IFAC Engineering Applications of Artificial Intelligence Journal

Visually-Guided Motion Planning for Autonomous Driving from Interactive Demonstrations

Rodrigo Pérez-Dattari[a,1,*], Bruno Brito[a,1], Oscar de Groot[a], Jens Kober[a], Javier Alonso-Mora[a]

[a] *Cognitive Robotics (CoR) department, Delft University of Technology, Mekelweg 2, 2628 CD, Delft, The Netherlands*

Abstract

The successful integration of autonomous robots in real-world environments strongly depends on their ability to reason from context and take socially acceptable actions. Current autonomous navigation systems mainly rely on geometric information and hard-coded rules to induce safe and socially compliant behaviors. Yet, in unstructured urban scenarios these approaches can become costly and suboptimal. In this paper, we introduce a motion planning framework consisting of two components: a data-driven policy that uses visual inputs and human feedback to generate socially compliant driving behaviors (encoded by high-level decision variables), and a local trajectory optimization method that executes these behaviors (ensuring safety). In particular, we employ Interactive Imitation Learning to jointly train the policy with the local planner, a Model Predictive Controller (MPC), which results in safe and human-like driving behaviors. Our approach is validated in realistic simulated urban scenarios. Qualitative results show the similarity of the learned behaviors with human driving. Furthermore, navigation performance is substantially improved in terms of safety, i.e., number of collisions, as compared to prior trajectory optimization frameworks, and in terms of data-efficiency as compared to prior learning-based frameworks, broadening the operational domain of MPC to more realistic autonomous driving scenarios.

Keywords: Interactive Imitation Learning, Model Predictive Control, Autonomous Driving, Deep Learning, Motion Planning, Human in the Loop

1. Introduction

Autonomous navigation in unstructured human environments (e.g., indoor and urban) poses a combination of problems, such as continuously changing conditions (e.g., sunny and cloudy), interaction/coordination with other agents (e.g., pedestrians, bicycles, human drivers and/or other automated vehicles), and ensuring the safety of people inside the vehicle and/or other agents in environment. Consequently, building robust robotic solutions in such environments remains challenging.

The safety of Autonomous Vehicles (AV) has been a main topic of interest in the research community. Optimization-based techniques for local trajectory planning, such as Model Predictive Control (MPC), have gained popularity, since they can provide safety guarantees through the enforcement of constraints, e.g., for collision avoidance. Nevertheless, the performance of optimization-based methods is limited in complex environments, since they typically rely on geometric information and hard-coded rules to control high-level variables (e.g., switching between behaviors, controlling velocity references, etc.), which are either costly or lead to suboptimal solutions. Hence, the interaction and coordination of AVs with other agents, in unstructured environments, remains challenging.

To address this limitation, recently, there has been a growing interest in approaches that combine the strengths of optimization-based methods with the ones of learning-based methods (Veličković and Blundell, 2021; Zamfirache et al., 2022). Learning techniques have shown to be a powerful tool for finding complex solutions directly from environment data, without requiring models.

In this work, we propose to learn human-like driving behaviors and encode them in a Model Predictive Contouring Control (MPCC) planner (Ferranti et al., 2019). Human-like driving behaviors are desired in AVs as they produce trust in other human drivers and facilitate coordination/interaction with other agents by acting predictably (Waytz et al., 2014). However, human data can be expensive to obtain, and modeling complex environments with changing conditions may require large amounts of data. Consequently, we propose to follow an Interactive Imitation Learning (IIL) approach (Amershi et al., 2014; Chernova and Veloso, 2009; Kelly et al., 2019), which—in contrast to non-interactive Imitation Learning approaches (e.g., Behavioral Cloning)— is data efficient. IIL employs online human feedback to transfer implicit knowledge from

[*] Corresponding author

Email addresses: `r.j.perezdattari@tudelft.nl` (Rodrigo Pérez-Dattari), `bruno.debrito@tudelft.nl` (Bruno Brito), `o.m.degroot@tudelft.nl` (Oscar de Groot), `j.kober@tudelft.nl` (Jens Kober), `j.alonsomora@tudelft.nl` (Javier Alonso-Mora)

[1] The authors contributed equally.

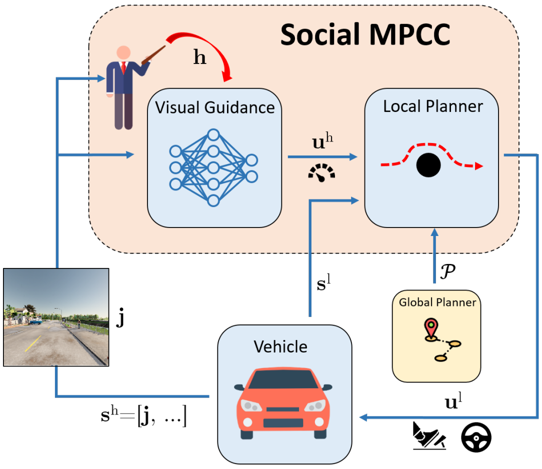


Figure 1: Proposed framework, Social MPCC. The Visual Guidance observes the environment from state $\mathbf{s}^{\mathrm{h}}$ and suggests the next velocity reference $\mathbf{u}^{\mathrm{h}}$ to the Local Planner. Then, the Local Planner, as a function of the state $\mathbf{s}^{\mathrm{l}}$, the velocity reference $\mathbf{u}^{\mathrm{h}}$ and the global path $\mathcal{P}$ computes a local trajectory and sends a control command $\mathbf{u}^{l}$ to the vehicle. Depending on the resulting vehicle behavior, the teacher may correct the Visual Guidance through the signal $\mathbf{h}$ to improve its behavior.

humans to robots.

To induce the learned behavior in the solutions of the MPCC, we propose to *learn to control high-level variables used in its objective function* such that the resulting optimization process yields solutions corresponding to the desired behavior. As a first step, we focus on controlling the forward velocity reference of the MPCC. This reference has a large impact on the vehicle's behavior and it is challenging to design otherwise, given that it depends on many variables (Kolekar et al., 2020). To closely match human behavior, we propose to learn to control this reference from (approximately) the same visual input that humans use, the first-person front-view of the vehicle (see Figure 1).

The main *contributions* of this work are:

- Combining the state-of-the-art from control and machine learning in a unified framework and problem formulation for motion planning.
- A framework to generate safe and socially-compliant trajectories in unstructured urban scenarios by learning human-like driving behavior efficiently.

We present simulation results in realistic driving scenarios using the CARLA simulator[2] (Dosovitskiy et al., 2017). The presented results show that our approach can data-efficiently learn velocity references from human feedback using images as input, enhancing the performance of local trajectory planners and generating safe and socially compliant behaviors. Furthermore, we compare our approach with optimization-based-only and learning-based-only baselines, demonstrating the strength of combining both methods. Finally, qualitative results show the ability of the method to learn human-like driving behaviors.

The remaining of this paper is organized as follows: Section 2 presents works related to decision-making algorithms for motion planning and IIL methods, Section 3 the problem formulation, Section 4 the proposed method and Section 5 the experimental results.

2. Related Work

In this section, we review work from the two fields that are brought together in this paper: 1) Motion Planning and 2) Interactive Imitation Learning.

2.1. Motion Planning

Classical autonomous navigation systems frequently employ a hierarchic planning architecture decomposing the navigation pipeline into a sequence of blocks performing different sub-tasks such as perception, high-level decision making and motion planning (Paden et al., 2016). These works can be divided into three main categories: rule-based, optimization-based and learning-based.

Rule-based methods aim to translate human-driving rules and behaviors into handcrafted functions. These methods have demonstrated good performance in some real structured scenarios such as precedence at an intersection (Baker and Dolan, 2008). Nevertheless, these methods are scenario specific and are prone to fail if the environment structure changes.

Optimization-based approaches typically model the decision-making problem as a Partially Observable Markov Decision Process (POMDP) as the other agents' intentions are not directly observable (Hubmann et al., 2017). To model interaction, Zhou et al. (2018) proposed a joint approach for behavior prediction and planning, combining online POMDP solvers (Cai et al., 2021) for behavior prediction and nonlinear receding horizon control for trajectory planning (Brito et al., 2019). Nevertheless, these approaches have scalability issues and assume structured navigation scenarios.

Learning-based methods can scale to cluttered and unstructured environments (Everett et al., 2021) allowing to incorporate high-dimensional data (e.g., RGB-D images, LiDAR point-clouds, etc.) into the decision-making policy (Fan et al.). For instance, Reinforcement Learning (RL) methods have been used to learn end-to-end control policies for autonomous racing (Schwarting et al., 2021) and indoor navigation (Kulhánek et al., 2020) by learning a policy optimizing for long-term rewards. To generate socially compliant behaviors, Chen et al. (2017b) proposed to introduce social rules into the learning framework by designing a reward function penalizing the agent when not respecting human navigation norms. Yet, these methods do not provide any robustness or safety guarantees (Huang et al., 2017).

