2020 IEEE Intelligent Vehicles Symposium (IV)
October 20-23, 2020. Las Vegas, USA

A Novel Approach to Neural Network-based Motion Cueing Algorithm for a Driving Simulator

Ahmet Burakhan Koyuncu*, Emeç Erçelik*, Eduard Comulada-Simpson†, Joost Venrooij†, Mohsen Kaboli† and Alois Knoll*

*Department of Informatics
Technical University of Munich, Munich, Germany
Email: {burakhan.koyuncu, emec.ercelik}@tum.de, knoll@mytum.de
†Research, New Technologies, Innovations
BMW Group, Munich, Germany
Email: {eduard.comulada-simpson, joost.venrooij, mohsen.kaboli}@bmw.de

***Abstract*—Generating realistic motion in a motion-based (dynamic) driving simulator is challenging due to the limited workspace of the motion system of the simulator compared to the motion range of the simulated vehicle. Motion Cueing Algorithms (MCAs) render accelerations by controlling the motion system of the simulators to provide the driver with a realistic driving experience. Commonly used methods such as Classical Washout-based MCA (CW-MCA) typically achieves suboptimal results due to scaling and filtering, which results in an inefficient usage of the workspace. The Model Predictive Control-based MCA (MPC-MCA) has been shown to achieve superior results and more efficient workspace use. However, it's performance is in practice constrained due to the computationally expensive operations and the requirement of an accurate prediction of future vehicle states. Finally, the Optimal Control (OC) has been shown to provide optimal cueing in an open-loop setup wherein the precalculated control signals are re-played to the driver. However, OC cannot be used in real-time with the driver-in-the-loop. Our work introduces a novel Neural Network-based MCA (NN-MCA), which is trained to imitate the behavior of the OC. After training, the NN-MCA provides an approximated model of the OC, which can run in real-time with the driver in-the-loop, while achieving similar quality. The experiments demonstrate the potential of this approach through objective evaluations of the generated motion-cues on the simulator model and the real simulator. A demonstration video for the performance comparison of the CW-MCA, Optimal-Control-based MCA (OC-MCA) and our proposed method is available at http://go.tum.de/708350.**

***Index Terms*—Neural Networks, Imitation Learning, Optimal Control, Motion Cueing Algorithms, Driving Simulators**

I. Introduction

Driving simulators have become an important part of the product development cycle in the automotive industry. They enable more cost-effective and shorter development processes [1], [2], and provide safe, reproducible, and controlled environments to assess, for example, the comfort of the driver by conducting subjective experiments [3], [4]. Regarding the benefits of the simulators, the automotive industry is increasingly concentrating on replacing real vehicle studies with driving simulations especially for validating the performance of Advanced Driver Assistance Systems (ADAS) and autonomous driving prototypes. In order to ensure the validity of the driving simulator studies, the quality of the driving experience is of the utmost importance. In dynamic simulators, the driving experience can be improved by reproducing the virtual vehicle motion in the physical world by means of a motion system.

However, reproducing the virtual motion with the simulator's motion system exactly ("one-to-one") is in practice impossible due to the constrained workspace of the motion system, compared to the motion range of the simulated vehicle, with which the driver can easily achieve high-amplitude and long-lasting accelerations. Especially, in the case of the long-lasting accelerations, only a limited amount of the whole motion is reproducible, since workspace limits can be exceeded in a couple of seconds. In order to achieve realistic motion in driving simulators, control strategies called Motion Cueing Algorithms (MCAs) were developed.

One of the most common strategies for motion-cueing is the so-called washout-based MCA, such as Classical Washout (CW) which has been used in flight simulators and later in driving simulators [5]–[7]. This technique is based on scaling and filtering the vehicle accelerations while applying a washout filter which creates an active "pull" towards the center of the workspace in order to avoid reaching the workspace limits. The main advantage of this strategy is the real-time capability and intuitive nature of tuning due to its simple design, where each tuning parameter has a physical interpretation. However, the main disadvantages are a relatively low quality of motion cueing, inefficient use of workspace due to scaling and filtering of inputs, and not considering the workspace limits in the algorithm explicitly.

Dagdelen et.al. [8] proposed to use Model Predictive Control (MPC) for motion-cueing in driving simulators. The MPC iteratively minimizes the error between the future predicted motion of the virtual vehicle and the simulator while considering the constraints of the workspace. The

978-1-7281-6673-5/20/$31.00 ©2020 IEEE

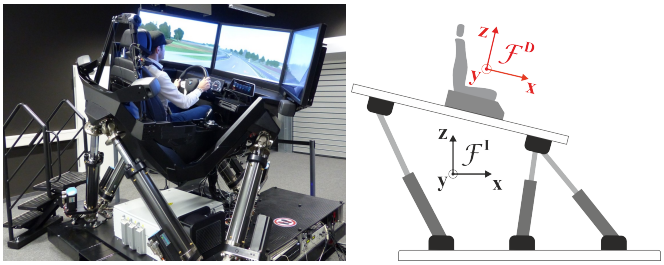
optimizer solves the problem for N steps (also called "prediction horizon") and applies the M steps of the resulting control sequence (also called "control horizon"). The main advantages of MPC-based MCA (MPC-MCA) are that the workspace limits and the future state of the simulator are considered in the optimization. However, since the method is computationally expensive, the method has to be constrained or simplified in order to ensure real-time performance. Additionally, the method requires an accurate prediction of future vehicle states which may not be available at run-time.

MCAs can also be used in an open-loop setup, where the route is driven in advance, and the control commands for the simulator are calculated offline accordingly. Afterward, the calculated motions and the recorded vehicle motion in the environment are replayed to the "driver" in the simulator, which is now a passive observer. Since no real-time interaction between the driver and the virtual environment is possible, this setup is commonly used in studies that involve autonomous vehicles. In case of MPC-MCA, the prediction and control horizon can be selected as the length of the entire trajectory since the whole trajectory of the driver is known a priori. As a consequence, the problem turns into an Optimal Control (OC) problem with constraints. The optimization parameters can be tuned in advance in order to obtain the *best* possible (optimal) motion-cueing while aiming to use the maximum amount of the workspace. However, since the objective function of OC specifically requires the entire trajectory of the vehicle motion in advance, it cannot be used for a driver-in-the-loop setup, where the driver is actively involved in the driving task.

