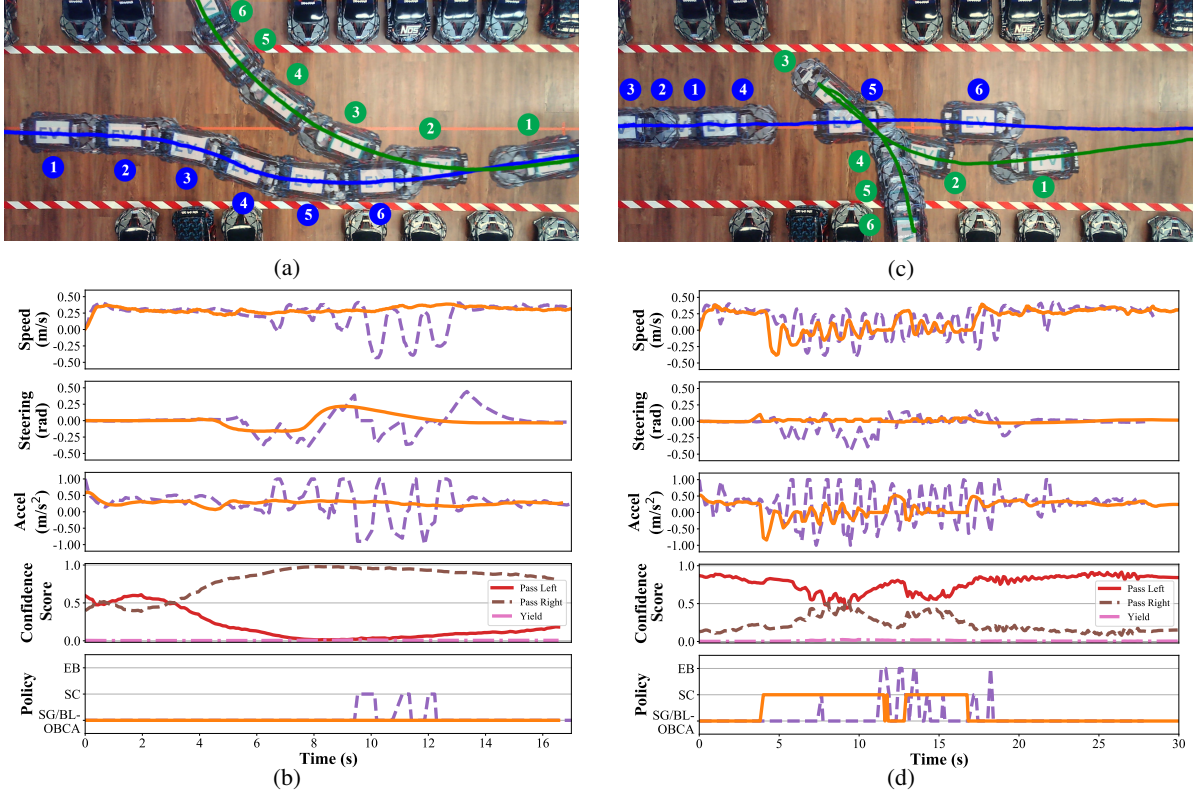


Fig. 7: a), c) Trajectories of the EV (blue) and TV (green) under the strategy-guided control scheme. Frame numbers are marked with circles; b), d) (From top to bottom) Speed and input profile of the EV under SG (solid orange) and BL (dashed purple). Confidence of strategy prediction and policy selection for the strategy-guided controller.

spot in the bottom row and a gear change is needed. The strategy and policy selections over the task are visualized in Fig. 7d, where the SG control scheme selects the Safety policy  $\pi^{\text{SC}}$  at an early stage and maintains a safe distance for a considerable amount of time due to low confidence in the selected strategy. This is caused by the time the TV spends idling during the gear change. In contrast, the short-sighted baseline controller again exhibits overly-aggressive behavior by trying to pass the TV while it is idle. This eventually leads to activation of the policy  $\pi^{\text{EB}}$  and collision with the TV. Finally, we again see in Fig. 7d that SG results in a smoother speed and input profile when compared to BL.

## VI. DISCUSSION AND CONCLUSION

In this work, we show that when compared to a baseline MPC approach, the proposed hierarchical data-driven control scheme significantly improves the success rate of a navigation task in a tightly-constrained dynamic environment. The design of a data-driven strategy predictor provides a structured and transparent approach to leveraging data in control, which is crucial to expanding the adoption of learning techniques in the control of safety critical systems.

The purpose of strategy-guided constraint generation and policy selection are to coerce the system towards regions of space where successful completion (in terms of recursive feasibility and stability) of the control problem is most likely. One interpretation of the strategy-guided constraints is that they are surrogates of the MPC terminal components,

where given the selected strategy, we attempt to construct constraints which drive the state of the system (2) into a set which is contractive towards the reference  $\mathbf{z}^{\text{ref}}$  while satisfying all constraints. Despite not affording the rigorous guarantees provided by an exact terminal cost function and terminal set, the value of our proposed surrogates lies in the fact that they can be straightforward to construct even when the exact formulations are intractable, which is typically the case for complex control tasks in dynamic environments.