Recently, works combining learning-based approaches for decision-making and optimization-based methods for

[2] Code available at: `https://github.com/rperezdattari/Social-MPCC`

motion planning have demonstrated to achieve superior performance by providing guidance on high-level decision variables needed to solve the optimization (Qiao et al., 2020; Song and Scaramuzza, 2020). Closely related to our work, Tolani et al. (2021) learn a subgoal policy from visual information using Model Predictive Control (MPC) as supervisor. In contrast, we propose to learn a visual decision-making policy from human feedback. Similarly, Huang et al. (2021) use adversarial learning to train an end-to-end decision-making module from human demonstrations. Nevertheless, it assumes a high-definition map to be available and considers a discrete set of decisions limiting the applicability of this approach only to well constrained driving scenarios.

2.2. Interactive Imitation Learning

Interactive Imitation Learning (IIL) is a branch of Imitation Learning (IL) whose objective is to develop algorithms that transfer a policy from a teacher to a learning agent (learner) through interactions between the teacher and the learner (Argall et al., 2009; Kelly et al., 2019; Pérez-Dattari et al., 2020). Some examples of feedback are demonstrations (Chernova and Veloso, 2009; Ross et al., 2011), relative corrections (Celemin and Ruiz-del Solar, 2019; Pérez-Dattari et al., 2020), preferences (Christiano et al., 2017) and evaluations (Knox and Stone, 2009). In autonomous driving, it is common to find methods that work with humans as teachers, and demonstrations as feedback (Bojarski et al., 2016; Kelly et al., 2019; Zhan et al., 2019; Prakash et al., 2020; Codevilla et al., 2019), given that 1) it is easy to find humans that know how to drive a vehicle, and 2) human-like driving is a desired feature in autonomous vehicles (Kolekar et al., 2020; Zhan et al., 2019). Therefore, building on top of this evidence, the IIL part of our work follows this same strategy.

Although, in the context of IIL, demonstrations are interpreted as feedback, they can be applied in non-interactive IL algorithms as well, i.e., Behavioral Cloning (BC) and Inverse Reinforcement Learning (IRL) (Osa et al., 2018). However, compared to non-interactive methods, IIL poses an ideal setting to learn from humans, as it reduces human effort by being data efficient. This is achieved by providing feedback online over trajectories induced by the learner's policy, which improves its behavior only in the relevant regions of the state space (i.e., the ones that are likely to be visited) (Ross et al., 2011; Spencer et al., 2022) Furthermore, IRL, not only suffers from inefficiency in terms of amount of demonstrations, but also suffers from inefficiency in terms of interactions with the environment (Ho and Ermon, 2016; Osa et al., 2018), which can be a limitation when a realistic simulator of the environment is not available.

The IIL method employed in this paper can be interpreted as a practical variation of DAgger (Ross et al., 2011), since DAgger is not designed to work interactively with humans. DAgger expects the teacher to provide demonstrations at every state visited by the learner, and the trajectories generated by the learner are a results of a mixed control setting that combines actions from the learner and from the teacher. However, humans are sensitive to timing and latency; therefore, providing good demonstrations over an agent that is partially controlled is counter intuitive and cognitively demanding (Laskey et al., 2017). Alternatively, the teacher can observe the learner's behavior and intervene whenever this behavior is not appropriate, taking control over the learner and use these actions as demonstrations, as proposed by Kelly et al. (2019); Waytowich et al. (2018); Spencer et al. (2022). The method used in this work belongs to this group of approaches. Note that this group can be extended (Mandlekar et al., 2020) and combined with other types of feedback (Chisari et al., 2022) and/or active learning (Ablett et al., 2020).

3. Preliminaries

Throughout this paper we use the term *ego-agent* to refer to the agent controlled by our method (e.g., autonomous vehicle or mobile robot) and *other agents* to refer to the non-controllable agents (e.g., human-driven vehicles, pedestrians, or robots) in the surrounding of the ego-agent. Moreover, vectors are denoted in bold lowercase letters, $\mathbf{x}$, and sets in calligraphic uppercase, $\mathcal{S}$. The Euclidean norm of $\mathbf{x}$ is denoted by $\|\mathbf{x}\|$ and $\|\mathbf{x}\|_Q = \mathbf{x}^T Q \mathbf{x}$ denotes the weighted squared norm.

3.1. Problem Formulation

Consider the navigation scenario where an ego-agent must navigate from an initial position $\mathbf{p}_0$ to a goal position $\mathbf{g}$. At the beginning of an episode, the ego-agent receives a global reference path $\mathcal{P}$ to follow from a path planner consisting of a sequence of M reference way points $\mathbf{p}_m^{\mathrm{ref}} = [x_m^{\mathrm{ref}}, y_m^{\mathrm{ref}}] \in \mathbb{R}^2$ with $m \in \mathcal{M} := \{1, \ldots, M\}$. Then, consider a hierarchical control structure with a high-level control policy $\pi_{\boldsymbol{\theta}}^{\mathrm{h}}$, defined as a parametrized function with parameters $\boldsymbol{\theta}$, and a predefined optimization-based low-level controller π^{l} that follows $\mathcal{P}$. The superscripts h and l are used to denote the variables related to the high-level and low-level controllers, respectively. For each time step k, the high-level policy receives the state $\mathbf{s}_k^{\mathrm{h}}$ and takes an action $\mathbf{u}_k^{\mathrm{h}} = \pi_{\boldsymbol{\theta}}^{\mathrm{h}}(\mathbf{s}_k^{\mathrm{h}})$. Subsequently, $\mathbf{u}_k^{\mathrm{h}}$ is provided, along with the state $\mathbf{s}_k^{\mathrm{l}}$ and the global reference path, to the low-level controller, which takes an action $\mathbf{u}_k^{\mathrm{l}} = \pi^{\mathrm{l}}(\mathbf{s}_k^{\mathrm{l}}, \mathbf{u}_k^{\mathrm{h}}; \mathcal{P})$. This action leads to the next state $\mathbf{s}_{k+1}^{\mathrm{l}} = f(\mathbf{s}_k^{\mathrm{l}}, \mathbf{u}_k^{\mathrm{l}})$, under the dynamic model $f(\mathbf{s}_k^{\mathrm{l}}, \mathbf{u}_k^{\mathrm{l}})$[3].

The policy that encompasses the combination of $\pi_{\boldsymbol{\theta}}^{\mathrm{h}}$ and π^{l} is denoted as $\pi_{\boldsymbol{\theta}}(\mathbf{s}_k; \mathcal{P})$, where $\mathbf{s}_k = [\mathbf{s}_k^{\mathrm{h}}, \mathbf{s}_k^{\mathrm{l}}]$. Note that the control output $\mathbf{u}_k = \pi_{\boldsymbol{\theta}}(\mathbf{s}_k; \mathcal{P})$ is the same as $\mathbf{u}_k^{\mathrm{l}}$, since $\mathbf{u}_k^{\mathrm{l}}$ is the output applied to the vehicle, while $\mathbf{u}_k^{\mathrm{h}}$ acts on the parameters of π^{l}.

[3] This is identical to the Vehicle Model used in the simulation defined in Section 4.3.1

Simultaneously, we consider that for each time step, the ego-agent receives the feedback signal $\mathbf{h}_k$, which provides information about a desired, expert behavior, π^{exp}. The goal is to employ $\mathbf{h}_k$ to find the parameters $\boldsymbol{\theta}$ such that $\pi_{\boldsymbol{\theta}}$ converges to π^{exp}. By doing so, $\pi^{\text{h}}_{\boldsymbol{\theta}}$ learns to guide π^{l} such that a desired behavior is achieved when following $\mathcal{P}$.

Let $p_{\pi_{\boldsymbol{\theta}}}(\tau)$ be a trajectory distribution induced by $\pi_{\boldsymbol{\theta}}$, and $p_{\pi^{\text{exp}}}(\tau)$ a trajectory distribution induced by π^{exp}, then, the problem can be formulated as the minimization of the (forward) Kullback–Leibler divergence between the trajectories induced by $\pi_{\boldsymbol{\theta}}$ and π^{exp} (Bishop, 2006):

$$\begin{aligned}
\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmin}} \quad & D_{\text{KL}}\left(p_{\pi_{\boldsymbol{\theta}}}(\tau) \| p_{\pi^{\text{exp}}}(\tau)\right) && \text{(1a)}\\
\text{s.t.} \quad & \mathbf{s}^{\text{l}}_{k+1} = f\left(\mathbf{s}^{\text{l}}_k, \mathbf{u}^{\text{l}}_k\right), && \text{(1b)}\\
& \mathbf{u}^{\text{l}}_k = \pi^{\text{l}}(\mathbf{s}^{\text{l}}_k, \mathbf{u}^{\text{h}}_k; \mathcal{P}), && \text{(1c)}\\
& \mathbf{u}^{\text{l}}_k \in \mathcal{U}^{\text{l}},\ \mathbf{u}^{\text{h}}_k \in \mathcal{U}^{\text{h}},\ \mathbf{s}^{\text{l}}_k \in \mathcal{S}^{\text{l}},\ \mathbf{s}^{\text{h}}_k \in \mathcal{S}^{\text{h}}, && \text{(1d)}\\
& \forall k \in \mathbb{R}^+
\end{aligned}$$

where (1b) are the kino-dynamic constraints and (1d) represents the state and control constraints where $\mathcal{S}^i$ and $\mathcal{U}^i$, $i \in \{\text{l}, \text{h}\}$, are the set of admissible states and control inputs (e.g., maximum agents' speed), respectively.

