vehicles

MDPI

Article

MPC-Based Motion-Cueing Algorithm for a 6-DOF Driving Simulator with Actuator Constraints

Yash Raj Khusro [1], Yanggu Zheng [2], Marco Grottoli [2,3] and Barys Shyrokau [2,*]

1 Rimac Automobili d.o.o., 10431 Sveta Nedelja, Croatia; yash.khusro@rimac-automobili.com
2 Department of Cognitive Robotics, Delft University of Technology, Mekelweg 2, 2628CD Delft, The Netherlands; y.zheng-2@tudelft.nl (Y.Z.); m.grottoli@tudelft.nl (M.G.)
3 Siemens Digital Industries Software nv, Interleuvenlaan 68, B-3001 Leuven, Belgium; marco.grottoli@siemens.com
* Correspondence: b.shyrokau@tudelft.nl

Received: 18 November 2020; Accepted: 27 November 2020; Published: 2 December 2020

Abstract: Driving simulators are widely used for understanding human–machine interaction, driver behavior and in driver training. The effectiveness of simulators in this process depends largely on their ability to generate realistic motion cues. Though the conventional filter-based motion-cueing strategies have provided reasonable results, these methods suffer from poor workspace management. To address this issue, linear MPC-based strategies have been applied in the past. However, since the kinematics of the motion platform itself is nonlinear and the required motion varies with the driving conditions, this approach tends to produce sub-optimal results. This paper presents a nonlinear MPC-based algorithm which incorporates the nonlinear kinematics of the Stewart platform within the MPC algorithm in order to increase the cueing fidelity and use maximum workspace. Furthermore, adaptive weights-based tuning is used to smooth the movement of the platform towards its physical limits. Full-track simulations were carried out and performance indicators were defined to objectively compare the response of the proposed algorithm with classical washout filter and linear MPC-based algorithms. The results indicate a better reference tracking with lower root mean square error and higher shape correlation for the proposed algorithm. Lastly, the effect of the adaptive weights-based tuning was also observed in the form of smoother actuator movements and better workspace use.

Keywords: driving simulator; motion-cueing algorithm; model predictive control; nonlinear; actuator constraints

1. Introduction

With the increasing demand for advanced driver assistance systems and automated driving vehicles, driving simulators hold the potential to transform the research and development of intelligent vehicles. They can reduce the cost and time incurred in the vehicle development process and help in designing robust and intelligent solutions. Furthermore, these simulators are increasingly being used for other purposes [1,2], such as human–machine interface studies, understanding driver behavior, and training of drivers in a safe controllable environment.

The effectiveness of such driving simulators is measured by their ability to generate realistic motion cues i.e., the driver or the passenger, sitting inside the simulator should perceived similar motion cues that s/he would perceived while sitting in a real vehicle performing the same maneuver. However due to its limited workspace, the driving simulator cannot be directly subjected to the vehicle motion as then the platform would quickly reach its physical limits and no motion cues could be provided to the driver any further. To overcome this limitation, motion-cueing algorithms have been

developed. A Motion-Cueing Algorithm (MCA) is the strategy that governs the process of producing motion cues while keeping the motion platform within its physical limits. Thus, the main objectives of an MCA are the following:

- Providing realistic motion cues to the driver or passenger sitting inside the simulator.
- Keeping the motion platform within its physical boundaries.

The classical approach of designing an MCA is done by using the classical washout filters. The algorithm is a combination of multiple linear filters (as shown in Figure 1). The translational accelerations are first high-pass filtered to extract fast dynamics, e.g., the change of acceleration during transitions. The resulting signal is double-integrated to determine the translational displacements of the platform. The slow dynamics, e.g., sustained accelerations, are extracted by filtering the translational accelerations using a low-pass filter in parallel to the high-pass filter. The resulting signals are reproduced by tilting the platform to exploit the gravitational acceleration (Tilt Coordination). The angular velocities are high-pass filtered and integrated to calculate the angular displacements. The purpose is to mainly generate motion cues during the transitions. Sustained yaw motion is eliminated from the signal as the motion platform has limited range of motion on this specific degree of freedom. The signals from the tilt channel and rotational channel are added to calculate the total angular displacements of the motion platform.