In the current work, we propose a novel approach to motion cueing by utilizing a Neural Network-based MCA (NN-MCA) which imitates an Optimal-Control-based MCA (OC-MCA). Hence, the NN-MCA offers a real-time implementation of an OC-MCA, which would otherwise not be able to operate in a driver-in-the-loop setup. In order to obtain the NN-MCA the following procedure was followed: first, a variety of maneuvers are collected from multiple drivers in a static simulator. Then the optimal control signals (motion-cues) for a 6-Degrees of Freedom (DoF) dynamic simulator (Fig. 1a) are calculated offline by using the OC-MCA as described above. Based on these results, the NN-MCA is trained to map driver accelerations to optimal control signals calculated by the OC-MCA. We showed that by learning from OC, the resulting NN-MCA generates control signals which are near-optimal for the whole trajectory. After the training phase, NN-MCA can be used as a stand-alone MCA, basically providing a real-time implementation of the OC-MCA. Hence, the NN-MCA can function in a driver-in-the-loop setup with a similar performance as the OC.

II. Related Work

To the best of our knowledge, few works have been published on the use of machine learning techniques in the context of motion cueing. Mohammadi et.al. [10] showed that the MPC-MCA can be improved by including a prediction of the driver's future accelerations. In order to approximate the driver's behavior, a Recurrent Neural Network (RNN) was trained with several driving maneuvers generated in a virtual environment. Instead of using a constant future reference, the trained RNN model provides MPC with the future reference trajectories. Compared to a MPC-MCA with constant reference, their approach reduces the Root Mean Squared Error (RMSE) of sensation about 35% for a test trajectory.



(a) (b)

Fig. 1. (a) The dynamic simulator including the hexapod motion system manufactured by Cruden [9] used in our work. (b) Illustration of the used dynamic simulator with the inertial coordinate system $\mathcal{F}^{\mathrm{I}}$ of the simulator and the body-fixed coordinate system $\mathcal{F}^{\mathrm{D}}$ (driver's coordinate frame) in the real world.

Rengifo et.al. [11] proposed to use a continuous-time RNN, in order to solve the optimization problem of a MPC-MCA by replacing the standard QP-Solver. Using this method, a faster computation was achieved without a significant performance drop. However, the resulting algorithm was still constrained by the limitations of MPC.

Outside the field of motion cueing, there are a few examples where an MPC controller was imitated using deep learning methods. Drgoňa et.al. [12] showed that a MPC can be approximated by a neural network for the control of heating, cooling, ventilation and air-conditioning systems in buildings. Since the MPC requires extensive hardware and software, a neural network-based MPC was deployed to low-level hardware with real-time capability. Furthermore, Pan et.al. [13] demonstrated a similar technique by imitating the MPC in order to perform high-speed off-road autonomous driving with AutoRally car [14]. However, our contribution differs substantially from mentioned prior methods where [12], [13] focused on learning from the MPC with a short horizon for stability, the OC-MCA used in our work considers the full trajectory as prediction horizon, in order to achieve control signals which are optimal for the whole trajectory. We solved the problem of accumulating error, which is a known problem in supervised learning techniques for control tasks [15], [16], by training the neural network for predicting not only the acceleration commands for the simulator but the full next state of the simulator. In order to reduce the possible inconsistency between the network's outputs we also introduced an additional cost.

III. Preliminaries

In this section we introduce the human motion perception, motion cueing in driving simulators in general and the OC-MCA in detail, which the NN-MCA was trained to imitate.

A. Human Motion Perception

Human motion perception has a number of limitations that can be exploited in motion cueing. One of these limitations is that, in the absence of reliable visual information, the vestibular system cannot differentiate between translation and rotation. Since, instead of the effective inertial force, the vestibular system detects the forces (so called specific forces), which are the result of the superposition of the gravitation and additional accelerations acting on the human head [17]–[19]. The perceptional ambiguity is exploited when we use slow rotations to provide the illusion of sustained acceleration. Instead of accelerating the simulator we slowly tilt the simulator. This is called tilt-coordination. Fig. 2 illustrates an example use of the tilt-coordination, where the longitudinal acceleration and pitching of head position result in the same resulting force detected by the vestibular system. Subsequently, the same ambiguous specific force for both cases is obtained. Similarly, the same ambiguous specific force is generated when a lateral acceleration or a roll angle is present.

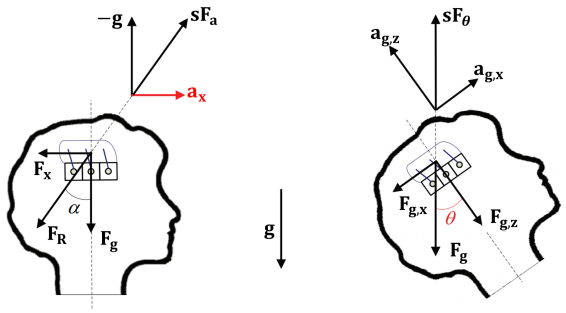


Fig. 2. The illustration of specific forces acting on human body under longitudinal acceleration (left) and rotation by angle an angle θ about y-axis (right) (adapted from [17]).

Another limitation of the perceptual system is that there is a certain threshold below which motions are not or not often perceived. We can use these perceptual thresholds to move the simulator without the driver feeling this motion. This, for example, is used in tilt-coordination (using sub-threshold rotations) [20] and prepositioning (using sub-threshold accelerations) [21], [22] to position the simulator to better utilize the available workspace. In our work, we used thresholds for the longitudinal, lateral and vertical specific forces ($\boldsymbol{sF}_{x,y,z}$) of 0.17, 0.17 and $0.28\,\mathrm{m/s^2}$ according to [18]. The threshold values for the angular velocities around the corresponding axes ($\boldsymbol{\omega}_{x,y,z}$) are selected as $2.04\,^\circ/\mathrm{s}$ for each DoF according to [23].