Note that, in this work, $\mathbf{h}_k$ is provided by a human; hence, $\boldsymbol{\theta}^*$ will depend on the human's judgment about the task.

4. METHOD

In this section, we introduce the proposed socially-aware Model Predictive Contouring Control (Social-MPCC) framework.

4.1. Overview

The proposed driving system can be divided into two parts: **Visual Guidance** and **Local Motion Planner**. The Local Motion Planner π^{l} follows a given set of way points and ensures to avoid obstacles. Simultaneously, the Visual Guidance system $\pi^{\text{h}}_{\boldsymbol{\theta}}$ uses images captured by the front camera view of the vehicle to command a desired forward velocity reference v^{ref} to the Local Motion Planner such that human-like driving behavior is generated. Given that the vehicle's steering commands are defined by the local planner only, it is not possible to exactly match a reference human-like behavior by means of controlling v^{ref} alone. Nevertheless, arguably, v^{ref}_k is expressive enough to accurately resemble human-like behavior in most of the situations; for instance, in the case of a vehicle in a city, the vehicle should reduce its velocity in the crossroads, stop at red lights, accelerate when overtaking other cars, etc.

4.2. Visual Guidance

For each time step k, the Visual Guidance system (VG), represented as the parametrized policy $\pi^{\text{h}}_{\boldsymbol{\theta}}$, translates human driving behavior and scene context into a forward velocity reference v^{ref}_k, which corresponds the high-level control output $\mathbf{u}^{\text{h}}_k := v^{\text{ref}}_k$. The state of this function $\mathbf{s}^{\text{h}}_k$ corresponds to the front camera view of the vehicle $\mathbf{j}_k$, and, eventually, other information such as the vehicle's current speed. The objective is to find $\mathbf{u}^{\text{h}*}_k$ $\forall k$ such that, given the Local Motion Planner, Eq. (1) is solved. As discussed in Section 1 and Section 2, given the challenges in modeling human behavior, Interactive Imitation Learning (IIL) arises as an appealing and effective approach to tackle this problem, since it allows to data-efficiently and robustly learn behaviors from humans.

4.2.1. Interactive Imitation Learning Formulation

In IIL, a human that acts as a teacher is involved in the learning process of an agent. Feedback signals $\mathbf{h}_k$ are generated by the human to modify the learner's policy towards a desired behavior in an online learning manner. The context of autonomous navigation provides a framework where humans, by driving a vehicle, are able to execute the actions which they consider to be the best for a given state. Consequently, it comes natural to use feedback in the form of demonstrations.

In this work, a Learning from Interventions scheme is employed, i.e., every time the human considers the agent to be executing an erroneous behavior, the teacher takes control over the agent's actions until it gets back into a region where the observed behavior is the desired one. The data gathered in these interventions is used as demonstrations for improving the agent's behavior following a supervised learning approach. The teacher's feedback is represented by two variables: 1) i, a Boolean that indicates if the human is intervening (if $i = 1$, the human intervenes; if $i = 0$, s/he does not), and 2) $\mathbf{u}^{\text{h}*}_k$, which corresponds to the teacher's optimal action for the high-level controller to take (i.e., these actions follow the expert policy π^{exp}). For $i = 1$, the feedback is defined as $\mathbf{h}_k = \mathbf{u}^{\text{h}*}_k$; for $i = 0$, it is not defined.

In practice, this works as follows: initially, the VG creates a velocity reference $\mathbf{u}^{\text{h}}_k$ that the Local Motion Planner tracks (along with a set of way points). The human only observes the behavior of the AV ($i = 0$), as long as s/he considers that it is adequate. Every time the planner generates undesired control commands, the human (indirectly) takes control over it ($i = 1$) by overwriting the output of the VG with the correct velocity reference $\mathbf{u}^{\text{h}*}_k$. The data from these interventions (i.e., trajectories containing every state-action pair $[\mathbf{s}^{\text{h}}_k, \mathbf{u}^{\text{h}*}_k]$) is collected, and employed to improve the agent's behavior.

Eq. (1) can be solved as an Imitation Learning problem $\forall \mathbf{h}_k$ when $i = 1$ (i.e., for every state-action pair collected from the interventions). If, every time the teacher intervenes, the demonstrated trajectories are stored in a dataset $\mathcal{D}$, Eq. (1) can be solved iteratively by sampling B trajectories with length K from $\mathcal{D}$ in every iteration and minimizing

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{B}\sum_{b=1}^{B}\sum_{k=0}^{K} \ln \pi^{\mathrm{h}}_{\boldsymbol{\theta}}(\mathbf{u}^{\mathrm{h}*}_{b,k}|\mathbf{s}^{h}_{b,k}) \tag{2}$$

through gradient descent (Abramson et al., 2020). Note that this formulation assumes that $\pi^{\mathrm{h}}_{\boldsymbol{\theta}}$ is a stochastic policy, but in this work $\pi^{\mathrm{h}}_{\boldsymbol{\theta}}$ is deterministic. However, if we assume that the optimized distribution is Gaussian with a fixed variance, the mean of this distribution can be equivalently obtained (and represented by the deterministic policy $\pi^{\mathrm{h}}_{\boldsymbol{\theta}}$) by minimizing the Mean Squared Error (MSE) (Osa et al., 2018; Mandlekar et al., 2020)

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{B}\sum_{b=1}^{B}\sum_{k=0}^{K} \left(\mathbf{u}^{\mathrm{h}*}_{b,k} - \pi^{\mathrm{h}}_{\boldsymbol{\theta}}(\mathbf{s}^{\mathrm{h}}_{b,k})\right)^2 . \tag{3}$$

This optimization process does not mention the constraints shown in Eq. (1) because they are implicitly captured in the demonstration data (given that it was collected following actions generated by the Local Motion Planner).

4.2.2. iDAgger

As depicted in Section 2, we use an IIL method based on demonstrations similar to the one described by Goecks et al. (2019) that solves Eq. (3); however, no name was provided by the authors to this method specifically, as they employed a subgroup of modules from a larger framework introduced by Waytowich et al. (2018). Hence, we will refer to it as iDAgger (for intervention DAgger). Note that similar ideas are employed in other works as well (Kelly et al., 2019; Spencer et al., 2020; Mandlekar et al., 2020).

iDAgger generates a dataset $\mathcal{D}$ online using feedback, in the form of interventions, provided by a human teacher. The state-action pairs generated in every intervention, i.e., $[\mathbf{s}^{\mathrm{h}}_k, \mathbf{u}^{\mathrm{h}*}_k]$, by the human are aggregated to $\mathcal{D}$. Every τ time steps, the learner updates its policy $\pi^{\mathrm{h}}_{\boldsymbol{\theta}}$ by sampling a subset of $\mathcal{D}$ and minimizing Eq. (3). Furthermore, $\pi^{\mathrm{h}}_{\boldsymbol{\theta}}$ can be initialized from an initial set of demonstrations collected offline. As shown by Spencer et al. (2022), the policies learned with this type of online learning algorithm are guaranteed to perform well (i.e., trajectory cost grows linearly in the task horizon and imitation error) under the intervention data distribution.

4.3. Local Motion Planner

We built upon the MPC formulation provided by the Model Predictive Contour Control (MPCC) (Ferranti et al., 2019) to generate control commands enabling the AV to follow a reference path provided by a global path planner (e.g., Rapidly-exploring Random Trees (RRT) (Berg et al., 2021)) and the forward velocity reference while satisfying dynamic and collision constraints when a feasible solution is found.

