LVD-NMPC: A Learning-based Vision Dynamics Approach to Nonlinear Model Predictive Control for Autonomous Vehicles

Journal Title XX(X):1–11 ©The Author(s) 2021 Reprints and permission: sagepub.co.uk/journalsPermissions.nav DOI: 10.1177/ToBeAssigned www.sagepub.com/

SAGE

Sorin Grigorescu, Cosmin Ginerica, Mihai Zaha, Gigel Macesanu, Bogdan Trasnea

Abstract

In this paper, we introduce a learning-based vision dynamics approach to nonlinear model predictive control for autonomous vehicles, coined LVD-NMPC. LVD-NMPC uses an a-priori process model and a learned vision dynamics model used to calculate the dynamics of the driving scene, the controlled system's desired state trajectory and the weighting gains of the quadratic cost function optimized by a constrained predictive controller. The vision system is defined as a deep neural network designed to estimate the dynamics of the images scene. The input is based on historic sequences of sensory observations and vehicle states, integrated by an Augmented Memory component. Deep Q-Learning is used to train the deep network, which once trained can be used to also calculate the desired trajectory of the vehicle. We evaluate LVD-NMPC against a baseline Dynamic Window Approach (DWA) path planning executed using standard NMPC, as well as against the PilotNet neural network. Performance is measured in our simulation environment GridSim, on a real-world 1:8 scaled model car, as well as on a real size autonomous test vehicle and the nuScenes computer vision dataset.

Keywords

Autonomous vehicles, Autonomous driving, Vision dynamics, Learning control, Artificial Intelligence, Deep learning, Perception and control, Robot vision

Introduction

Research in the area of autonomous driving has been boosted in the last decade both by academia and industry. Autonomous vehicles are intelligent agents equipped with driving functions designed to understand their surroundings and derive control actions. As shown in the deep learning for autonomous driving survey of Grigorescu et al. 1, the driving functions are traditionally implemented as perception-planning-action pipelines. Recently, approaches based on End2End learning from Bojarski et al. 2 and Pan et al. 3, or the Deep Reinforcement Learning (DRL) shown by Kendall et al. 4 have also been proposed, although mostly as research prototypes.

In a modular perception-planning-action system, visual perception is most of the times decoupled from low-level control. A tighter coupling of perception and control was researched in the field of robotic manipulation with the concept of visual servoing, as in the case of the manipulation fault detector og Gu et al.⁵. However, this is not the case in autonomous vehicles, were intrinsic dependencies between the different modules of the driving functions are not taken into account.

This work is a contribution in the area of vision dynamics and control, where the proposed *Learning-based Vision Dynamics Nonlinear Model Predictive Control* (LVD-NMPC) algorithm is used for controlling autonomous vehicles. An introduction of the vision dynamics concept in learning control can be found in

the work of Grigorescu⁶. The block diagram of LVD-NMPC is shown in Fig. 1, where the main components are a vision dynamics model defined as a Deep Neural Network (DNN) and a constrained non-linear model predictive controller (NMPC) which receives input from the vision model. The model, trained using the Q-learning algorithm, calculates desired state trajectories and the tuning gains of the NMPC.

Synergies between data driven and classical control methods have been considered for imitation learning, where steering and acceleration control signals have been calculated in an End2End manner, as by Pan et al.³. Their approach is designed for driving environments with predefined boundaries, without any obstacles present on the driving track.

As shown by Grigorescu et al. ¹, traditional decoupled visual perception systems use visual localization to estimate the pose of the ego-vehicle relative to a reference trajectory, together with obstacle detection. The information is further used by a path and behavioral planner to determine a safe driving

The authors are with RovisLab: Robotics, Vision and Control Laboratory (www.rovislab.com), Transilvania University of Brasov, Romania and Elektrobit Automotive (www.elektrobit.com).

Corresponding author:

Sorin Grigorescu, Institute of Automation Transilvania University of Brasov Mihai Viteazu 5, Corp V, et. 3 500174 Brasov, Romania Email: s.grigorescu@unitbv.ro

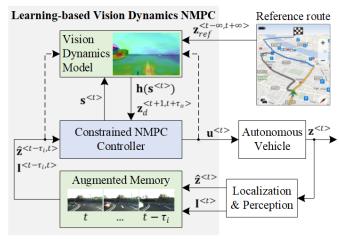


Figure 1. LVD-NMPC: Learning-based Vision Dynamics Nonlinear Model Predictive Control for autonomous vehicles. The dotted lines illustrate the data flow used during training.

trajectory, which is executed by the motion controller. In our work, we improve the traditional visual approach by replacing the classical perception-planning pipeline with a learned vision dynamics model. The model is used to calculate a safe driving trajectory, as well as to estimate the optimal tuning gains of the NMPC's quadratic cost function. Our formulation exploits the advantages of model-based control with the prediction capabilities of deep learning methods, enabling us to encapsulate the vision dynamics within the layers of a DNN. The key contributions of the paper are:

- the autonomous vehicle LVD-NMPC controller, based on an a-priori process model and a Vision Dynamics model;
- a deep neural network architecture acting as a nonlinear vision dynamics approximator used to estimate optimal desired state trajectories and the NMPC's tuning gains;
- a method for training the LVD-NMPC controller, based on imitation learning and the Q-Learning training approach.

The rest of the paper is organized as follows. The related work is covered in the next section. The methodology of LVD-NMPC is given in *Vision Dynamics Model Learning and Control System*, followed by the experimental results. Finally, the paper is summarized in the conclusions section.

Related Work

Recent years have witnessed a growing trend in applying deep learning techniques to autonomous driving, especially in the areas of *End2End* learning, as in the methods proposed by Pan et al.³, Fan et al.⁷ and Bojarski et al.², as well as in *Deep Reinforcement Learning* (DRL). Relevant algorithms for self-driving based on DLR can be found in the works of Kiran et al.⁸, Kendal et al.⁴ and Wulfmeier et al.⁹. Flavors of machine

learning techniques have also been encountered in more traditional control approaches, such as *Nonlinear Model Predictive Control* (NMPC), such as the uncertainty aware NMPC of Lucia and Karg 10 and the *Learning Controllers* of Ostafew et al. 11 and McKinnon and Schoellig 12 .