B. Motion Cueing in Driving Simulators

In a typical driving simulator setup, the driver navigates a virtual vehicle in a virtual environment by using the external hardware such as steering wheel, gas and brake pedal. A dynamic vehicle model calculates the motion of the simulated vehicle corresponding to the driver commands, and - based on these motions - the MCA generates the control signals for the motion system of the simulator with respect to the accelerations or specific forces of the vehicle model. Finally, the driver receives visual, vestibular, and auditory cues corresponding his/her virtual motion, which aim to provide a realistic driving experience.

The goal of the MCA is to reproduce the specific forces, which are generated in the virtual environment, in the physical world while convincing the driver for a real driving experience. However, generating realistic motion cues is especially challenging due to the limited workspace of the motion system of the simulator. Even large simulators will not be able to reproduce the vehicle motion exactly ("one-to-one") as that would require a motion space that is as large as the space the simulated vehicle is moving in. Therefore, a MCA adjusts the accelerations from the driving vehicle model by considering the human perception of motion, and at the same time, it applies strategies for using the workspace efficiently. Since the MCA determines the physical motion of the simulator, in the literature, it often takes only the vestibular perception into account.

C. Optimal Control based Motion Cueing Algorithm

Optimal Control (OC) deals with finding the control strategy for a dynamical system based on minimizing a cost function under some system constraints [24]. In the context of motion cueing, the cost function defines the error between the virtual and the simulator's motion by using the state-space model of the simulator. Additionally, the simulator states and control signals can also be penalized in order to regulate the simulator's motion and workspace usage. The problem of OC-MCA for 6-DoF dynamic simulator can be written by means of following basic formulation:

$$\underset{\substack{x_1,x_2,\dots x_N\\ u_1,u_2,\dots u_N}}{\operatorname{argmin}} \quad \sum_{k=0}^{N} \|\boldsymbol{r}_k - \boldsymbol{r}_{ref,k}\|_{\boldsymbol{Q}} + \|\boldsymbol{x}_k\|_{\boldsymbol{R}} + \|\boldsymbol{u}_k\|_{\boldsymbol{P}} \tag{1a}$$

$$\text{subject to} \quad \boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k \tag{1b}$$

$$\mathbf{r}_k = \begin{bmatrix} F(\boldsymbol{x}_k, \boldsymbol{u}_k) & x_{\phi,k} & x_{\theta,k} & x_{\psi,k} \end{bmatrix}^T \tag{1c}$$

$$\boldsymbol{x}_{act,k} = G(\boldsymbol{x}_k) \tag{1d}$$

$$\boldsymbol{x}_{low} \leq \boldsymbol{x}_k \leq \boldsymbol{x}_{up} \tag{1e}$$

$$\boldsymbol{x}_{act,low} \leq \boldsymbol{x}_{act,k} \leq \boldsymbol{x}_{act,up} \tag{1f}$$

where $\boldsymbol{x}$ is an augmented state vector including the position and velocity of the moving platform for each DoF defined in the inertial coordinate system in the real world (see Fig. 1b). The control signal $\boldsymbol{u}$ contains the acceleration of the moving platform. The dynamic behaviour of the moving platform is described by the system matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ and is often implemented as an ideal simulator by using the double integrator model. The reference signal $\boldsymbol{r}_{ref}$ includes the specific forces, angular velocities and angular position of the simulated vehicle in the (vehicle) body-fixed coordinate system in the virtual world. Similarly, $\boldsymbol{r}$ describes the specific forces, angular velocities and angular

position of the simulator in body-fixed coordinate system (driver's coordinate frame) in the real world. The function $F(\cdot)$ applies necessary coordinate system transformations in order to obtain the specific forces and angular velocities in driver's coordinate frame. The box constraints of simulator's workspace as well as the actuator limitations are denoted as $\boldsymbol{x}_{low,up}$ and $\boldsymbol{x}_{act,low,up}$. To calculate the actuators' length we used inverse kinematics of the simulator which is denoted by the function $G(\cdot)$. The operator $\|\boldsymbol{x}\|_{\boldsymbol{T}}$ computes $\boldsymbol{x}^T\boldsymbol{T}\boldsymbol{x}$ for given weighting matrix $\boldsymbol{T}$. For our application, we used the diagonal weighting matrices $\boldsymbol{Q}$, $\boldsymbol{R}$, and $\boldsymbol{P}$, and their weights are determined by domain experts.

IV. Data Collection

In order to train a neural network for imitating the OC-MCA, we created a training dataset by conducting data collection sessions with a static simulator (i.e., a simulator without motion base). For our driving scenario, we selected a model of a real route near Munich in Germany, which was also investigated in multiple other studies [25]–[27] (see Fig. 3). The route includes some sections, such as country roads, a roundabout and a combination of multiple left and right turns. These various features yielded a "rich" set of training data, which enabled the NN-MCA to learn the input-output mappings for a large range of motions.

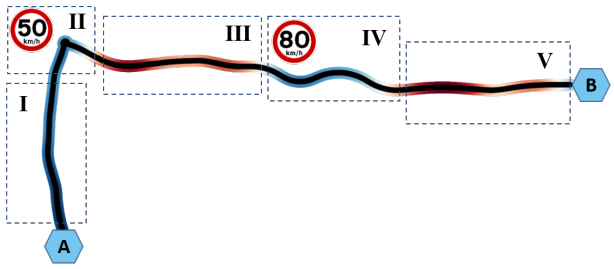


Fig. 3. The route that the participants are asked to follow. The driving direction is either A → B or B → A. The road height profiles are color-coded from blue to red, which corresponds to lowest and highest altitudes, respectively.