Algorithm 1 iDAgger

```
Require: $\mathcal{D}$ with initial demonstrations, pre-trained
    policy $\pi^{\mathrm{h}}_{\boldsymbol{\theta}}$ and policy update period $\tau$
for $k = 1, 2, \dots$ do
    observe $\mathbf{s}^{\mathrm{h}}_k$
    get intervention signal $i$
    if $i$ is True then
        get feedback $\mathbf{h}_k \leftarrow \mathbf{u}^{\mathrm{h}*}_k$
        aggregate $\{\mathbf{s}^{\mathrm{h}}_k, \mathbf{h}_k\}$ to $\mathcal{D}$
        $\mathbf{u}^{\mathrm{h}}_k \leftarrow \mathbf{h}_k$
    else
        $\mathbf{u}^{\mathrm{h}}_k \leftarrow \pi^{\mathrm{h}}_{\boldsymbol{\theta}}(\mathbf{s}^{\mathrm{h}}_k)$
    end if
    execute $\mathbf{u}^{\mathrm{h}}_k$
    update $\pi^{\mathrm{h}}_{\boldsymbol{\theta}}$ from $\mathcal{D}$ if mod$(k, \tau)$ is 0
end for
```

4.3.1. Vehicle Model

We use a kinematic bicycle model for the AV with state $\mathbf{s}^{\mathrm{l}} = [x, y, \phi, v]$, where x and y are the agent's Cartesian position coordinates, ϕ the heading angle and v the forward velocity fixed in a global inertial frame $\mathcal{W}$. The model is described as follows:

$$\begin{aligned}
\dot{x} &= v\cos(\phi + \beta) \\
\dot{y} &= v\sin(\phi + \beta) \\
\dot{\phi} &= \frac{v}{l_r}\sin(\beta) \\
\dot{v} &= u^a \\
\beta &= \arctan\left(\frac{l_r}{l_f + l_r}\tan\left(u^\delta\right)\right)
\end{aligned} \tag{4}$$

where β is the velocity angle and $\mathbf{u}^{\mathrm{l}}$ is the vehicle control input composed by the forward acceleration u^a and steering angle u^δ, $\mathbf{u}^{\mathrm{l}} = [u^a, u^\delta]$. l_r and l_f are the distances of the rear and front tires from the center of gravity of the vehicle, respectively, and are assumed to be identical.

4.3.2. Cost Function

The velocity reference v^{ref} generated by the VG allows controlling the AV driving behavior directly: high-velocity reference values lead to highly aggressive behavior while low-velocity reference values lead to cautious driving behavior. Hence, we design the local planner's cost function as follows:

$$\begin{aligned}
J(\mathbf{s}^{\mathrm{l}}_{\mathbf{k}}, \mathbf{u}^{\mathrm{l}}_k, \lambda_k) = {} & \left\|e^c_k(\mathbf{s}^{\mathrm{l}}_k, \lambda_k)\right\|_{q_c} + \left\|e^l_k(\mathbf{s}^{\mathrm{l}}_k, \lambda_k)\right\|_{q_l} \\
& + \left\|v^{\mathrm{ref}}_k - v_k\right\|_{q_v} + \left\|u^a_k\right\|_{q_u} + \left\|u^\delta_k\right\|_{q_\delta}
\end{aligned} \tag{5}$$

where $\mathcal{Q} = \{q_c, q_l, q_v, q_u, q_\delta\}$ denotes the set of cost weights and λ_k is the estimated progress along the reference path. First, we minimize the contour error (e^c_k) and lag error (e^l_k), to track the reference path closely. The contour error quantifies how much the ego vehicle deviates from the reference path laterally, whereas lag error is the deviation of the ego vehicle from the reference path longitudinally. Please refer to (Ferranti et al., 2019) for more details on path parameterization and tracking error. The third term, $\|v_k^{\mathrm{ref}} - v_k\|$, motivates the planner to follow the velocity reference provided by the Visual Guidance system closely. Finally, we add a quadratic penalty to the control commands, u_k^a and u_k^δ, to generate smooth trajectories.

4.3.3. Dynamic Obstacle Avoidance

First, we approximate the AV's occupied area, $\mathcal{A}^{\mathrm{ego}}$, as union of n_c circles, i.e., $\bar{A}^{\mathrm{ego}} \subseteq \bigcup_{c\in\{1,\dots,n_c\}} \mathcal{A}_c$, where $\mathcal{A}_c$ represents the c-th circle's area with radius r. For the other vehicles, the occupied area by the i-th vehicle, $\mathcal{A}^i$, is approximated by an ellipse of semi-major axis a_i, semi-minor axis b_i and orientation ϕ. Then, we define a set of non-linear constraints enforcing that each AV's circle c does not intersect with the i-th vehicle's elliptical:

$$c_k^{i,c}(\mathbf{s}_k^{\mathrm{l}}, \mathbf{s}_k^{\mathrm{l}_i}) = \begin{bmatrix}\Delta x_k^c \\ \Delta y_k^c\end{bmatrix}^{\mathrm{T}} R(\phi)^{\mathrm{T}} \begin{bmatrix}\frac{1}{\alpha^2} & 0 \\ 0 & \frac{1}{\beta^2}\end{bmatrix} R(\phi) \begin{bmatrix}\Delta x_k^c \\ \Delta y_k^c\end{bmatrix} > 1, \quad (6)$$

The parameters Δx_k^c and Δy_k^c represent x-y relative distances between the disc c and the ellipse i for planning step k. α and β are function of the AV's radius and the other vehicle's semi-major and semi-minor axis, respectively, and an enlarging coefficient ensuring collision avoidance, with $\alpha = a + r_{\mathrm{disc}} + \epsilon$ and $\beta = b + r_{\mathrm{disc}} + \epsilon$. We refer the reader to (Brito et al., 2019) for details on how ϵ is computed.

4.3.4. Road boundaries

To compute motion plans respecting the road boundaries, we employ constraints on the AV's lateral distance (i.e., contour error) with respect to the reference path ensuring that the vehicle stays within the road limits:

$$-w_{\mathrm{left}}^{\mathrm{road}} \leq e_k^{\mathrm{c}}(\mathbf{s}_k^{\mathrm{l}}) \leq w_{\mathrm{right}}^{\mathrm{road}} \quad (7)$$

where $w_{\mathrm{left}}^{\mathrm{road}}$ and $w_{\mathrm{right}}^{\mathrm{road}}$ are the left and right road limits, respectively.

4.3.5. MPCC Formulation

We formulate the motion planner as a Receding Horizon Trajectory Optimization problem (8) with planning horizon H conditioned on the following constraints:

$$\begin{aligned}
\mathbf{u}_{0:H-1}^{\mathrm{l}*} = \min_{u_{0:H-1}^{\mathrm{l}}} & \sum_{k=0}^{H-1} J(\mathbf{s}_k^{\mathrm{l}}, \mathbf{u}_k^{\mathrm{l}}, \lambda_k) + J(\mathbf{s}_H^{\mathrm{l}}, \lambda_H) && \text{(8a)} \\
\text{s.t.} \quad & \mathbf{s}_{k+1} = f(\mathbf{s}_k^{\mathrm{l}}, \mathbf{u}_k^{\mathrm{l}}), && \text{(8b)} \\
& \lambda_{k+1} = \lambda_k + v_k \Delta t && \text{(8c)} \\
& -w_{\mathrm{left}}^{\mathrm{road}} \leq e^c(\mathbf{s}_k^{\mathrm{l}}) \leq w_{\mathrm{right}}^{\mathrm{road}} && \text{(8d)} \\
& c_k^{i,c}(\mathbf{s}_k^{\mathrm{l}}, \mathbf{s}_k^{\mathrm{l}_i}) > 1 \ \forall c \in \{1,\dots,n_c\}, && \text{(8e)} \\
& \mathbf{u}_k^{\mathrm{l}} \in \mathcal{U}^{\mathrm{l}}, \quad \mathbf{s}_k^{\mathrm{l}} \in \mathcal{S}^{\mathrm{l}}, && \text{(8f)} \\
& \forall k \in \{0,\dots,H\}. && \text{(8g)}
\end{aligned}$$

where Δt is the discretization time and $\mathbf{u}_{0:H-1}^{\mathrm{l}*}$ the locally optimal control sequence for H time-steps. The solver employed attempts to find a solution for the MPCC problem for a fixed number of iterations. If a feasible solution is found, we apply only the first control input for each step and recompute a new solution in the next iteration considering new observed information in a receding horizon fashion. Otherwise we employ a safety control command $\mathbf{u}_{\mathrm{safety}}^{\mathrm{l}}$.

Algorithm 2 Social-MPCC

```
 1: Require: global planner, iDAgger (Algorithm 1), MPCC and number of episodes $n_{\mathrm{episodes}}$
 2: run Visual Guidance training with iDAgger in separate thread
 3: while $i_{\mathrm{episode}} < n_{\mathrm{episodes}}$ do
 4:     get reference path $\mathcal{P}$ from a global planner
 5:     for $k = 1, 2, \dots$ do
 6:         get states $\mathbf{s}_k^{\mathrm{h}}$, $\mathbf{s}_k^{\mathrm{l}}$ from environment
 7:         send $\mathbf{s}_k^{\mathrm{h}}$ to iDAgger (Algorithm 1, line 3)
 8:         receive $\mathbf{u}_k^{\mathrm{h}}$ from iDAgger (Algorithm 1, line 12)
 9:         set $v_k^{\mathrm{ref}} \leftarrow \mathbf{u}_k^{\mathrm{h}}$
10:         compute $\mathbf{u}_k^{\mathrm{l}} \leftarrow \pi^{\mathrm{l}} = \mathrm{MPCC}(v_k^{\mathrm{ref}}, \mathbf{s}_k^{\mathrm{l}}; \mathcal{P})$ (Eq. 8)
11:         execute $\mathbf{u}_k^{\mathrm{l}}$ in vehicle
12:         compute done ← collision/deadlock detected or teacher request
13:         if done then
14:             increment $i_{\mathrm{episode}}$
15:             break
16:         end if
17:     end for
18: end while
```

4.4. Social-MPCC

Overall, the Social-MPCC framework utilizes the Visual Guidance policy to provide a velocity reference that controls the vehicle's behavior through the cost function that is optimized by the Local Motion Planner. Imitation Learning is used to optimize the VG's parameters to model human behavior.