End2end learning, as described by Amini et al. ¹³, directly maps raw input data to control signals. The approach in LVD-NMPC is similar to the one considered by Pan et al.³. Compared with our method, their DNN policy is trained for agile driving on a predefined obstacle-free track. This approach limits the applicability of their system to autonomous driving, since a self-driving car has to navigate roads with dynamics obstacles and undefined lane boundaries. An end2end neural motion planner has been proposed by Zeng et al. 14, while Fan et al. 7 designed an End2End learning system to predict the drivable surface. The work considers no obstacle detection and avoidance, improving solely the perception system, without tackling the intrinsic dependencies between perception and low-level vehicle control.

Deep Reinforcement Learning (DRL) is a type of machine learning algorithm where agents are taught actions by interacting with their environment. An extensive review of DRL for autonomous driving has been published by Kiran et al.⁸. The main challenge with DRL for real-world physical systems is that the agent, in our case a self-driving car, learns by exploring its environment. A solution here is provided by Inverse Reinforcement Learning (IRL), which is an imitation learning method for solving Markov Decision Processes (MDPs). Wulfmeier et al. 9 have extended the Maximum Entropy IRL with a convolutional DNN for learning a navigation cost map in urban environments. However, such methods usually do not take into consideration the vehicle's state and the feedback loop required for lowlevel control.

Nonlinear Model Predictive Control (NMPC), as presented by Garcia et al. ¹⁵, is a control strategy which computes control actions by solving an optimization problem tailored around a nonlinear dynamic system model. In the last decades, it has been successfully applied to autonomous driving applications, both in research as well as in the automotive industry. NMPC controllers have been proposed by Nascimento et al. ¹⁶ and ¹⁷ for trajectory tracking in nonholonomic mobile robots. In order to deal with uncertainties, learning based approaches to model predictive control have been used by Lucia and Karg ¹⁰, as well as by Gango et. all ¹⁸ for approximating an explicit NMPC system.

Learning Controllers. Traditional feedback controllers, such as NMPC, make use of an a-priori model composed of fixed parameters. Unlike controllers with fixed parameters, learning controllers make use of training information to learn their models over time. In previous works, learning controllers have been introduced based on simple function approximators, such as the Gaussian Process (GP) modeling in the work of Ostafew

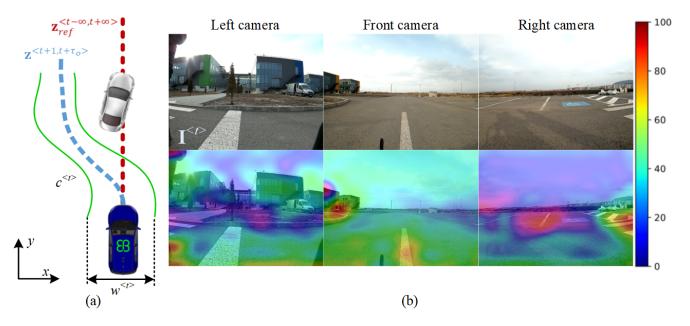


Figure 2. Autonomous driving problem definition. Given the vehicle's state, the global route (red line)and a set of sensory measurements, the goal is to calculate a safe path for tracking (blue line) over the control horizon $[t+1,t+\tau_o]$. The vision dynamics model estimates the curvature $c^{< t>}$ and width $w^{< t>}$ of the road. (a) Illustration of obstacle avoidance. (b) Observation $\mathbf{I}^{< t>}$ (top images) and activation maps (bottom images) within the vision dynamics model. The chromatic scale shows the contribution of the neurons in the activation maps at pixel level, where red corresponds to a high contribution and blue to almost no contribution (best viewed in color).

et al. 11 or bayesian regression algorithm of Mckinnon and Schoellig 12 .

In the light of the current approaches and their limitations, the LVD-NMPC method is proposed for encapsulating the driving scene's dynamics within a vision dynamics model, which adapts a constrained NMPC for executing the desired vehicle state trajectory.

Vision Dynamics Model Learning and Control System

Problem Definition

Fig. 2 shows a simple illustration of the autonomous driving problem. Given past vehicle states $\mathbf{z}^{< t - \tau_i, t>}$, a sequence of observations $\mathbf{I}^{< t - \tau_i, t>}$ and a global reference route for tracking $\mathbf{z}_{ref}^{< t - \infty, t + \infty>}$, the task is to calculate the control signals $\mathbf{u}_{opt}^{< t + 1>}$ for time t+1, such that the self-driving car tracks a safe trajectory $\mathbf{z}^{< t + 1, t + \tau_o>}$. Considering a discrete sampling time t, τ_i and τ_o are past and future temporal horizons, respectively.

The reference trajectory \mathbf{z}_{ref} represents the global route which should be followed by the vehicle, from its start position to destination. It can be given as a set of waypoints (e.g. GPS coordinates). Since \mathbf{z}_{ref} is a global trajectory, it is considered to vary in the $[t-\infty,t+\infty]$ interval. \mathbf{z}_{ref} is different from the desired trajectory $\mathbf{z}_d^{< t+1,t+\tau_o>}$, the later one being calculated for the interval $[t+1,t+\tau_o]$. \mathbf{z}_d is executed by the NMPC controller and must take into account the drivable area and the obstacles present in the scene.

The vehicle is modeled based on the kinematic bicycle model of a robot described by Paden et al. ¹⁹,

with position state $\mathbf{z}^{< t>} = (x^{< t>}, y^{< t>}, \rho^{< t>})$ and noslip assumptions. x, y and ρ represent the position and heading of the vehicle in the 2D driving plane, respectively. We apply the longitudinal velocity and the steering angle as control actions: $\mathbf{u}^{< t>} = (v^{< t>}_{cmd}, \omega^{< t>}_{cmd})$, where $\omega^{< t>}_{cmd}$ is the angle of the current velocity of the center of mass with respect to the longitudinal axis of the car, as proposed by Borrelli et al. 20 .

The driving scene is modeled as the vision dynamic state $(c^{< t>}, w^{< t>})$, where c and w are the curvature and traversable width of the road. A high w corresponds to relaxed constraints when estimating the desired trajectory, while a low w represents a path that has to be followed precisely in order to satisfy safety constraints. We consider a normalized value for the traversable width $w \in [0,1]$, where 1 is the maximum road width and 0 signals a non-traversable area, in which case the vehicle has to stop.