A total number of 25 participants drove the route four times, by starting twice at point A and twice at point B. By doing so, the variability in the training data was increased. The participants were asked to drive the given route while exhibiting natural driving behavior, i.e., paying attention to the traffic rules and other traffic participants like cars and trucks. The sections (I), (III) and (V) are two-lane country roads. The section (III) and (V) are relatively flat whereas the section (III) has several hills. Section (II) contains a small town including a roundabout. A city limit sign (implying a speed limit of 50 km/h) was placed before the participants entered the town. Section (IV) contains two sharp turns where the speed limit was $80\,\mathrm{km/h}$.

During the data collection sessions the motion of the simulated vehicle (dynamic vehicle model) was recorded. The specific forces, angular velocities and angular position of the simulated vehicle were calculated and stored in a dataset $\mathcal{X}^D$. Using OC-MCA, as described above, optimized motion cueing data was generated from the trajectories in $\mathcal{X}^D$ and the results were stored in a second data set $\mathcal{X}^S$. The $\mathcal{X}^S$ contains the acceleration commands $\boldsymbol{u_t}$ for time t, as well as the platform's position and velocity $\boldsymbol{x_t}$ for each DoF.

V. Neural Network based Motion Cueing Algorithm

According to the universal approximation theorem, a neural network with at least one hidden layer can approximate a large variety of functions $f(\cdot)$, *input-output-mappings*, by choosing appropriate parameters, such as activation functions, number of neurons in hidden layers and number of hidden layers [28], [29]. From a mathematical point of view the underlying function to approximate could be also a controller, which can be then achieved by training the network in a supervised manner on the dataset.

In context of motion cueing, the task is to find a mapping from the driver's motion in the virtual world to a control command for the simulator in the real world, which satisfies the driver's perception of motion. In order to achieve the best possible motion cueing, we trained a neural network to approximate the results obtained by the OC-MCA. In other words: given the driver's motion $\boldsymbol{x}_t^D \in \mathcal{X}^D$ in the virtual world, the trained model will generate similar control commands $\boldsymbol{u}_t^S \in \mathcal{X}^S$ as OC-MCA (in order to avoid any confusion, we use superscript ^{S} for the signals from $\mathcal{X}^S$). Since $\boldsymbol{x}_t^D$ does not provide any information about the current state of the simulator, we also provided the network with position and velocity states $\boldsymbol{x}_t^S$ as an input in order to obtain awareness for the workspace, which results in a model similar to an autoregressive network with the exogenous inputs [30].

In the literature, deep-learning-related robot control tasks are typically handled by learning to control at the level of joint torque or joint velocities [31], [32]. However, the controller of the simulator requires position, velocity, and acceleration commands, which are then internally fused to generate actuator commands. Depending on the type of MCA, this can be done in various ways. In classical MCAs, acceleration commands $\boldsymbol{u}_t^S$ are integrated in order to obtain next velocity and position $\boldsymbol{x}_{t+1}^S$, whereas in the case of the optimization-based MCAs proposed in the current paper, the optimization calculates all three signals directly while using the state-space model of the simulator. Therefore, the neural networks can be trained to generate either only the target accelerations $\boldsymbol{u}_t^S$ which are then integrated, or all target signals ($\boldsymbol{x}_{t+1}^S$, $\boldsymbol{u}_t^S$) in an "end-to-end" fashion. Empirically, we observed that only generating accelerations results in poor performance in the test phase due to the error accumulation, and so the NN-MCA was configured to predict $\boldsymbol{u}_t^S$ and $\boldsymbol{x}_{t+1}^S$ of $\mathcal{X}^S$ in an "end-to-end" fashion.

However, since the trained NN-MCA also learns integration operations internally, an inconsistency between outputs may occur in this approach due to the noisy estimator characteristic of the neural networks. The inconsistency may give rise to unpredictable control behavior during the test phase with the real simulator. In order to reduce the inconsistency, an additional cost was introduced to the network for training.

The final cost function extended with the inconsistency cost is given as:

$$\text{cost} = \sum_i (\boldsymbol{x}^S_{t+1,i} - \hat{\boldsymbol{x}}^S_{t+1,i})^2 + \sum_i (\boldsymbol{u}^S_{t,i} - \hat{\boldsymbol{u}}^S_{t,i})^2 + \alpha \sum_i (\widetilde{\boldsymbol{x}}^S_{t+1,i} - \hat{\boldsymbol{x}}^S_{t+1,i})^2 \quad (2)$$

The first two terms of the cost are the regression loss (mean squared error), and the third term corresponds to the proposed inconsistency cost. During the training, an additional simulator state $\widetilde{\boldsymbol{x}}^S_{t+1}$ is obtained from the generated acceleration $\hat{\boldsymbol{u}}^S_t$ and current state of the simulator $\boldsymbol{x}^S_t$ by using the state-space model. The new state $\widetilde{\boldsymbol{x}}^S_{t+1}$ corresponds to the next simulator state which would be achieved when only using generated accelerations $\hat{\boldsymbol{u}}^S_t$. The squared error between $\widetilde{\boldsymbol{x}}^S_t$ and $\hat{\boldsymbol{x}}^S_t$ is similar to an additional cost on predicted terms corresponding to the position and velocity from the dataset. Additionally, the introduced cost penalizes the inconsistency between the predicted outputs, rather than imposing a penalty according to the dataset. The term α regulates the weight of the inconsistency cost.

In order to obtain generalizable neural networks, a large data set is required for training. Obtaining such a data set through human-in-the-loop recording sessions in the simulator is costly. Instead, in order to obtain good generalization, a reasonable amount of random noise may be injected into the inputs of the network [33]. In fact, noise injection can also be interpreted as a form of data augmentation where the dataset extends artificially. In the test phase, even though small perturbations may exist in the simulator's current state, the NN-MCA should predict a similar control signal according to the $\boldsymbol{x}^D_t$. Thus, Gaussian noise is introduced to the inputs of the network, which corresponds to $\boldsymbol{x}^S_t$. Additionally, injecting noise on $\boldsymbol{x}^S_t$ practically results in the neural network giving more weight on learning the relationship between driver's motion and next simulator state rather than learning a sequence of consecutive simulator states. Fig. 4 illustrates the resulting NN-MCA with the inconsistency cost (2), wherein the double integrator is used for the state-space model as in (1).