Algorithm 2 presents the overall framework. First, iDAgger (Algorithm 1) is initialized to start the training of the Visual Guidance (line 2). Then, at the beginning of each episode, the reference path $\mathcal{P}$ to be followed by the MPCC is obtained from a global planner (line 4). Afterwards, for every time step of each episode, the velocity reference $v_k^{\mathrm{ref}} = \mathbf{u}_k^{\mathrm{h}}$ is received from iDAgger (lines 8-9) and fed to the MPCC to compute the control command $\mathbf{u}_k^{\mathrm{l}}$ (line 10). Finally, $\mathbf{u}_k^{\mathrm{l}}$ controls the AV (line 11). Each episode ends if a collision or a deadlock is detected. Moreover, the human teacher can also request the end of the episode (lines 12-15).

5. RESULTS

This section presents simulation results in a realistic urban scenario populated with pedestrians and other vehicles (Fig. 3). First, we quantify the performance throughout the training procedure (Section 5.2). Then, we show a qualitative evaluation of the method (Section 5.3). Finally, we present performance results (Section 5.4) of the proposed method against baselines.

5.1. *Experimental setup*

Simulation results were carried out on an Intel Core i9, 32GB of RAM CPU @ 2.40GHz. The non-linear and non-convex MPCC problem presented in Section 4.3 was solved using the ForcesPro (Zanelli et al., 2020) solver. The Visual Guidance was modeled with a Deep Neural Network (DNN) implemented and optimized in TensorFlow 2 (Abadi et al., 2015). We used the open-source Carla simulator (Dosovitskiy et al., 2017) to create the simulation environment where the *Traffic Manager* module was employed to simulate other vehicles and the *AI controller* to control the pedestrians. The complete framework was interfaced using the Robot Operating System (ROS) (Quigley et al., 2009).

5.1.1. *Visual Guidance: Deep Neural Network Architecture*

The DNN architecture employed to represent the VG is defined by the mapping $\pi^{\mathrm{h}}_{\boldsymbol{\theta}} : \mathbf{s}^{\mathrm{h}}_k \mapsto v^{\mathrm{ref}}_k$. The VG has to 1) be able to process images $\mathbf{j}_k$, 2) deal with partial observability due to the absence of temporal information in $\mathbf{j}_k$. Moreover, to further improve the input state of the VG, the vehicle's speed v_k can also be provided to the network. Convolutional layers were employed to process $\mathbf{j}_k$ and recurrent layers to deal with the mentioned partial observability (Goodfellow et al., 2016). Hence, the high-level state was defined as $\mathbf{s}^{\mathrm{h}}_k = [\mathbf{j}_k, \mathbf{s}^{\mathrm{rec}}_k, v_k]$, where $\mathbf{s}^{\mathrm{rec}}_k$ corresponds to the hidden state of the recurrent layers.

To increase the generalization properties and data-efficiency of the network, two techniques were employed: 1) semantic segmentation (Minaee et al., 2021), and 2) Transfer Learning (TL) (Tan et al., 2018). The input image $\mathbf{j}_k$ was semantically segmented using a Carla module; however, in practice, DNN models such as Segnet (Badrinarayanan et al., 2017) or DeepLab (Chen et al., 2017a) can be employed. For TL, we employed a VGG (Simonyan and Zisserman, 2014) model pretrained on ImageNet (Deng et al., 2009). The last layer of the VGG was removed and replaced with recurrent and fully connected layers with trainable parameters to be optimized with Eq. (3). Hence, the VGG was used as a state representation/feature extraction machine and its weights were not modified during the optimization of $\pi^{\mathrm{h}}_{\boldsymbol{\theta}}$. LSTM (Hochreiter and Schmidhuber, 1997) layers were employed as the recurrent layers of the network. Finally, to optimize Eq. (3) with a recurrent DNN, we employed the *bootstrapped random updates* method (Hausknecht and Stone, 2015). Figure 2 shows the complete VG model.

Table 1 presents the hyperparameters values used for the local planner, training algorithm and simulation environment.

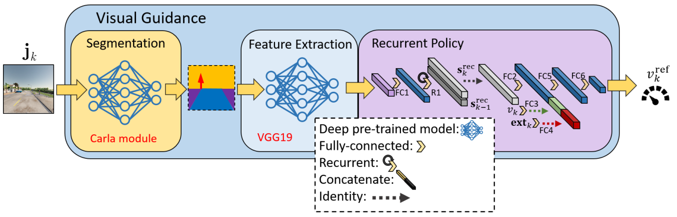


Figure 2: Visual Guidance architecture (Section 5.1.1). The *segmentation* module receives the image $\mathbf{j}_k$ and provides a segmented image to the *feature extraction* module. The *recurrent policy* takes these features as an input and generates the velocity reference v^{ref}_k. In this work, a Carla module was used for the segmentation module and a pre-trained VGG19 network was used for feature extraction. The recurrent policy consists of fully-connected layers (FCX), and one recurrent layer (R1). FC1, FC2, FC3, FC4 and FC5 use *Leaky ReLU* (Maas et al., 2013) as activation function. FC1 has 150 neurons and the other layers have 1000 neurons. FC6 has a linear activation and one neuron, as it is the output layer. The hidden state size of R1 is 150. The variable ext_k corresponds to extensions to the input of the network, such as traffic light state and/or information about where to go when learning in an end-to-end manner.

5.2. *Training procedure*

Fig. 3 shows the training environment. At the beginning of each episode, N_{cars} cars and $N_{\mathrm{pedestrians}}$ pedestrians are spawned in random locations. The AV receives a sequence of way-points towards a random goal position provided by the Carla *Route Planner*. An episode ends if the AV collides, if it reaches the goal position successfully, if a deadlock occurs, or if a teacher request is received.

Fig. 4 presents the VG's learning performance. The first plot shows the amount of times the teacher corrected the policy's actions, and the second plot the moving average of the mean squared error between the teacher and the policy's actions. The training procedure incorporates two phases: collection of an initial set of demonstrations used to train an initial policy, and the interactive learning process (as shown in Algorithm 1). Given that during the first $N_{\mathrm{Supervised}}$ steps the teacher provides feedback continuously, the number of demonstrations grows linearly, as depicted in the upper plot of Fig. 4 from $t = 0$ s to $t \approx 1500$ s. Afterwards, feedback is only provided when the policy acts erroneously, which will depend on the episode's complexity and the novelty it provides. After $t \approx 4500$ s, the number of demonstrations remains constant, showing that the policy is performing well and the teacher does not need to provide more feedback.

The bottom plot shows that the moving average of the root mean squared error between the VG's action and the provided supervised action reduces over time, stabilizing at around 2 m/s (note that $0 \text{ m/s} \leq \mathbf{u}^{\mathrm{h}}_k \leq 8 \text{ m/s}$). Although this error may seem large, this result is expected, since humans are not always consistent about the feedback they provide (Chernova and Veloso, 2008; Valletta et al., 2021), and data with irreducible error is collected. Nevertheless, this is not considered to be an issue in this experiment, as, when the mean squared error is minimized, the modes present in the data are averaged, which was not observed as

Table 1: Key hyperparameters used for local planner, learning algorithm and simulated environment.

	Hyperparameter	Value
Simulated environment	N_{Cars}	100
	$N_{\text{Pedestrians}}$	200
	$N_{\text{Supervised}}$	5
	Camera FoV	90.0°
Visual Guidance	Interactive training time	2 hours
	Image resolution	64 × 64 pixels
	Feature extractor	VGG
	Recurrent layer	LSTM
	Optimizer	Adam
	Learning rate	$1e-5$
	Batch size	100
	Training iterations per episode	1000
Local planner	$Q = \{q_c, q_l, q_v, q_u, q_\delta\}$	$\{0.1, 0.2, 1.0, 1.2, 0.1\}$
	Number solver iterations	500
	$\mathbf{u}_{\text{safety}}$	$[-2.0, 0.0]$
	Solver method	Primal-Dual Interior-Point Method

being detrimental. Inconsistencies only arose in situations where their average would not jeopardize the safety of the learned behavior (e.g., cruise speed in long roads or the response time to start accelerating after a green light), while the general rules of driving (e.g., stopping at red lights or if there are pedestrians crossing the street) were always respected.