When acquiring training samples, the following quantities are stored as sequence data: the historic position states $\mathbf{z}^{< t-\tau_i,t>}$, the sensory information acquired from a front facing camera $\mathbf{I}^{< t-\tau_i,t>}$, the global reference trajectory $\mathbf{z}_{ref}^{< t-\tau_i,t+\tau_o>}$ and the control actions $\mathbf{u}^{< t-\tau_i,t>}$ recorded from a human driver. For practical reasons, the global reference trajectory is stored at sampling time t over the finite horizon $[t-\tau_i,t+\tau_o]$. A single image corresponds to an observation instance $\mathbf{I}^{< t>}$, while a continuous sequence of images is denoted as $\mathbf{I}^{< t-\tau_i,t>}$.

Control System Design

The block diagram of LVD-NMPC is shown in Fig. 1. Consider the following nonlinear, state-space system:

$$\mathbf{z}^{\langle t+1\rangle} = \mathbf{f}_{true}(\mathbf{z}^{\langle t\rangle}, \mathbf{u}^{\langle t\rangle}), \tag{1}$$

where $\mathbf{z}^{< t>} \sim \mathcal{N}(\bar{\mathbf{z}}^{< t>}, \sigma_f^2)$ is the observable state, $\mathbf{z}^{< t>} \in \mathbb{R}^n$, $\bar{\mathbf{z}}^{< t>}$ is the mean state and $\mathbf{u}^{< t>} \in \mathbb{R}^m$ is the control input at discrete time t. We assume that each state measurement is corrupted by zero-mean additive Gaussian noise with variance σ_f^2 . \mathbf{f}_{true} is approximated as a sum between an a-priori model and an experience-based vision dynamics model:

$$\mathbf{z}^{< t+1>} = \mathbf{f}(\mathbf{z}^{< t>}, \mathbf{u}^{< t>}) + \mathbf{h}(\mathbf{s}^{< t-\tau_i, t>}), \quad (2)$$
a-priori model vision dynamics model

where $\mathbf{s}^{< t>} \in \mathbb{R}^p$ represent environmental dependencies. This dependencies are composed of the system's and environment's states at time t, defined as:

$$\mathbf{s}^{\langle t \rangle} = (\mathbf{z}^{\langle t - \tau_i, t \rangle}, \mathbf{I}^{\langle t - \tau_i, t \rangle}), \tag{3}$$

where $\mathbf{s}^{< t>}$ represents the set of historic dependencies integrated along time interval $[t - \tau_i, t]$ by the so-called *Augmented Memory* component.

The models $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ are nonlinear process models. $\mathbf{f}(\cdot)$ is a known process model, representing our knowledge of $\mathbf{f}_{true}(\cdot)$, while $\mathbf{h}(\cdot)$ is the learned vision dynamics model. With the sampling time defined as δt , the nominal process model employed by LVD-NMPC is:

$$\mathbf{f}(\mathbf{z}^{}, \mathbf{u}^{}) = \mathbf{z}^{} + \delta t \begin{bmatrix} \cos(\rho^{} + \omega_{cmd}^{}) \\ \sin(\rho^{} + \omega_{cmd}^{}) \\ \frac{1}{L}\sin(\omega_{cmd}^{})) \end{bmatrix} v_{cmd}^{}$$

$$(4$$

where L is the length between the front and the rear wheel

We distinguish between the given reference trajectory $\mathbf{z}_{ref}^{< t-\infty,t+\infty>}$, which, from a control perspective, is practically infinite, and a desired state trajectory $\mathbf{z}_d^{< t+1,t+\tau_o>}$, calculated over a finite prediction horizon τ_o . The optimal future states are computed by a constrained NMPC controller, based on a desired state trajectory $\mathbf{z}_d^{< t+1,t+\tau_o>}$, estimated by the vision dynamics model.

The vision dynamics model learns to predict the driving scene's dynamics $(c^{< t>}, w^{< t>}) \sim \mathcal{N}(\bar{c}^{< t>}, \bar{w}^{< t>}, \sigma_h^2)$ from historic states integrated by the augmented memory component. The visual measurements are corrupted by zero-mean Gaussian noise with variance σ_h^2 . When queried, it returns a prediction for $c^{< t>}$ and $w^{< t>}$, given a sequence of historic states. The $\mathbf{h}(\cdot)$ model in Eq. 2 is calculated by multiplying c and w with three weighting constants. c is used to correct the heading, while w adapts the position of the vehicle in the 2D driving plane.

The scene's dynamics are used to calculate the desired trajectory $\mathbf{z}_d^{< t+1, t+\tau_o>}$, taking into account the global reference route $\mathbf{z}_{ref}^{< t-\infty, t+\infty>}$. $\mathbf{z}_d^{< t+1, t+\tau_o>}$ can be interpreted as the quantitative deviation of the vehicle from the reference route:

$$\mathbf{z}_{d}^{< t+1, t+\tau_{o}>} = \mathbf{z}_{ref}^{< t+1, t+\tau_{o}>} + \mathbf{z}_{c}^{< t+1, t+\tau_{o}>}, \quad (5)$$

where $\mathbf{z}_c^{< t+1,t+\tau_o>}$ is the vision dynamics estimation for the optimal trajectory of the vehicle's path, computed in xy-coordinates relative to the vehicle's pose. Once $\mathbf{h}(\cdot)$ has been queried, the coordinates of the predicted path are sampled over $[t+1,t+\tau_o]$ based on $(c^{< t>},w^{< t>})$, as:

$$y^{< t+i>} = w^{< t>} - \rho \cdot ^{< t>} (x^{< t+i>} - l) + 0.5 \cdot c^{< t>} \cdot x^{< t+i>}^2,$$
(6)

where $(t+i) \in [t+1, t+\tau_o]$ and l is a lookahead distance calculated for a sampling time of 3s.