We divided the collected dataset randomly into training and test sets with a ratio of 90/10. The NN-MCA is only trained with the training set. The test set is unknown to the neural network during training and is only used to evaluate the final performance of the NN-MCA in our experiments. For the training, we used a mini-batch size of 512, and the ADAM optimizer with the default recommended parameters provided in [34]. The maximum number of training epochs was set to 200, and the validation score was monitored to save the weights of the neural network each time the validation score increases. In general, we observed that the validation score stopped increasing after 150 epochs. For hyperparameter search, we used the grid-search method. The final best performing model is a fully connected network with two hidden layers with 64 neurons and hyperbolic tangent activation function. The output layer has the linear activation. We observed the best performance with Gaussian noise of $\mathcal{N}(0, 2.0)$ and inconsistency weight of $\alpha = 5.0$. We also observed that increasing inconsistency weight increased stability by up to a factor of 10. Additionally, for better generalization performance, we also experimented with dropout regularization [35]. However, empirically we observed no benefit of using dropout. Since both dropout and noise injection are regularization methods against overfitting, increasing the noise injection reduces the need for dropout.

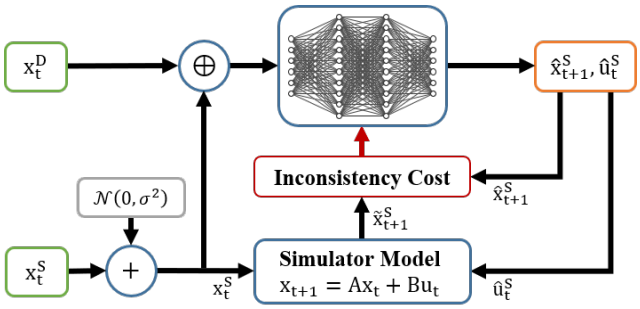


Fig. 4. Illustration of proposed NN-MCA during training. In the test phase, only the neural network part is used to generate (predicted) control commands $(\hat{\boldsymbol{x}}^S_{t+1}, \hat{\boldsymbol{u}}^S_t)$ for given driver's motion $\boldsymbol{x}^D_t$ in the virtual world and simulator's current state $\boldsymbol{x}^S_t$ in the real world. The operator $\oplus$ corresponds to the vector concatenation of both inputs.

VI. Experiments

Before testing the NN-MCA on the real driving simulator, we first evaluated the performance of our method on an emulator (i.e., a software model) of the simulator. For this purpose, we used eMoveRT Controller Software from E2M Technologies to emulate a model of the 6-DoF hexapod motion system (see Fig. 1a). The emulator includes both the forward and inverse kinematic models of the simulator and has nearly identical performance and control logic as the real simulator. In the test phase, the NN-MCA receives the motion data of the simulated vehicle $\boldsymbol{x}^D_t$ from the dataset and simulator's state $\boldsymbol{x}^S_t$ from the simulator (or emulator). The NN-MCA predicts the control command $(\hat{\boldsymbol{x}}^S_{t+1}, \hat{\boldsymbol{u}}^S_t)$, and sends these to the simulator (or emulator), and receives the next $\boldsymbol{x}^D_{t+1}$ and $\boldsymbol{x}^S_{t+1}$.

In order to analyse the results from human perception point of view, we also calculated the specific forces and angular velocities of the simulator (or emulator) during the simulation. Since the aim is to generate a simulator motion as close as possible to the motion of the virtual vehicle, we calculated the Root Mean Squared Error RMSE between the specific forces and angular velocities of the simulated vehicle and the simulator (or emulator). For this metric, a lower value represents better cueing quality. The RMSE is preferred over other metrics, since the resulting error has the same units as the perception thresholds. Additionally, we applied the same evaluation methods for two additional MCAs: first, a Classical Washout-based MCA (CW-MCA) which is tuned for this route and, second, the OC-MCA. This allowed us to compare the performance of the NN-MCA with two relevant

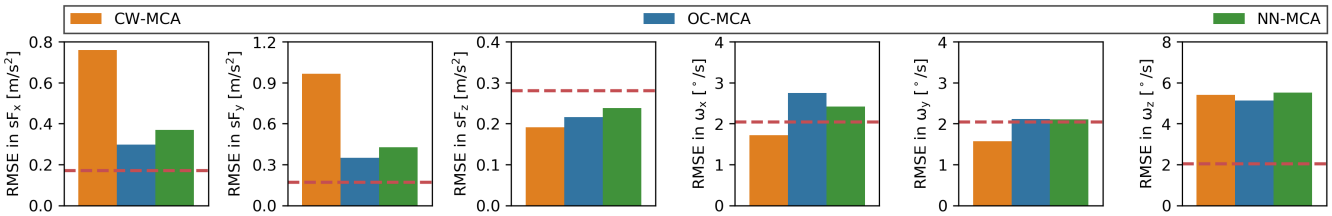


Fig. 5. The root mean squared error of the Classical Washout (CW), Optimal Control (OC), and proposed Neural Network (NN) based MCA in the test set for each DoF. The perception thresholds for each DoF are shown with red dashed lines.