5.3. Qualitative Results

This section analyzes the AV behavior using our method for two driving scenarios. In the first scenario, depicted in Fig. 5a, the AV approaches a crossing area and has to perform a left-turn maneuver. Between t_1 and t_2, the VG continuously reduces the velocity reference as the AV approaches the crossing area yielding to the vehicle coming from the right. Then, the velocity reference initially increases, motivating the AV to cross the road, between $t = 215$ s and $t = 217$ s, and keeps a continuous reference while turning left, between $t = 217$ s and $t = 222$ s. Once the vehicle finishes turning left, approximately at $t = 222$ s, the velocity reference is increased again.

In the second scenario, depicted in Fig. 5b, a pedestrian crosses the road in front of the AV. The VG reduces the velocity reference to let the pedestrian cross, between $t \approx 595$ s and $t \approx 598$ s. Once the pedestrian finishes crossing, the reference is increased. Afterwards, to safely perform a right turn maneuver, the velocity reference is reduced.

More scenarios can be found in the attached video[4], where it is possible to appreciate that the exhibited behaviors resemble human driving.

5.4. Quantitative Results

The objective of this section is to study, quantitatively, the effect on the performance of a trajectory planner when optimization-based and learning-based methods are combined. To achieve this, we study three types of algorithms: 1) optimization-based only (MPCC), 2) optimization-based and learning-based combined (Social-MPCC with and without traffic information) and 3) completely data-driven (End-to-end learning with traffic information), which are described below:

- **MPCC:** Local Motion Planner with constant velocity reference.
- **Social-MPCC:** the proposed Social-MPCC framework.
- **Social-MPCC with traffic information:** Social-MPCC with traffic lights' information in its state.
- **End-to-end learning with traffic information:** same as before, but the AV's control $\mathbf{u}^l = [u^a, u^\delta]$ is learned using iDAgger alone.

To test the flexibility of the proposed framework, two variations of Social-MPCC are presented, one that is general to any autonomous driving scenario and one that is specific to driving in a city. In the general case, the structure of the VG is as explained in Section 5.1.1; in the specific case, the input is extended with the traffic lights' state (see Fig. 2). Note that, strictly speaking, the traffic lights' status is also fed to the neural network in the general case, as it can be perceived from few pixels in $\mathbf{j}_k$ when the AV approaches a traffic light. Nevertheless, to obtain real-time performance, it is necessary to limit the input's resolution; hence, the resolution of the images was not high enough to effectively use the traffic lights' information from them.

It is to be expected that Social-MPCC will perform better when traffic information is included into the system than when it is not. Therefore, to obtain a fair comparison

[4] Available at: `https://youtu.be/Ph7v25mEg7c`

(a) Top view city.

(b) Ego vehicle's back view.

Figure 3: Carla simulation environment.

against the end-to-end policy, the traffic lights' state is also employed in this case. Moreover, when the complete behavior is learned from data, it is also necessary to provide information to the network about where the vehicle should go, as in the other methods this information is given to the MPCC through $\mathcal{P}$. To achieve this, the network was provided with $\sin(\gamma)$ (in the same way as the traffic lights' state, see Fig. 2), where γ is the angle between the center of the vehicle and the next way-point located at distance of ~ 15 m from it.

Finally, to test the data efficiency of Social-MPCC, **only 2 hours** were employed for the complete learning process of the experiments, as opposed to other methods in the literature that can use 100–200× more human time (e.g., ~ 300 hours (Vitelli et al., 2021)). Table 2 compares the performance of the introduced methods in terms of number of collisions per traveled distance and percentage of deadlocks. In terms of computation performance, the VG (i.e., DNN) has an average computation time of 5.1 ± 0.9 ms, while the MPCC optimization problem (Eq. (8)) takes on average 3.0 ± 1.35 ms.

5.4.1. Discussion

From Table 2 it can be observed that the performance of MPCC is drastically improved by Social-MPCC with only 2 hours of training. With the general Social-MPCC framework, it was possible to reduce the amount of collisions per kilometer by 3.66 times. Furthermore, in the case in which the traffic lights' information was provided to the VG, this value incremented to 7.03 and the percentage of deadlocks was reduced to 13 %.

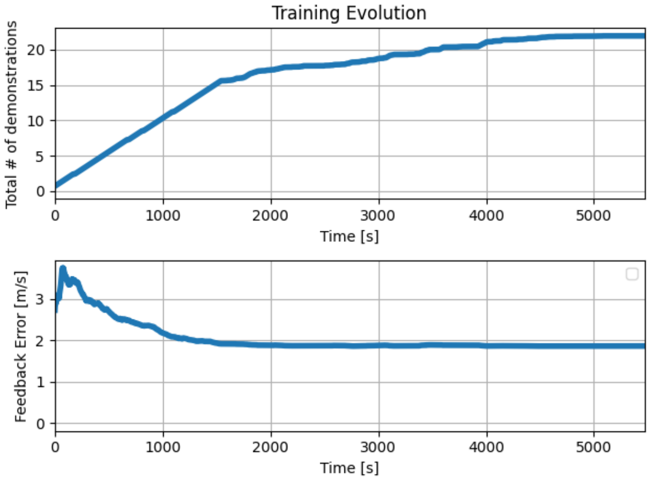


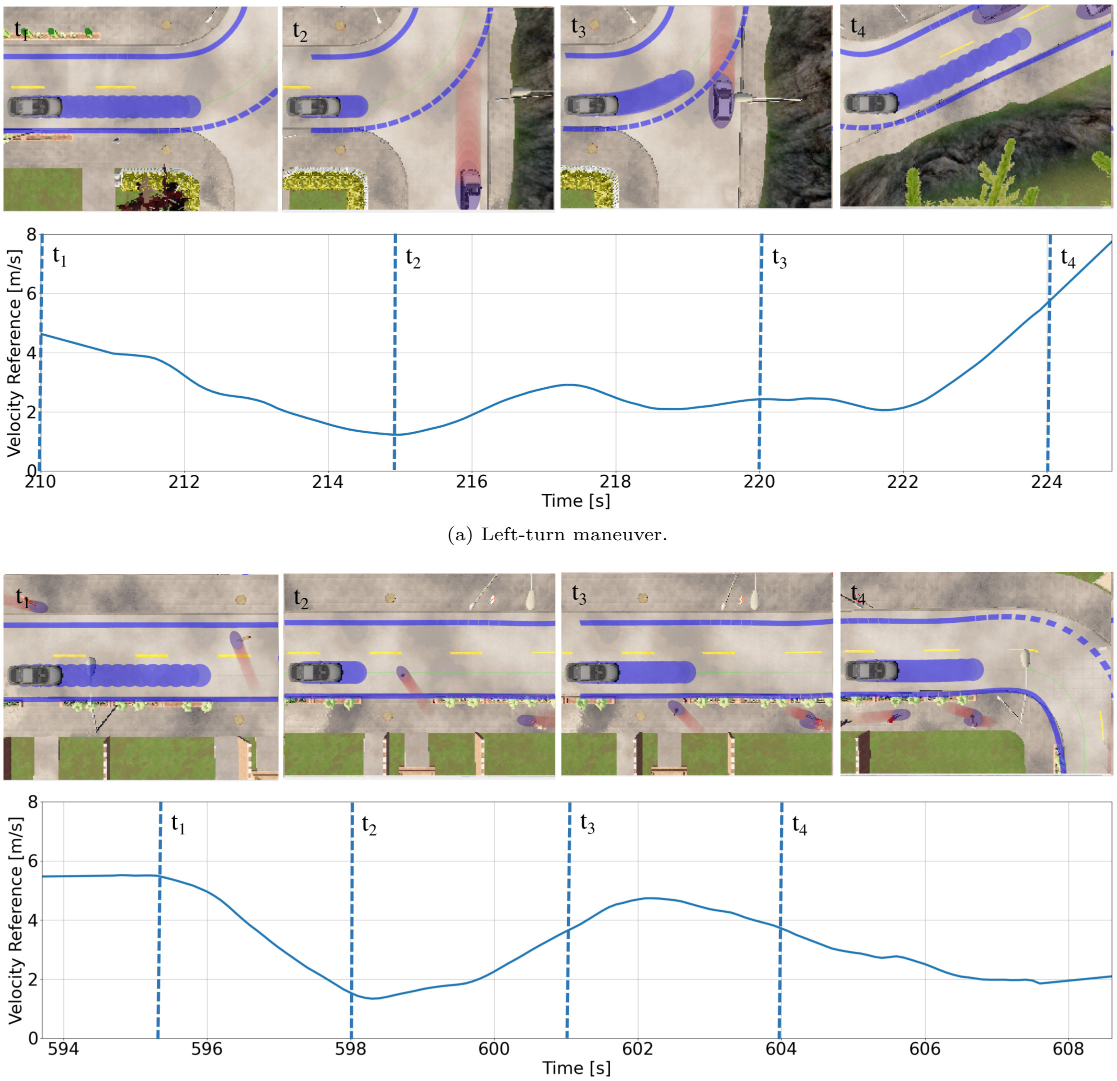
Figure 4: Accumulated feedback and policy error evolution during training. The top plot shows the number of times the teacher had to correct the policy's action and the bottom the average policy's action error.