As detailed in the Learning a Vision Dynamics Model section, a DNN is utilized to encode $\mathbf{h}(\cdot)$ and on top of $\mathbf{h}(\cdot)$, we define a quadratic cost function to be optimized by the constrained NMPC over discrete time interval $[t+1, t+\tau_o]$:

$$J(\mathbf{z}, \mathbf{u}) = (\mathbf{z}_d - \mathbf{z})^T \mathbf{Q} (\mathbf{z}_d - \mathbf{z}) + \mathbf{u}^T \mathbf{R} \mathbf{u}, \tag{7}$$

where $\mathbf{Q} \in \mathbb{R}^{\tau_o n \times \tau_o n}$ is a positive semi-definite scalar state weight matrix, $\mathbf{R} \in \mathbb{R}^{\tau_o M \times \tau_o M}$ is a positive definite scalar input weight matrix, $\mathbf{z}_d^{< t+1, t+\tau_o>}$ is a sequence of desired states estimated by the vision dynamics model:

$$\mathbf{z}_{d}^{< t+1, t+\tau_{o}>} = [\mathbf{z}_{d}^{< t+1>}, ..., \mathbf{z}_{d}^{< t+\tau_{o}>}], \tag{8}$$

where $\mathbf{z}^{< t+1, t+\tau_o>}$ is the sequence of predicted states:

$$\mathbf{z}^{\langle t+1, t+\tau_o \rangle} = [\mathbf{z}^{\langle t+1 \rangle}, ..., \mathbf{z}^{\langle t+\tau_o \rangle}], \tag{9}$$

and $\mathbf{u}^{\langle t,t+\tau_o-1\rangle}$ is the control input sequence:

$$\mathbf{u}^{\langle t, t + \tau_o - 1 \rangle} = [\mathbf{u}^{\langle t \rangle}, \dots, \mathbf{u}^{\langle t + \tau_o - 1 \rangle}]. \tag{10}$$

The curvature $c^{< t>}$ is calculated using a polynomial interpolation of the trajectory points in $\mathbf{z}_d^{< t+1,t+\tau_o>}$, while $w^{< t>}$ is proportional to the velocity of the agent (maximum velocity corresponds to $w^{< t>} = 1$, while no motion is represented as $w^{< t>} = 0$). Eq. 6 is used to convert between trajectories and $(c^{< t>}, w^{< t>})$. At training time, \mathbf{z}_d is given as human driven trajectories.

The traversable width $w^{< t>} \in [0,1]$ is used to automatically determine the ratios between the state and input weight matrices in Eq. 7. The operation is performed by weighting the coefficients of the main diagonals in R and Q based on w, as:

$$diag(Q) = w^{\langle t \rangle},\tag{11}$$

$$diag(R) = 1 - w^{\langle t \rangle}.$$
 (12)

In this way, more aggressive control actions can be chosen when the road has high traversability, indicated by a high value of w.

The constrained NMPC objective is to find a set of control actions that optimize the vehicle's motion over a given time horizon τ_o , while satisfying a set of hard and soft constraints:

$$(\mathbf{z}_{opt}^{\langle t+1\rangle}, \mathbf{u}_{opt}^{\langle t+1\rangle}) = \underset{\mathbf{z}, \mathbf{u}}{\operatorname{arg \, min}} J\left(\mathbf{z}^{\langle t+1, t+\tau_o \rangle}, \mathbf{u}^{\langle t+1, t+\tau_o \rangle}\right)$$
(13a)

such that
$$\mathbf{z}^{<0>} = \mathbf{z}^{}$$
 (13b)

$$\mathbf{z}^{\langle t+i+1\rangle} = \mathbf{f}(\mathbf{z}^{\langle t\rangle}, \mathbf{u}^{\langle t\rangle}) + \mathbf{h}(\mathbf{s}^{\langle t-\tau_i, t\rangle}), \qquad (13c)$$

$$\mathbf{e}_{\min}^{< t+i>} \le \mathbf{e}^{< t+i>} \le \mathbf{e}_{\max}^{< t+i>},$$
 (13d)

$$\mathbf{u}_{\min}^{\langle t+i\rangle} \le \mathbf{u}^{\langle t+i\rangle} \le \mathbf{u}_{\max}^{\langle t+i\rangle},\tag{13e}$$

$$\dot{\mathbf{u}}_{\min}^{< t+i>} \le \frac{\dot{\mathbf{u}}^{< t+i>} - \dot{\mathbf{u}}^{< t+i-1>}}{\Delta t} \le \dot{\mathbf{u}}_{\max}^{< t+i>}, \qquad (13f)$$

where $i=0,1,...,\tau_o-1,~\mathbf{z}^{<0>}$ is the initial state and Δt is the sampling time of the controller. $\mathbf{e}^{< t+i>} = \mathbf{z}_d^{t+i} - \mathbf{z}^{t+i}$ is the cross-track error, $\mathbf{e}_{min}^{< t+i>}$ and $\mathbf{e}_{max}^{< t+i>}$ are the lower and upper tracking bounds, respectively. Additionally, we consider $\mathbf{u}_{\min}^{< t+i>}$, $\dot{\mathbf{u}}_{\min}^{< t+i>}$ and $\mathbf{u}_{\max}^{< t+i>}$, $\dot{\mathbf{u}}_{\max}^{< t+i>}$ as lower and upper constraint bounds for the actuator and actuator rate of change, respectively. The DL-NMPC-RSD controller implements:

$$\mathbf{u}^{\langle t \rangle} = \mathbf{u}_{opt}^{\langle t+1 \rangle},\tag{14}$$

at each iteration t.

We leverage on the quadratic cost function from Eq. 7 and solve the nonlinear optimization problem described above using the Broyden-Fletcher-Goldfarb-Shanno algorithm of Fletcher²¹. The optimization problem from Eq. 13a has been solved in real-time using the C++ version of the open-source Automatic Control and Dynamic Optimization (ACADO) toolkit of Houska et al. ^{22,23}.

Learning a Vision Dynamics Model

The role of the vision dynamics model is to estimate the scene's dynamics $(c^{< t>}, w^{< t>})$ using the temporal information stored in the Augmented Memory component and the global reference trajectory $\mathbf{z}_{ref}^{< t-\infty,t+\infty>}$. For practical reasons, we consider the reference trajectory to vary within the finite time interval $[t-\tau_i,t+\tau_o]$. The model is encoded by combining a Convolutional DNN with the robust temporal predictions of two Long Short-Term Memory (LSTM) networks, one for each element in $(c^{< t>}, w^{< t>})$.

Although the model could learn local state sequences directly, as in the previous NeuroTrajectory work of Grigorescu et al. 24 , we have chosen to learn the vision dynamics model $(c^{< t>}, w^{< t>})$, which can be used both for state prediction in the form of ego-vehicle poses, as well as for tuning the NMPC's quadratic cost function.

Given a sequence of temporal observations $\mathbf{I}^{< t-\tau_i,t>}$: $\mathbb{R}^w \times \tau_i \to \mathbb{R}^w \times \tau_o$, the system's state $\mathbf{z}^{< t>} \in \mathbb{R}^n$ in $\mathbf{I}^{< t>}$ and the reference set-points $\mathbf{z}_{ref}^{< t+\tau_o>} \in \mathbb{R}^n$ in observation space at time t, the task is to learn $(c^{< t>}, w^{< t>})$ in order to navigate from state $\mathbf{z}^{< t>}$ to destination state $\mathbf{z}^{< t+\tau_o>}$.