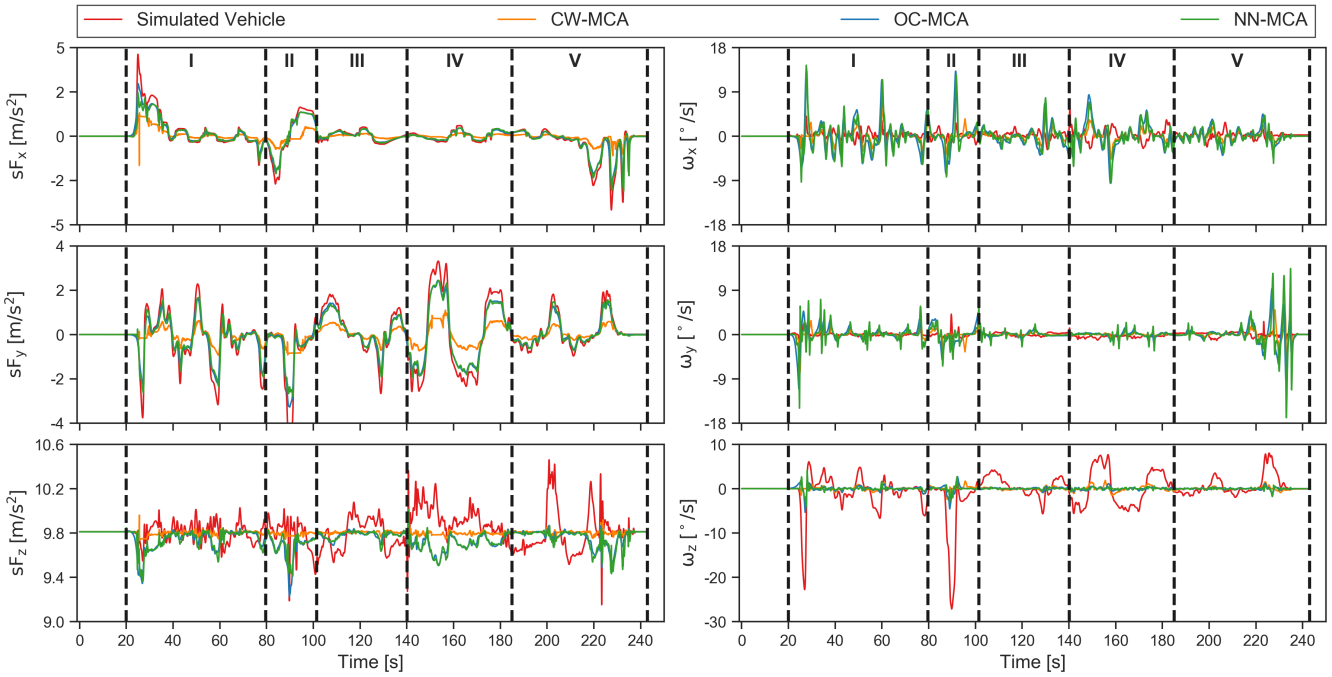


Fig. 6. The specific forces $\boldsymbol{sF}$ and angular velocities $\boldsymbol{\omega}$ of the simulated vehicle and the ones achieved by the Classical Washout (CW), Optimal Control (OC), and proposed Neural Network (NN) based MCA for an arbitrary trial from the test dataset

benchmarks: a commonly used MCA and a reference MCA which exhibits close-to-optimal cueing behavior (the "best-case-scenario").

Fig. 5 shows the results of the CW-MCA, OC-MCA and NN-MCA for the entire test data. As one can see, the NN-MCA has similar performance as the OC-MCA for each DoF and outperforms the CW-MCA in $\boldsymbol{sF}_{x,y}$ with up to 4x better performance. For $\boldsymbol{sF}_z$, $\boldsymbol{\omega}_y$ and $\boldsymbol{\omega}_z$ the CW-MCA shows a (slightly) lower RMSE (i.e., better performance) than the other MCAs. This can be explained by the fact that both OC-MCA and NN-MCA use these DoFs to produce or correct for cueing effects such as tilt-coordination to illicit illusions of sustained accelerations (see Sec. III-A). Note that the RMSE of both NN-MCA and OC-MCA are close to or below the perception threshold showing that the cueing errors in these DoFs are not or barely perceptible.

In order to highlight the performance of the NN-MCA, the results for an arbitrary trial from the test dataset is displayed in Fig. 6, where the participant drove the route A→B (see Fig. 3). Compared to the CW-MCA, the NN-MCA can generate more accurate specific forces with high absolute amplitudes for $\boldsymbol{sF}_{x,y}$. The CW-MCA can realize only small amplitudes due to the linear filtering and scaling , which need to be tuned such that the strongest motion still fits inside the simulator's motion space. As a result, the CW-MCA cueing has a typical "flat" characteristic. These effects can especially be observed for longitudinal dynamics in sections (I) and (V) and for lateral dynamics in sections (II) and (IV). Similar to OC-MCA, the NN-MCA applies relatively high rotational rates $\boldsymbol{\omega}_{x,y}$ to accomplish generating demanded $\boldsymbol{sF}_{y,x}$ with high absolute amplitudes through tilt-coordination. In order to have more realistic specific forces in x and y directions, the OC-MCA and NN-MCA regularly generate angular velocities which have higher absolute amplitudes than the vehicle model. This results in relatively high (poor) RMSE values in these DoFs, but they also result in additional sustained specific forces in x- and y-direction. And, even though the rates are higher, they are still below or around the perceptual threshold. Moreover, the $\boldsymbol{sF}_z$ direction includes noise such as road irregularities; therefore it is harder to achieve good performance for this DoF. Since