Deadlocks occurred when the vehicle was stationary for an extended period of time (600 time steps in this experiment). Hence, when no feasible solutions were found by the MPCC, the activation of $\mathbf{u}^{\mathrm{l}}_{\mathrm{safety}}$ could have led to deadlocks. Interestingly, the VG also generated deadlocks. It was observed that the DNN could get stuck by constantly providing zero forward velocity reference to the local planner. Nevertheless, the combination of MPCC and VG did not increase the number of deadlocks when combined in Social-MPCC; furthermore, the number of deadlocks was reduced when the traffic lights' information was employed in the system. This occurred because an adaptive forward velocity reference can help the MPCC find solutions in cases where it would otherwise get stuck.

Finally, it was observed that the end-to-end learner achieved an acceptable performance; however, Social-MPCC showed to be superior after two hours of training. Increasing the action space of the VG to also include a steering angle reference makes the learning problem considerably harder. This is can be observed with both the number of collisions per kilometer (5.24× more) and the amount of deadlocks (1.38× more) that the end-to-end learner obtained.

5.4.2. Analysis Social-MPCC

Although IIL methods are very data efficient, it is still not possible to learn a flawless behavior in 2 hours; moreover, assumptions in the MPCC's formulation may cause it to perform suboptimally. Hence, there are situations in which our method fails. We have visually inspected the



(a) Left-turn maneuver.

(b) Pedestrian crossing the road.

Figure 5: The blue circles depict the MPCC's velocity reference provided by the VG and the blue lines the road constraints. The red circles depict the predicted constant velocity trajectory for the other vehicles or pedestrians.

Table 2: Statistic results for 100 episodes of Social-MPCC compared to baselines.

	No. Collisions per km.	% of deadlocks
MPCC (Ferranti et al., 2019)	2.60	17
Social-MPCC	0.71	17
Social-MPCC with traffic lights	**0.37**	**13**
End-to-end with traffic lights	1.94	18

training episodes and identified the main factors leading to failure (i.e., collisions). Table 3 presents the five main failure factors and their frequency considering a total of 100 episodes.

The categories in Table 3 are presented bellow:

- **Unusual situations:** occasionally, the AV may get into situations that are not common, such as interacting with oddly shape vehicles or with multiple vehicles that got stuck and not moving, that are unlikely to by encountered during training. Therefore, the VG may not be trained in similar circumstances and consequently generate incorrect behaviors.
- **Outside the camera Field of View (FoV):** due to the limited FoV of the first person view camera used by the VG, our system is not able to obtain all of the relevant visual information for driving in every situation. Hence, there are cases in which obstacles are not perceived on time, leaving the system too

Table 3: Analysis of failure episodes: number of episodes per factor leading to failure. We consider a total of 100 episodes.

Factor	№ episodes
Unusual situations	5
Outside the camera FoV	1
Wrong predictions	1
Small obstacles	8
Other agents contempt driving rules	1
Total	16/100

little time to react safely.

- **Wrong predictions:** the MPCC framework works under the assumption that other vehicles and pedestrians have constant velocities. This assumption does not hold in every situation, which may cause failures.
- **Small obstacles:** Small obstacles, such as children and bicycles, are not always easily perceived by the VG. Furthermore, they are not frequently encountered by the AV, which makes it more challenging to properly learn about these cases during training. Therefore, our system was not fully robust in avoiding collisions with small obstacles.
- **Other agents contempt driving rules:** in some cases, other agents, such as vehicles or pedestrians, do not respect the driving rules. Other vehicles may ignore red lights or pedestrians may cross the street in places where they are not allowed to, inducing collisions with our system.

Small obstacles and unusual situations were the two most frequent types of failures. Both cases occurred, in large part, due to the limited number of episodes during which the policy was trained. More training time or data augmentation techniques would largely help to decrease the frequency of these failures.

The rest of the failure cases did not affect the performance of the AV to a great extent, as they happened once each. However, the proposed framework could be extended to reduce these types of collisions. The failure episodes due to limited FoV can be solved by, for instance, by incorporating 360° visual information, allowing the policy to reason about the surrounding environment completely. Secondly, failures due to wrong predictions can be solved with a high-fidelity prediction model (Brito et al., 2020) reasoning about interaction and environment constraints. Lastly, in the cases where other agents contempt driving rules, the local planner's safety bounds can be increased; moreover, more training time can help make the VG be more robust.

6. CONCLUSION

In this paper we presented a framework, Social-MPCC, that combines an optimization-based control method (MPCC) with a learning-based method (iDAgger) for learning and executing safe, human-like, driving behaviors. Learning human-like driving behaviors is a desired feature for AVs, as they produce trust in other human agents and facilitate collision avoidance by acting predictably. To achieve this, the forward velocity reference of a local trajectory planner is modified in real time by a Visual Guidance system that learns, from humans, to control this variable using first-person view images of a vehicle. The learning method follows an Interactive Imitation Learning training procedure that enables obtaining well-performing policies in only two hours of human training time, as opposed to other methods in the literature that require 100–200× more human time.

The method was experimentally validated in a realistic simulator. Qualitative results show the capacity of the method to successfully encode human-like driving behaviors in the MPCC. Quantitative results compare the performance of Social-MPCC against baselines that are optimization-based (i.e., MPCC) or learning-based only (i.e., end-to-end iDAgger). Social-MPCC substantially improved the performance of MPCC, both in terms of number of collisions and deadlocks. Furthermore, after two hours of interactive training, the proposed method showed to be superior to the end-to-end learning method. Finally, Social-MPCC achieved real-time performance, which allows it to be implemented on a real platform.

Future works can extend Social-MPCC to control a larger family of high-level control variables of the MPCC with the Visual Guidance. For instance, way points could be locally modified to enforce specific behaviors. Furthermore, modifying the weights in the MPCC's cost function could also be employed for this purpose. Finally, the proposed framework could also be extended with other Interactive Learning techniques: for example, corrective advice could be used to teach behaviors that may be challenging to demonstrate (Pérez-Dattari et al., 2020).

Acknowledgments

This work was supported by the Amsterdam Institute for Advanced Metropolitan Solutions and the Netherlands Organisation for Scientific Research (NWO) domain Applied Sciences with projects Veni 15916, FlexCRAFT P17-01 and NWA.1292.19.298.

References

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

Ablett, T., Marić, F., Kelly, J., 2020. Fighting failures with fire: Failure identification to reduce expert burden in intervention-based learning. arXiv preprint arXiv:2007.00245 .

Abramson, J., Ahuja, A., Barr, I., Brussee, A., Carnevale, F., Cassin, M., Chhaparia, R., Clark, S., Damoc, B., Dudzik, A., et al., 2020. Imitating interactive intelligence. arXiv preprint arXiv:2012.05672 .

Amershi, S., Cakmak, M., Knox, W.B., Kulesza, T., 2014. Power to the people: The role of humans in interactive machine learning. Ai Magazine 35, 105–120.

Argall, B.D., Chernova, S., Veloso, M., Browning, B., 2009. A survey of robot learning from demonstration. Robotics and autonomous systems 57, 469–483.

Badrinarayanan, V., Kendall, A., Cipolla, R., 2017. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE transactions on pattern analysis and machine intelligence 39, 2481–2495.

Baker, C.R., Dolan, J.M., 2008. Traffic interaction in the urban challenge: Putting boss on its best behavior. 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS , 1752–1758doi:10.1109/IROS.2008.4651211.

Berg, B., Brito, B., Alonso-Mora, J., Alirezaei, M., 2021. Curvature Aware Motion Planning with Closed-Loop Rapidly-exploring Random Trees, in: 2021 IEEE Intelligent Vehicles Symposium (IV).

Bishop, C.M., 2006. Pattern recognition. Machine learning 128.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., et al., 2016. End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316 .

Brito, B., Floor, B., Ferranti, L., Alonso-Mora, J., 2019. Model predictive contouring control for collision avoidance in unstructured dynamic environments. IEEE Robotics and Automation Letters 4, 4459–4466.

Brito, B., Zhu, H., Pan, W., Alonso-Mora, J., 2020. Social-vrnn: One-shot multi-modal trajectory prediction for interacting pedestrians. Conference on Robot Learning .

Cai, P., Luo, Y., Hsu, D., Lee, W.S., 2021. Hyp-despot: A hybrid parallel algorithm for online planning under uncertainty. The International Journal of Robotics Research 40, 558–573.

Celemin, C., Ruiz-del Solar, J., 2019. An interactive framework for learning continuous actions policies based on corrective feedback. Journal of Intelligent & Robotic Systems 95, 77–97.

Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L., 2017a. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence 40, 834–848.

Chen, Y.F., Everett, M., Liu, M., How, J.P., 2017b. Socially aware motion planning with deep reinforcement learning, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1343–1350. doi:10.1109/IROS.2017.8202312.

Chernova, S., Veloso, M., 2008. Learning equivalent action choices from demonstration, in: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE. pp. 1216–1221.