In reinforcement learning terminology, the autonomous driving problem can be described as a *Partially Observable Markov Decision Process* (POMDP):

$$M := (I, S, Z_d, \mathcal{T}, R, \gamma), \tag{15}$$

where:

- \bullet I represents the sensory measurements;
- S is a finite set of states;
- Z_d is a set of trajectory sequences, used by the vehicle to navigate the driving environment measured via $\mathbf{I}^{< t-\tau_i,t>}$:
- $\mathcal{T}: S \times Z_d \times S \to [0,1]$ is a stochastic transition function:
- $R: S \times A \times S \to \mathbb{R}$ is a scalar reward controlling the computation of $\mathbf{z}_d^{< t+1, t+\tau_o>}$;
- γ is a discount factor regulating the importance of future versus immediate rewards.

The training objective is to find the desired trajectory that maximizes the associated cumulative future reward. We define the optimal action-value function $Q^*(\cdot, \cdot)$ for computing future discounted rewards for the starting $\mathbf{s}^{< t>}$ and control commands $\mathbf{u}^{< t+1, t+\tau_o>}$ for the state trajectory $\mathbf{z}_d^{< t+1, t+\tau_o>}$:

$$Q^*(\mathbf{s}, \mathbf{z}_d) = \max_{\pi} \mathbb{E}\left[R^{\langle \hat{t} \rangle} | \mathbf{s}^{\langle \hat{t} \rangle} = \mathbf{s}, \ \mathbf{z}_d^{\langle \hat{t} + 1, \hat{t} + \tau_o \rangle} = \mathbf{z}_d, \ \pi\right],$$
(16)

where π is an action, also known as policy, encapsulating a probability density function over a set of possible actions that can take place in a given state. For computing $Q^*(\mathbf{s}, \mathbf{z}_d)$, we propose the DNN from Fig. 3, where sequences of consecutive images are processed by a set of convolutional layers, before being fed to two LSTM branches.

Our deep network is processing sequences of continuous temporal observations from the Augmented Memory component. The Augmented memory acts as a buffer where the observations $\mathbf{I}^{< t-\tau_i,t>}$ and vehicle states $\mathbf{z}^{< t-\tau_i,t>}$ are synchronized and stored for the past temporal interval $[t-\tau_i,t]$, where t is the current time and τ_i is the maximum amount of time for which we store observations and states.

The architecture of our DNN is mainly based on convolutional, recurrent and fully connected layers, illustrated in Fig. 3 in gray, orange and blue, respectively. The sequence of four convolutional layers are used for encoding the visual input into a latent onedimensional intermediate representation that can be fed to the subsequent recurrent layers. In particular, the visual input is firstly passed through 64 convolutional filters of size 7×7 , followed by 32 filters of size $5 \times$ 5 and two blocks of 32 filters, both having a 3×3 size. The recurrent layers have been implemented as Long-Short Term Memory (LSTM) networks, where the input is represented by the sequences stored in the Augmented Memory. The first two LSTMs are used to encode the sequence of vehicle states $\mathbf{z}^{\langle t, t-\tau_i \rangle}$ and given reference trajectory $\mathbf{z}_{ref}^{< t-\tau_i, t+\tau_o>}$. The outputs are concatenated as a flatten extension of the onedimensional latent representation. The complete latent space is composed of 1.536 neurons, all activated using the Rectified Linear Unit (ReLU) activation

Figure 3. Vision Dynamics model implemented as a deep neural network. The training data consists of observation sequences, historic system states and reference state trajectories. A convolutional neural network firstly processes the observations data stream. Secondly, separate LSTM branches are responsible for calculating the road's curvature and width, which are then used to obtain the desired path.

function. For improving convergence during training, the intermediate representation reduced to 1.024 neuron units using fully an additional connected layer. The network then branches into two heads responsible for estimating the current curvature $c^{< t>}$ and width $w^{< t>}$ of the road, respectively. Both branches use an LSTM and two sequential fully connected layers of 128 neurons each, also activated using the ReLU function. Finally, the obtained curvature and road width are used to project the desired future trajectory of the vehicle $\mathbf{z}_d^{< t+1,t+\tau_o>}$, which in turn is used for calculating the optimal action-value function in Eq. 16.

One of the biggest challenges of using data-driven techniques for control is the so-called "DaGGer effect" described by Pan et al.³, which is a significant drop in performance when the training and testing trajectories are significantly different. One measure to cope with this phenomenon is to ensure that sufficient data is provided at training time, thus increasing the generalization capabilities of the neural network. The "DaGGer effect" is also the main reason why a deep Qlearning approach, which uses the reward function to explore different trajectories, is preferred over standard supervised imitation learning. In the next section, it is shown that a high generalization can be achieved, demonstrating that LVD-NMPC can safely navigate the driving environment, even if the encountered obstacles were not given at training time.

Experiments

The performance of LVD-NMPC was benchmarked against a baseline non-learning approach, coined DWA-NMPC, as well as against PilotNet of Bojarski et al. ². DWA-NMPC uses the Dynamic Window Approach (DWA) proposed by Fox et al. ^{25,26} for path planning and a constraint NMPC for motion control, relying for perception on the YoloV3 algorithm of Redmon and Farhadi ²⁷.

LVD-NMPC has been tested on three different environments: I) in the GridSim simulator, II) for indoor navigation using the 1:8 scaled model car from Fig. 4(a) and III) on real-world driving with the full

scale autonomous driving test vehicle from Fig. 4(b), as well as on the nuScenes computer vision dataset.

Competing Algorithms and Performance Metrics

DWA has been implemented based on the Robot Operating System (ROS) DWA local planner, taking into account obstacles provided by the YoloV3 object detector of Redmon and Farhadi 27 . In the case of the PilotNet algorithm proposed by Bojarski et at. 2 , the input images are mapped directly to the steering command of the vehicle. The steering commands are executed with an incremental value of 0.01deg, dependent on the PilotNet's output, while the velocity is controlled using a proportional feedback law, with gain K=1.6.

To assess the success rate of each algorithm, the ground truth is considered as the path driven by a human driver. The ground truth of the curvature and road width is calculated as for the trajectory sequences Z_d in the POMDP setup from the Learning a Vision Dynamics Model section. Namely, the curvature is given by the polynomial interpolation of the human driven path, while the road width's ground truth is correlated to the longitudinal velocity of the vehicle.