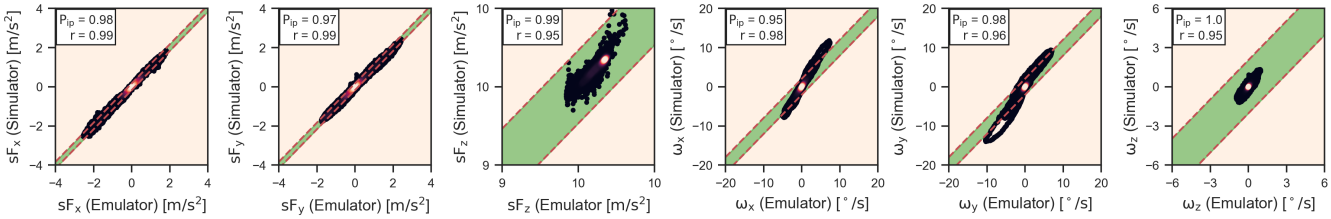


Fig. 7. The Pearson correlation r between the specific forces $\boldsymbol{sF}$ (top) and angular velocities $\boldsymbol{\omega}$ (bottom) while testing with emulator and simulator using NN-MCA. The human perception limits are shown with red dashed lines, and the imperceptible region is colored as green, where the deviation between two tests is imperceptible. The percentage of the data points lie in this region and are denoted with $\mathrm{P_{ip}}$. Sample colors indicate the density of the region, whereas yellow corresponds to high density.

the specific forces in longitudinal and lateral plane are of more interest for motion feeling, tuning the NN-MCA for z-direction is deferred to future work.

In addition to the comparison with CW-MCA, it is important to note that the behavior and performance of the NN-MCA are close to that of the OC-MCA. The NN-MCA can mimic the control behaviour of the OC-MCA through the entire trajectory for each DoF with limited complexity and real-time capability. Even though the NN-MCA is not trained for the selected trajectory, our proposed method achieved a significantly good approximation of the OC-MCA and showed good generalization capabilities. On the contrary to the OC-MCA, the trained NN-MCA does not require the future vehicle states and high computational effort. During our tests with NN-MCA, the complete control cycle runs approximately in 100 Hz on Intel® Core™ i5-8365U processor, which satisfies the required 100 Hz operational speed of the motion system for the real-time control.

In addition to the analysis described above, the NN-MCA results were also tested on the real simulator (Fig. 1) by using the same offline data for the simulated vehicle. During the real simulator tests, the NN-MCA demonstrated a real-time and significantly close performance to OC-MCA. Indeed, we observed no major differences compared to the evaluations on the emulator. For a performance comparison of the CW-MCA, OC-MCA and our proposed method (NN-MCA), a demonstration video of an arbitrary trajectory from the test data is available at http://go.tum.de/708350.

In order to assess the difference between the emulator and simulator, we also evaluated the correlation of the resulting $\boldsymbol{sF}$ and $\boldsymbol{\omega}$ using the emulator and simulator with respect to human perceptual thresholds. Fig. 7 shows the correlation and joint distribution of the emulator and simulator experiments. The perceptual thresholds from Sec. III-A for each DoF are displayed with a dashed red line. Furthermore, perceptible and imperceptible stimuli regions are colored red and green, respectively. For the data points lying between thresholds, the difference between the emulator and simulator is not noticeable by the vestibular perception system. The results show that $97\% - 99\%$ of the specific forces generated by NN-MCA lies between perception limits, which means, the performance difference on emulator and simulator for these specific forces are not perceivable by the driver. Similarly, the imperceivable difference for the the angular velocities is $95\% - 100\%$. The performance difference between the emulator and simulator can be explained by both the model error of the emulator and the inconsistency of the NN-MCA. Even though NN-MCA is penalized with an inconsistency cost during training, it may still generate signals which are not consistent. However, this does not affect the final performance of the NN-MCA significantly.

VII. Conclusion and Future Work

We presented a novel Motion Cueing Algorithm (MCA), using a neural network to imitate Optimal-Control-based MCA (OC-MCA), which provides optimal cueing, but cannot run with a driver-in-the-loop due to the offline nature of the algorithm. After training, neural network realizes an approximated OC-MCA which can run with a driver-in-the-loop. We also applied a Gaussian noise injection, as a data augmentation technique, to achieve robust control even for small perturbations of the simulator in the test phase. Furthermore, we trained our proposed model in an "end-to-end" fashion, where the neural network learns not only the control signals, but the full next state of the simulator. The end-to-end method provided more stable results on a dataset with a limited sample size as that was used in this work. We also introduced a second cost term for neural network training to reduce the inconsistencies between the neural network's outputs, which may be caused by learning the full next state of the simulator. Finally, through analysis of the model of the simulator (emulator) and real simulator, we showed that, the resulting Neural Network-based MCA (NN-MCA) outperforms a commonly used and well tuned Classical Washout-based MCA (CW-MCA), both in overall performance, as in realizing complex driving maneuvers, which our experiments contain. Through the entire trajectory, NN-MCA achieved relatively close performance to the OC-MCA. The NN-MCA also demonstrated significant performance and generalization capabilities for the trajectories, which the network was not trained with. That makes our method not only superior to a commonly used technique (CW-MCA), but also a real-time representation of an OC-MCA, which provides the optimal cueing but cannot work in real-time with driver-in-the-loop.

Even though the NN-MCA approach proposed here requires a data set obtained from human drivers, the results shown here indicate that this data set can be obtained in a static simulator. Since operating a static simulator is typically

less costly than a dynamic simulator, the effort required to obtain this dataset remains within reasonable boundaries. Furthermore, since the optimal motion-cues are generated offline, the cost function of OC-MCA can be tuned offline with respect to the desired performance, which is less time consuming compared to the tuning of an online MCA. Theoretically, the performance of NN-MCA is limited by the performance of OC-MCA, which can vary based on tuning parameters. As a future work, we recommend evaluating performance of the NN-MCA using OC-MCAs with different settings.

Further future work includes the generation of a large training dataset with more variability to evaluate whether that improves performance and generalizability. Also, the data set could be generated by autonomous agents in the virtual environment, which can perform driving maneuvers similar to the human driver. Furthermore, generative models such as Generative Adversarial Networks [36] might be used to generate realistic artificial data. Finally, future studies could explore model-free Reinforcement Learning (RL) techniques. The RL might provide the pre-trained NN-MCAs with much higher and stable performance by learning through exploration.