In terms of the policy selection, predicting the strategy allows the system to “prepare” for an upcoming safety-critical encounter by selecting an appropriate control policy. In the context of this work, when faced with ambiguous behavior from the TV, the strategy-guided control approach can select the safety control policy, so that the system will only attempt the risky passing maneuver when it is confident enough to do so. In contrast, the baseline control scheme only reverts to the safety control when the nominal MPC becomes infeasible. This is usually at a time when the situation is already very challenging for the system to maintain safety.

Typical challenges due to uncertainty, brought about by localization errors, system delay, and model mismatch were encountered in the experimental setting. Therefore, future work will firstly focus on formulating a robust extension using both model-based and data-driven methods. Furthermore, applying the proposed approach to more complex environments with a higher number of dynamic objects or interactive/adversarial agents is of particular interest.

## REFERENCES

- [1] C. Pek, P. Zahn, and M. Althoff, "Verifying the safety of lane change maneuvers of self-driving vehicles based on formalized traffic rules," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1477–1483.
- [2] J. F. Canny, *The Complexity of Robot Motion Planning*. Cambridge, MA, USA: MIT Press, 1988.
- [3] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, 2016.
- [4] C. Katrakazas, M. Quddus, W. H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transp. Res. Part C Emerg. Technol.*, vol. 60, pp. 416–442, 2015. [Online]. Available: <http://dx.doi.org/10.1016/j.trc.2015.09.011>
- [5] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.
- [6] S. Vaskov, S. Kousik, H. Larson, F. Bu, J. Ward, S. Worrall, M. Johnson-Roberson, and R. Vasudevan, "Towards provably not-at-fault control of autonomous robots in arbitrary dynamic environments," *arXiv*, 2019.
- [7] Y. S. Shao, C. Chao, S. Kousik, and R. Vasudevan, "Reachability-based Trajectory Safeguard (RTS): A safe and fast reinforcement learning safety layer for continuous control," *arXiv*, 2020.
- [8] W. Swarting, J. Alonso-Mora, and D. Rus, "Planning and Decision-Making for Autonomous Vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. annurev-control-060117-105157, 2018. [Online]. Available: <http://www.annualreviews.org/doi/10.1146/annurev-control-060117-105157>
- [9] Z. Yang, Y. Zhang, J. Yu, J. Cai, and J. Luo, "End-to-end multi-modal multi-task vehicle control for self-driving cars with visual perceptions," in *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, 2018, pp. 2289–2294.
- [10] S. Li, M. Egorov, and M. J. Kochenderfer, "Optimizing collision avoidance in dense airspace using deep reinforcement learning," *13th USA/Europe Air Traffic Management Research and Development Seminar 2019*, no. 3, 2019.
- [11] L. Sun, J. Zhai, and W. Qin, "Crowd navigation in an unknown and
- [12] B. Mirchevska, C. Pek, M. Werling, M. Althoff, and J. Boedecker, "High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2156–2162.
- [13] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.
- [14] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-Based Collision Avoidance," *IEEE Transactions on Control Systems Technology*, pp. 1–12, 2020. [Online]. Available: <https://ieeexplore.ieee.org/document/9062306/>
- [15] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A Review of Motion Planning Techniques for Automated Vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1135–1145, 2016.
- [16] M. Walch, K. Lange, M. Baumann, and M. Weber, "Autonomous driving," vol. 21, no. 5, pp. 11–18, 2015.
- [17] C. Vallon and F. Borrelli, "Data-driven hierarchical predictive learning in unknown environments," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, 2020, pp. 104–109.
- [18] M. Kneissl, A. K. Madhusudhanan, A. Molin, H. Esen, and S. Hirche, "A Multi-Vehicle Control Framework With Application to Automated Valet Parking," *IEEE Trans. Intell. Transp. Syst.*, pp. 1–11, 2020.
- [19] C. Ericson, *Real-time collision detection*. CRC Press, 2004.
- [20] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2015-August, pp. 1094–1099, 2015. [Online]. Available: <https://ieeexplore.ieee.org/document/8619433/>
- [22] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.
- [23] X. Shen, I. Batkovic, V. Govindarajan, P. Falcone, T. Darrell, and F. Borrelli, "Parkpredict: Motion and intent prediction of vehicles in parking lots," *arXiv preprint arXiv:2004.10293*, 2020.
- [24] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari, "Forces nlp: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs," *International Journal of Control*, pp. 1–17, 2017.
- [25] A. Domahidi and J. Jerez, "Forces professional," Embotech AG, url=<https://embotech.com/FORCES-Pro>, 2014–2019.