Ideally, each method should navigate the environment collision-free, at maximum speed and as close as possible to the ground truth. As pointed out by Codevilla et al. 28, there are limits to the offline policy evaluation employed in Experiments III, which can be partially overcome by choosing an appropriate evaluation metric. The cumulative speed-weighted absolute error of Codevilla et al. 28 has been chosen as performance metric. This metric is intended to equally quantify Experiments I and II, which are pure closed loop experiments, with the offline evaluation performed in Experiments III. Additionally, the average speed, the curvature error e_c and the processing time have been evaluated. e_c represents the difference between the estimated and actual path curvature, calculated using polynomial interpolation, while e_{xy} is defined as:

$$e_{xy} = \frac{1}{m} \left\| \sum_{t=0}^{T} (\hat{\mathbf{p}}^{\langle i+t \rangle} - \mathbf{p}^{\langle i+t \rangle}) \cdot v_{i+t} \right\|_{1},$$
 (17)



Figure 4. Test vehicles used for data acquisition and testing. (a) Audi 1:8 scaled model car. (b) Real-sized VW Passat autonomous test vehicle.

where $\hat{\mathbf{p}}$ and \mathbf{p} are coordinates on the estimated and human driven trajectories, respectively. l is a lookahead distance calculated for a prediction horizon $\tau_o = 3$ s. In order to compare the three competing methods, the metric in Eq. 17 quantifies the total system error.

The percentage of times an algorithm crashed the vehicle and the number of times the destination goals were reached have also been measured for experiments I and II. In the following, we discuss the obtained values of the computed metrices for the three competing algorithms in the experiments, as summarized in Table 1.

Experiment I: Simulation Algorithm Comparison

The first set of experiments are simulations over 10 goal-navigation trials performed in GridSim. GridSim, proposed by Trasnea et al. ²⁹, is our autonomous driving simulation simulation engine that uses kinematic models to generate synthetic occupancy grids from simulated sensors. It allows for multiple driving scenarios to be easily represented and loaded into the simulator. The simulation parameters are the same as in the NeuroTrajectory state trajectory planning approach of Grigorescu et al. ²⁴.

For training PilotNet and LDV-NMPC, the goal navigation task was run over 10 driving routes, as follows. A trajectories database has been mapped to sensory information, while the ego-vehicle was manually driven by a person. This resulted in over 200.000 pairs of data samples. Since GridSim provides observations in the form of occupancy grids, the raw input data consisted of distances between the vehicle and the obstacles, sampled at an angle of 2°, without any visual observation being considered. Due to the simplified structure of the occupancy grid observations, which does not consider visual data, the sets of simulated experiments are used as simple sanity checks for evaluating the competing methods.

Overall, as indicated by the values of the percentage of crashes (13%), goal reach (87%) and position (14.03m) and curvature (0.12) errors from Experiments set I (GridSim simulation) in the first row of Table 1, the

analytical non-learning DWA-NMPC approach provided top quantitative results in this experiment. This comes to no surprise, since the full state of the vehicle and the precise locations of the obstacles are known a-priori. Nevertheless, as pointed by the results for GridSim simulation if Table 1, LVD-NMPC came relatively close to similar values for the computed metrices (15% crashes, 85% goal reaching, an average speed of 6.17m/s, 16.95m and 0.14 for position and curvature errors, respectively, as well as 38ms computation time), even outperforming the other methods is terms of vehicle speed. As also shown in the next experiments, PilotNet provided the fastest processing time, mainly due to the fact that the most cost efficient component of PilotNet is the forward propagation of the input data over its neural network. The DaGGer effect could be avoided, since the simulation environment allowed us to gather as much training data as necessary.

Experiment II: Indoor Algorithm Comparison

In this experiment we have tested using the 1:8 scaled Audi model-car vehicle from Fig. 4(a), with different indoor navigation tasks. The reference routes which the car had to follow were defined as straight lines, sinusoids, circles and a 75m prerecorded loop. The first set of 10 trials were performed without any obstacles present on the reference routes, while the second 10 trials set contained static and dynamic obstacles. The static obstacles are composed of cardboard boxes which the model-car can sidestep, while the dynamic obstacles are represented by two other model-cars, manually driven at different speeds and with different trajectories. The state of the vehicle was measured using wheels odometry and an Inertial Measurement Unit (IMU).

LVD-NMPC provided the highest quantitative results, apart from the processing time, which was better for PilotNet. The main reasons for DWA-NMPC's increase in computation time comes from uncertainties in environment perception and localization. This is

Additional information available at www.rovislab.com/gridsim.html.

Experiments	Method	Crash	Goal reach	Avg. $peed [m/s]$	$e_{xy} \pm \text{ STD [m]}$	$e_c \pm \text{STD}$	Processing time [ms]
\overline{I}	DWA-NMPC	13 %	87%	4.74	14.03	0.12	77
$\operatorname{GridSim}$	PilotNet	35%	65%	5.29	30.15	0.37	21
simulation	LVD-NMPC	15%	85%	6.17	16.95	0.14	38
II	DWA-NMPC	30%	70%	0.83	0.21	0.26	98
Indoor	PilotNet	40%	60%	1.28	1.52	0.83	53
navigation	LVD-NMPC	20 %	80%	1.61	0.21	0.20	61
III	DWA-NMPC	-	-	25	56.38	0.14	97
Real-world	PilotNet	-	-	25	118.36	0.28	54
driving	LVD-NMPC	-	-	25	49.04	0.08	63
III	DWA-NMPC	-	-	4.44	41.81	0.11	101
nuScenes	PilotNet	-	-	4.44	93.50	0.26	58
dataset	LVD-NMPC	-	-	4.44	37.02	0.07	66

Table 1. Results for experiments I, II and III, where STD represents the standard deviation.

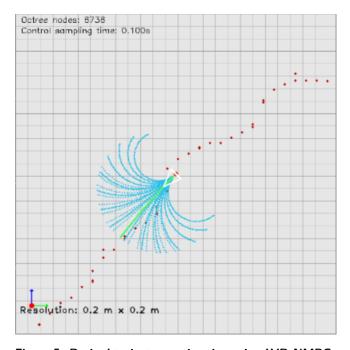


Figure 5. Desired trajectory estimation using LVD-NMPC. The vehicle selects the best desired trajectory (green) from the set of possible candidates (blue) (best viewed in color).

a common phenomenon encountered in decoupled processing pipelines, where a decrease in perception accuracy produces a decrease in control performance and vice versa. This is not the case with LVD-NMPC, since perception is tightly coupled to motion control through our vision dynamics model. The model-based nature of our algorithm allowed us to outperform a model-free method such as PilotNet, which tends to have a jittering effect in its control output.