Chernova, S., Veloso, M., 2009. Interactive policy learning through confidence-based autonomy. Journal of Artificial Intelligence Research 34, 1–25.

Chisari, E., Welschehold, T., Boedecker, J., Burgard, W., Valada, A., 2022. Correct me if i am wrong: Interactive learning for robotic manipulation. IEEE Robotics and Automation Letters .

Christiano, P., Leike, J., Brown, T.B., Martic, M., Legg, S., Amodei, D., 2017. Deep reinforcement learning from human preferences. arXiv preprint arXiv:1706.03741 .

Codevilla, F., Santana, E., López, A.M., Gaidon, A., 2019. Exploring the limitations of behavior cloning for autonomous driving, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 9329–9338.

Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L., 2009. ImageNet: A Large-Scale Hierarchical Image Database, in: CVPR09.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V., 2017. CARLA: An open urban driving simulator, in: Proceedings of the 1st Annual Conference on Robot Learning, pp. 1–16.

Everett, M., Chen, Y.F., How, J.P., 2021. Collision avoidance in pedestrian-rich environments with deep reinforcement learning. IEEE Access 9, 10357–10377. doi:10.1109/ACCESS.2021.3050338.

Fan, T., Long, P., Liu, W., Pan, J., . Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. The International Journal of Robotics Research .

Ferranti, L., Brito, B., Pool, E., Zheng, Y., Ensing, R.M., Happee, R., Shyrokau, B., Kooij, J., Alonso-Mora, J., Gavrila, D.M., 2019. SafeVRU: A Research Platform for the Interaction of Self-Driving Vehicles with Vulnerable Road Users. 2019 IEEE Intelligent Vehicles Symposium .

Goecks, V.G., Gremillion, G.M., Lawhern, V.J., Valasek, J., Waytowich, N.R., 2019. Efficiently combining human demonstrations and interventions for safe training of autonomous systems in real-time, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 2462–2470.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep learning. MIT press.

Hausknecht, M., Stone, P., 2015. Deep recurrent q-learning for partially observable mdps, in: 2015 aaai fall symposium series.

Ho, J., Ermon, S., 2016. Generative adversarial imitation learning. Advances in neural information processing systems 29, 4565–4573.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural computation 9, 1735–1780.

Huang, J., Xie, S., Sun, J., Ma, Q., Liu, C., Shi, J., Lin, D., Zhou, B., 2021. Learning a decision module by imitating driver's control behaviors. arXiv:1912.00191.

Huang, S., Papernot, N., Goodfellow, I., Duan, Y., Abbeel, P., 2017. Adversarial attacks on neural network policies. arXiv preprint arXiv:1702.02284 .

Hubmann, C., Becker, M., Althoff, D., Lenz, D., Stiller, C., 2017. Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles, in: 2017 IEEE Intelligent Vehicles Symposium (IV), pp. 1671–1678. doi:10.1109/IVS.2017.7995949.

Kelly, M., Sidrane, C., Driggs-Campbell, K., Kochenderfer, M.J., 2019. Hg-dagger: Interactive imitation learning with human experts, in: 2019 International Conference on Robotics and Automation (ICRA), IEEE. pp. 8077–8083.

Knox, W.B., Stone, P., 2009. Interactively shaping agents via human reinforcement: The tamer framework, in: Proceedings of the fifth international conference on Knowledge capture, pp. 9–16.

Kolekar, S., de Winter, J., Abbink, D., 2020. Human-like driving behaviour emerges from a risk-based driver model. Nature communications 11, 1–13.

Kulhánek, J., Derner, E., Babuška, R., 2020. Visual navigation in real-world indoor environments using end-to-end deep reinforcement learning. arXiv:2010.10903.

Laskey, M., Chuck, C., Lee, J., Mahler, J., Krishnan, S., Jamieson, K., Dragan, A., Goldberg, K., 2017. Comparing human-centric and robot-centric sampling for robot deep learning from demonstrations, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE. pp. 358–365.

Maas, A.L., Hannun, A.Y., Ng, A.Y., et al., 2013. Rectifier nonlinearities improve neural network acoustic models, in: Proc. icml, Citeseer. p. 3.

Mandlekar, A., Xu, D., Martín-Martín, R., Zhu, Y., Fei-Fei, L., Savarese, S., 2020. Human-in-the-loop imitation learning using remote teleoperation. arXiv preprint arXiv:2012.06733 .

Minaee, S., Boykov, Y.Y., Porikli, F., Plaza, A.J., Kehtarnavaz, N., Terzopoulos, D., 2021. Image segmentation using deep learning: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence .

Osa, T., Pajarinen, J., Neumann, G., Bagnell, J.A., Abbeel, P., Peters, J., 2018. An algorithmic perspective on imitation learning. arXiv preprint arXiv:1811.06711 .

Paden, B., Čáp, M., Yong, S.Z., Yershov, D., Frazzoli, E., 2016. A survey of motion planning and control techniques for self-driving urban vehicles. IEEE Transactions on intelligent vehicles 1, 33–55.

Pérez-Dattari, R., Celemin, C., Franzese, G., Ruiz-del Solar, J., Kober, J., 2020. Interactive learning of temporal features for control: Shaping policies and state representations from human feedback. IEEE Robotics & Automation Magazine 27, 46–54.

Prakash, A., Behl, A., Ohn-Bar, E., Chitta, K., Geiger, A., 2020. Exploring data aggregation in policy learning for vision-based urban autonomous driving, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11763–11773.

Qiao, Z., Schneider, J., Dolan, J.M., 2020. Behavior planning at urban intersections through hierarchical reinforcement learning. arXiv:2011.04697.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y., et al., 2009. Ros: an open-source robot operating system, in: ICRA workshop on open source software, Kobe, Japan. p. 5.

Ross, S., Gordon, G., Bagnell, D., 2011. 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, JMLR Workshop and Conference Proceedings. pp. 627–635.

Schwarting, W., Seyde, T., Gilitschenski, I., Liebenwein, L., Sander, R., Karaman, S., Rus, D., 2021. Deep latent competition: Learning to race using visual control policies in latent space. arXiv preprint arXiv:2102.09812 .

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 .

Song, Y., Scaramuzza, D., 2020. Learning high-level policies for model predictive control, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 7629–7636. doi:10.1109/IROS45743.2020.9340823.

Spencer, J., Choudhury, S., Barnes, M., Schmittle, M., Chiang, M., Ramadge, P., Srinivasa, S., 2020. Learning from interventions, in: Robotics: Science and Systems (RSS).

Spencer, J., Choudhury, S., Barnes, M., Schmittle, M., Chiang, M., Ramadge, P., Srinivasa, S., 2022. Expert intervention learning. Autonomous Robots 46, 99–113.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., Liu, C., 2018. A survey on deep transfer learning, in: International conference on artificial neural networks, Springer. pp. 270–279.

Tolani, V., Bansal, S., Faust, A., Tomlin, C., 2021. Visual navigation among humans with optimal control as a supervisor. IEEE Robotics and Automation Letters 6, 2288–2295.

Valletta, P., Pérez-Dattari, R., Kober, J., 2021. Imitation learning with inconsistent demonstrations through uncertainty-based data manipulation, in: 2021 IEEE International Conference on Robotics and Automation (ICRA), IEEE. pp. 3655–3661.

Veličković, P., Blundell, C., 2021. Neural algorithmic reasoning. arXiv preprint arXiv:2105.02761 .

Vitelli, M., Chang, Y., Ye, Y., Wołczyk, M., Osiński, B., Niendorf, M., Grimmett, H., Huang, Q., Jain, A., Ondruska, P., 2021. Safetynet: Safe planning for real-world self-driving vehicles using machine-learned policies. arXiv preprint arXiv:2109.13602 .

Waytowich, N.R., Goecks, V.G., Lawhern, V.J., 2018. Cycle-of-learning for autonomous systems from human interaction. arXiv preprint arXiv:1808.09572 .

Waytz, A., Heafner, J., Epley, N., 2014. The mind in the machine: Anthropomorphism increases trust in an autonomous vehicle. Journal of Experimental Social Psychology 52, 113–117.

Zamfirache, I.A., Precup, R.E., Roman, R.C., Petriu, E.M., 2022. Reinforcement learning-based control using q-learning and gravitational search algorithm with experimental validation on a nonlinear servo system. Information Sciences 583, 99–120.

Zanelli, A., Domahidi, A., Jerez, J., Morari, M., 2020. Forces nlp: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. International Journal of Control 93, 13–29.

Zhan, W., Sun, L., Wang, D., Shi, H., Clausse, A., Naumann, M., Kummerle, J., Konigshof, H., Stiller, C., de La Fortelle, A., et al., 2019. Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps. arXiv preprint arXiv:1910.03088 .

Zhou, B., Schwarting, W., Rus, D., Alonso-Mora, J., 2018. Joint multi-policy behavior estimation and receding-horizon trajectory planning for automated urban driving, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 2388–2394. doi:10.1109/ICRA.2018.8461138.