REFERENCES

[1] P. Onesti, "Verbindung von Human- und Hardware-in-the-Loop-Testing verkürzt Entwicklungszeit [Combining human and hardware-in-the-loop testing shortens development time (translation by the author)]," *ATZ - Automobiltechnische Zeitschrift*, vol. 111, no. 10, pp. 764–769, 10 2009.

[2] K.-H. Chang, "Chapter 8 - Motion Analysis," in *e-Design*, K.-H. Chang, Ed. Boston: Academic Press, 2015, pp. 391 – 462.

[3] W. Jianqiang, L. Shengbo, H. Xiaoyu, and L. Keqiang, "Driving simulation platform applied to develop driving assistance systems," *IET intelligent transport systems*, vol. 4, no. 2, pp. 121–127, 2010.

[4] M. Alonso, M. H. Vega, and O. Martín, "Driving simulation study for the analysis of distraction effects in longitudinal driving behaviour," *Cognition, Technology & Work*, vol. 14, no. 4, pp. 283–297, 11 2012.

[5] B. Conrad and S. Schmidt, *Motion drive signals for piloted flight simulators*. National Aeronautics and Space Administration, 1970, vol. 1601.

[6] P. Grant, B. Artz, M. Blommer, L. Cathey, and J. Greenberg, "A paired comparison study of simulator motion drive algorithms," *DSC Europe 02 Proceedings, Paris*, 2002.

[7] M. Fischer, H. Sehammer, and G. Palmkvist, "Motion cueing for 3-, 6-and 8-degrees-of-freedom motion systems," *Actes INRETS*, pp. 121–134, 2010.

[8] M. Dagdelen, G. Reymond, A. Kemeny, M. Bordier, and N. Maïki, "MPC based motion cueing algorithm: Development and application to the ULTIMATE driving simulator," in *Conférence simulation de conduite*, 2004, pp. 221–233.

[9] Cruden B.V., "The company's webplatform," https://www.cruden.com/, [Online; accessed 02.11.2019].

[10] A. Mohammadi, H. Asadi, S. Mohamed, K. Nelson, and S. Nahavandi, "Future reference prediction in model predictive control based driving simulators," 12 2016.

[11] C. Rengifo, J.-R. Chardonnet, D. Paillot, H. Mohellebi, and A. Kemeny, "Solving the constrained problem in model predictive control based motion cueing algorithm with a neural network approach," 2018.

[12] J. Drgoňa, D. Picard, M. Kvasnica, and L. Helsen, "Approximate model predictive building control via machine learning," *Applied Energy*, vol. 218, pp. 199 – 216, 2018.

[13] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots, "Learning deep neural network control policies for agile off-road autonomous driving," 2017.

[14] B. Goldfain, P. Drews, C. You, M. Barulic, O. Velev, P. Tsiotras, and J. M. Rehg, "AutoRally An open platform for aggressive autonomous driving," 2018.

[15] S. Levine, "Lecture notes in deep reinforcement learning CS 285," September 2019.

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

[17] M. Fischer, "Motion-cueing algorithms for a realistic motion feedback in moving-base simulators (German)," Ph.D. dissertation, Braunschweig, 5 2009.

[18] L. Reid and M. Nahon, "Flight simulation motion-base drive algorithms: Part 1. developing and testing equations," University of Toronto, Tech. Rep., 1985.

[19] D. E. Angelaki and T. A. Yakusheva, "How vestibular neurons solve the tilt/translation ambiguity: comparison of brainstem, cerebellum, and thalamus," *Annals of the New York Academy of Sciences*, vol. 1164, p. 19, 2009.

[20] F. Colombet, Z. Fang, and A. Kemeny, "Tilt thresholds for acceleration rendering in driving simulation," *SIMULATION*, vol. 93, 11 2016.

[21] T. Chapron and J.-P. Colinot, "The new PSA Peugeot-Citroen advanced driving simulator overall design and Motion Cue Algorithm," in *Proceedings of Driving Simulation Conference*, vol. 42, 2007, p. 173.

[22] C. Weiß, "Control of a dynamic driving simulator: Time-variant motion cueing algorithms and prepositioning," Ph.D. dissertation, Universität Stuttgart, DLR, 2006.

[23] M. A. Nahon and L. D. Reid, "Simulator motion-drive algorithms-a designer's perspective," *Journal of Guidance, Control, and Dynamics*, vol. 13, no. 2, pp. 356–362, 1990.

[24] A. E. Bryson, "Optimal control-1950 to 1985," *IEEE Control Systems Magazine*, vol. 16, no. 3, pp. 26–33, June 1996.

[25] A. Parduzi, J. Venrooij, M. Peller, Y. Forster, and S. Marker, "Evaluation of a 3-dof mid-size simulator concept: A behavioural validation study," 09 2019.

[26] S. Agabekov, "Implementierung und Evaluierung eines hybriden Motion Cueing Algorithmus [Implementation and evaluation of a hybrid motion cueing algorithm (translation by the author)]," Master's Thesis, Technische Universität München, 2019.

[27] M. Spannagl, "Hybrider Motion-Cueing-Algorithmus für einen 9-DOF Fahrsimulator [Hybrid motion cueing algorithm for a 9-DOF driving simulator (translation by the author)]," Master's Thesis, Technische Universität München, 2018.

[28] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural networks*, vol. 6, no. 6, pp. 861–867, 1993.

[29] B. C. Csáji, "Approximation with artificial neural networks," *Faculty of Sciences, Etvs Lornd University, Hungary*, vol. 24, p. 48, 2001.

[30] X. Ren, A. B. Rad, P. Chan, and W. L. Lo, "Identification and control of continuous-time nonlinear systems via dynamic neural networks," *IEEE Transactions on Industrial Electronics*, vol. 50, no. 3, pp. 478–486, 2003.

[31] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[32] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3389–3396.

[33] C. M. Bishop, "Training with noise is equivalent to tikhonov regularization," *Neural Computation*, vol. 7, no. 1, pp. 108–116, 1995.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[35] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[36] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.