A snapshot from the control loop of LVD-NMPC is shown in Fig. 5, where the desired trajectory (depicted in green) is calculated using the output of the proposed deep neural network from Fig 3 and a set of candidate trajectories (shown in blue) calculated using the dynamic model of the vehicle from Eq. 4. We have observed that the advantage of LVD-NMPC relies in the combination of the analytical dynamic model of the car, which is subsequently adapted to unseen situations by

the deep network's estimations, as specified in the state estimation equation 2.

Fig. 6 illustrates velocity, steering, position errors and heading errors for a trial distance of 10m, containing an obstacle at position (0m,5m) on the reference route. The obstacle's position is relative to the starting position. It can be observed that LVD-NMPC has a smoother trajectory, starting to adapt the control inputs earlier than the competing methods. DWA-NMPC and PilotNet are more aggressive, a jittering effect being encountered in both control outputs. This uncertainty is correlated to the accuracy of the perception system and the lack of a system model in the case of DWA-NMPC and PilotNet, respectively.

In order to assess the behavior of the methods with respect to the DaGGer effect, we have placed on the reference route obstacles which were not given at training time. The DNNs embedded in LVD-NMPC and PilotNet managed to bypass the obstacles, leveraging on environment landmarks present in the training data. This points to the fact that although the learning based approaches were able to correctly adjust the vehicle's trajectory, they still require enough training data to recognize environment landmarks.

Experiment III: On-Road Algorithm Comparison

Finally, the third experiment tested LVD-NMPC on over 100km of real-world driving in different environments. In total, approx. 463.000 samples were acquired and varied into 75% training and 25% test sets. This data has been used for training the deep network from Fig. 3 in a self-supervised fashion, where pairs of observations (images) and labels (driving trajectories) were given as input to the optimization function in Eq. 16. Although the training performed in a self-supervised manner, the same real-world data could be used to train a DNN solely based on the observation using Inverse Deep Reinforcement Learning. Such a method was proposed by Wulfmeier et al.⁹, where cost functions for mobile navigation have been learned using Inverse DRL. Nevertheless, the most straightforward approach to self-driving using DRL is to learn the parameters of the network in a simulated environment, where a car would autonomously explore its driving environment.

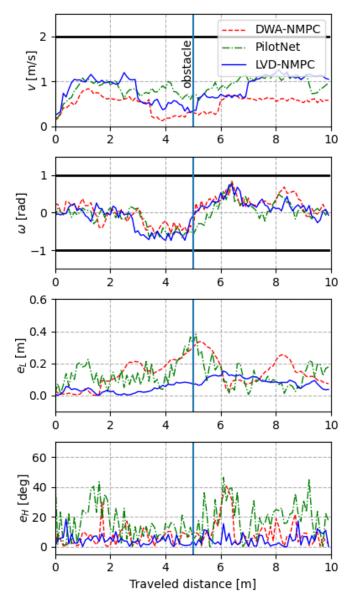


Figure 6. Velocity, steering, position errors and heading errors vs. a $10\mathrm{m}$ travel distance using the model-car from Fig. 4(a). LVD-NMPC provides a smoother vehicle trajectory, leveraging on learned obstacles and environmental landmarks when learning the vision dynamics model. Actuator constraints are shown with black lines.

In such a system, the reward function would change the position and orientation of the car based on the visual input. Finally, the challenge in this case would be the mapping of the trained DNN to the real-world vehicle.

In addition to our own real-world driving data, we have evaluated the competing methods on the nuScenes dataset (https://www.nuscenes.org/). Among different benchmarking datasets, we have chosen nuScenes due to its sensor setup and odometry information. The data collection contains over 15h of driving data divided into 1000 driving scenes collected in Boston and Singapore, each scene having a length of 20s. 242km were covered at an average speed of 16km/h. We have converted the ego-vehicle poses to our global 2D

reference path coordinates. The original $1600px \times 900px$ resolution has been downscaled to $640px \times 360px$.

We have encountered similar results to the ones in experiment II, with LVD-NMPC providing a more accurate estimate of control outputs. Due to the fact that the test vehicle is a real-sized car (as opposed to the 1:8 Audi model-car), the values of e_{xy} are larger that in experiment II. On the other hand, the curvature error is lower, since the driving itself contained less curves than in the case of the indoor navigation experiment. The results on the nuScene dataset were slightly better, mainly due to the relative low speed of the vehicle during data acquisition.

As shown by the results from the real-world experiments I and II, given in Table 1, our model delivers an inference time of slightly above 60ms on an embedded NVIDIA AGX Xavier development board, equipped with an integrated Volta GPU processor, having a 512 CUDA cores. Depending on the speed of the vehicle, this inference time could be sufficient if the vehicle is traveling at a relatively low speed. However, an increase in computation time is required for high speed vehicles, where the environment also varies with a higher speed.

The metrics used in Table 1 could be aggregated together in a single metric, where each element, that is percentages of crashes and goal reach, average speed, position and curvature errors and processing time, would be combined in a single weighted function. Nevertheless, in this case the intrinsic values of the individual measurements would be lost. As an example, a model yielding an optimal metric value due to high processing time could crash more often than a model which is slower in terms of computation time.

Conclusions

This paper introduces the *learning-based vision dynamics nonlinear model predictive control* approach for controlling autonomous vehicles. The method uses a deep neural network as a vision dynamics model which estimates both the desired state trajectory of the vehicle, given as input to a constrained non-linear model predictive controller, as well as the weighting gains of the aforementioned controller. One of the advantages of LVD-NMPC is that the Q-learning training is self-supervised, without requiring the manual annotation of the training data. The experimental results show the robustness of the approach with respect to state-of-the-art competing algorithms, both classical, as well as learning based.

As future work, we plan to investigate the stability of LVD-NMPC, especially in relation to the functional safety requirements needed for automotive grade deployment. Being already implemented on an embedded device, that is the NVIDIA AGX Xavier, we believe that the controller can be used on real-world cars, provided that the safety requirements are meet. Setting safety aside, its current implementation is directly linked to the computation power of the vehicle's

computer. The faster its deep learning accelerator is, the more dynamic situations LVD-NMPC could cope with.

Acknowledgements

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 800928, European Processor Initiative EPI.

References

- Grigorescu S, Trasnea B, Cocias T et al. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics* 2019;
- Bojarski M, Testa DD, Dworakowski D et al. End to end learning for self-driving cars. CoRR 2016; abs/1604.07316. 1604.07316.
- Pan Y, Cheng CA, Saigol K et al. Imitation Learning for Agile Autonomous Driving. In *Int. Journal of Robotics Research (IJRR)*, volume 39. pp. 286–302.
- Kendall A, Hawke J, Janz D et al. Learning to drive in a day. In 2019 Int. Conf. on Robotics and Automation (ICRA). pp. 8248–8254.
- Gu H, Wang H, Xu F et al. Active fault detection of soft manipulator in visual servoing. *IEEE Transactions* on *Industrial Electronics* 2020; : 1–1DOI:10.1109/TIE. 2020.3028813.
- Grigorescu S. Vision dynamics-based learning control. In Zhang D and Wei B (eds.) *Learning Control*. Elsevier. ISBN 978-0-12-822314-7, 2021. pp. 243-257.
- Fan R CP Wang H and M L. Sne-roadseg: Incorporating surface normal information into semantic segmentation for accurate freespace detection. In *European Conference* on Computer Vision ECCV 2020, volume 12375. DOI: 10.1109/TIE.2020.3028813.
- Kiran BR, Sobh I, Talpaert V et al. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems* 2021; : 1–18DOI:10.1109/TITS.2021.3054625.
- Wulfmeier M, Wang DZ and Posner I. Watch this: Scalable cost-function learning for path planning in urban environments. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2016).
- 10. Lucia S and Karg B. A deep learning-based approach to robust nonlinear model predictive control. IFAC-PapersOnLine 2018; 51(20): 511-516. DOI: https://doi.org/10.1016/j.ifacol.2018.11.038. URL https://www.sciencedirect.com/science/article/pii/S2405896318326958. 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018.
- Ostafew CJ, Schoellig AP and Barfoot TD. Robust constrained learning-based NMPC enabling reliable mobile robot path tracking. *International Journal of Robotics Research* 2016; 35(13): 1547–1563.
- McKinnon CD and Schoellig AP. Learn fast, forget slow: Safe predictive control for systems with locally linear actuator dynamics performing repetitive tasks. *IEEE Robotics and Automation Letters* 2019; 4(2): 2180–2187.
- 13. Amini A, Gilitschenski I, Phillips J et al. Learning robust control policies for end-to-end autonomous

- driving from data-driven simulation. *IEEE Robotics* and Automation Letters 2020; 5(2): 1143–1150. DOI: 10.1109/LRA.2020.2966414.
- Zeng W, Luo W, Suo S et al. End-to-end interpretable neural motion planner. In *IEEE Conf. on Computer* Vision and Pattern Recognition (CVPR). pp. 8652–8661.
- 15. Garcia CE, Prett DM and Morari M. Model predictive control: Theory and practice a survey. *Automatica* 1989; 25(3): 335 348.
- 16. d Nascimento TP, Basso GF, Dórea CET et al. Perception-driven motion control based on stochastic nonlinear model predictive controllers. *IEEE/ASME Transactions on Mechatronics* 2019; 24(4): 1751–1762. DOI:10.1109/TMECH.2019.2916562.
- 17. d Nascimento TP, Dórea CET and Gonçalves LMG. Nonlinear model predictive control for trajectory tracking of nonholonomic mobile robots: A modified approach. *International Journal of Advanced Robotic* Systems 2018; : 1–14DOI:10.1177/1729881418760461.
- 18. Gángó D, Péni T and Tóth R. Learning based approximate model predictive control for nonlinear systems**this work was partially supported by the jános bolyai research scholarship of the hungarian academy of sciences and the Únkp-18-4 new national excellence program of the ministry of human capacities. it was also supported by the research program titled "exploring the mathematical foundations of artificial intelligence (2018-1.2.1-nkp-00008)". IFAC-PapersOnLine 2019; 52(28): 152-157. DOI:https://doi.org/10.1016/j.ifacol.2019.12. 363. URL https://www.sciencedirect.com/science/article/pii/S2405896319322633. 3rd IFAC Workshop on Linear Parameter Varying Systems LPVS 2019.
- 19. Paden B, Cáp M, Yong SZ et al. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Trans Intelligent Vehicles* 2016; 1(1): 33–55.
- Kong J, Pfeiffer M, Schildbach G et al. Kinematic and dynamic vehicle models for autonomous driving control design. In 2015 IEEE Intelligent Vehicles Symposium (IV). pp. 1094–1099.
- Fletcher R. Practical Methods of Optimization. Second ed. New York, NY, USA: John Wiley & Sons, 1987.
- Houska B, Ferreau H and Diehl M. ACADO Toolkit An Open Source Framework for Automatic Control and Dynamic Optimization. Optimal Control Applications and Methods 2011; 32(3): 298–312.
- 23. Houska B, Ferreau H and Diehl M. An Auto-Generated Real-Time Iteration Algorithm for Nonlinear MPC in the Microsecond Range. *Automatica* 2011; 47(10): 2279–2285. DOI:10.1016/j.automatica.2011.08.020.
- Grigorescu SM, Trasnea B, Marina L et al. Neurotrajectory: A neuroevolutionary approach to local state trajectory learning for autonomous vehicles. *IEEE Robotics and Automation Letters* 2019; 4(4): 3441–3448.
- Fox D, Burgard W and Thrun S. The dynamic window approach to collision avoidance. *Robotics Automation Magazine*, *IEEE* 1997; 4(1): 23–33.
- 26. Lu Chang CJ Liang Shan and Dai Y. Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment. *Autonomous Robots* 2021; 45: 51–76.

- 27. Redmon J and Farhadi A. Yolov3: An Incremental Improvement. $arXiv\ preprint\ arXiv:180402767\ 2018;$.
- 28. Codevilla F, López A, Koltun V et al. On offline evaluation of vision-based driving models. In 15th European Conference Computer Vision (ECCV). Munich, Germany, pp. 246–262.
- 29. Trasnea B, Marina L, Vasilcoi A et al. Gridsim: A simulated vehicle kinematics engine for deep neuroevolutionary control in autonomous driving. In *Int. Conf. on Robotic Computing IRC 2019*. Naples, Italy.