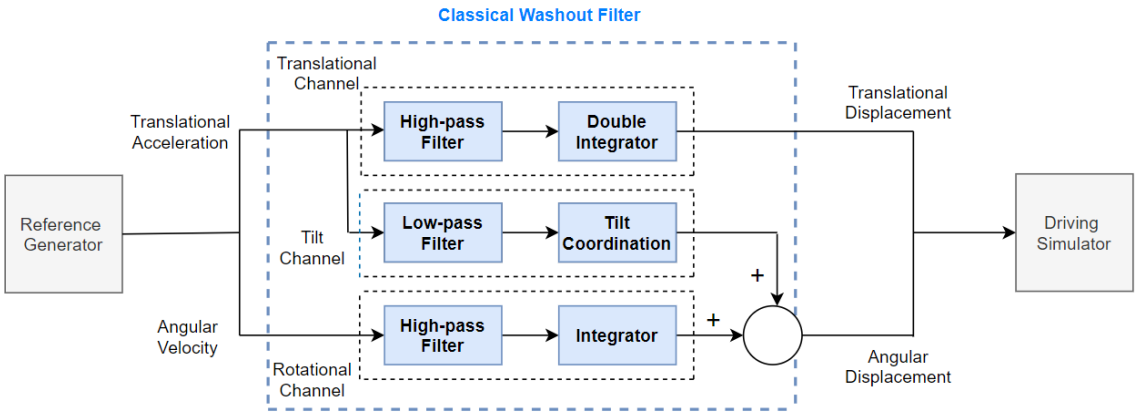

Figure 1. Scheme of classical washout filter-based MCA.

As per Nahon et al. [3], the major advantage of using such algorithm is that its design is simple and computationally efficient. However, these algorithms have the following shortcomings as well:

- Since the parameters of the filters are fixed, they must be designed for the worst-case maneuver. As a result, the algorithm does not use the available workspace for gentle maneuvers resulting in minimum motion.
- Tuning the filters is a complex task because the filter coefficients should be modified based on subjective participant feedback without taking into account meaningful physical quantities.
- Since there is no provision for incorporating the physical limits of the motion platform within the algorithm, the filters have to be tuned for each maneuver/participant to ensure that the motion platform remains within its physical limit.

To overcome these limitations, adaptive washout filter-based MCA is proposed. It tends to produce more realistic cues when the simulator is near the neutral position and only reduces the fidelity when the simulator is near its physical limits. The algorithm is based on minimizing a cost function comprising of penalties on the tracking error and the platform states. Generally, the optimization is performed by using the steepest descent method. The control scheme of a typical adaptive washout filter is shown in Figure 2. Furthermore, similar to the adaptive washout filter, another scheme named

the optimal washout filter is often used for motion-cueing applications [4]. The difference between this approach is that instead of using a gradient descent to minimize a cost, it uses the solution of the algebraic Riccati equation to derive the optimal gain. Although the adaptive and the optimal washout-based filters provide a better solution than the classical washout filters [3], the optimization problem is still solved without imposing any constraints on the physical states of the simulator, resulting in sub-optimal workspace use.

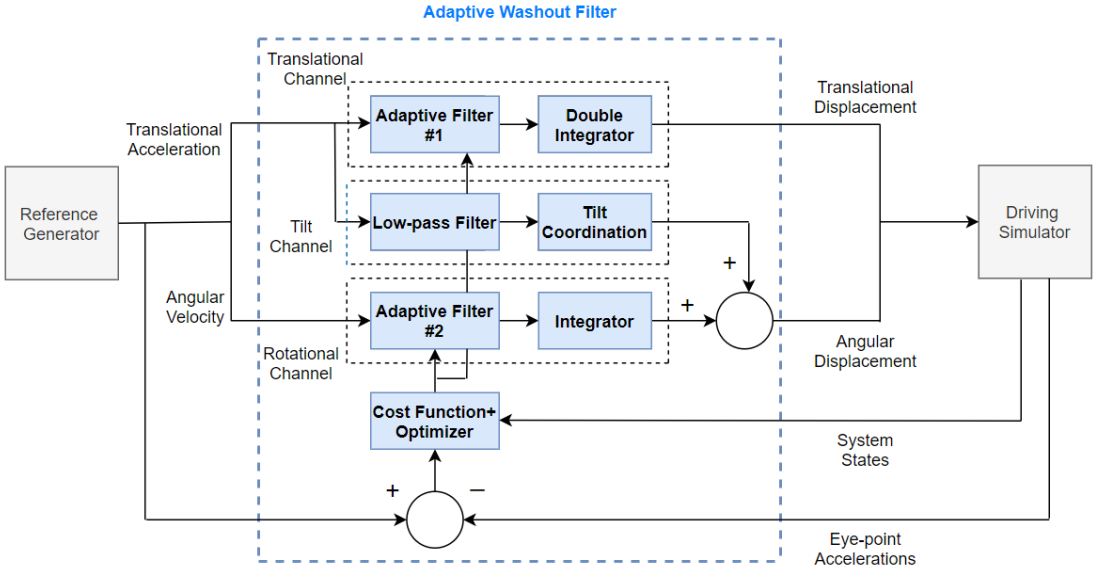

Figure 2. Scheme of adaptive washout filter-based MCA.

To address this issue, Model Predictive Control (MPC) technique has been recently applied to design an MCA. Its ability to handle constraints on system states and usage of the prediction model to regulate the current state makes it a well-suited alternative for this application. It has been shown that besides producing realistic motion cues, undesired effects like the occurrence of motion sickness are also lowered when using MPC-based MCA compared to the conventional filter-based approaches [5,6]. Recently, various approaches to MPC-based MCA have been explored and the superiority of this method compared to the conventional approach has been established for offline simulation and passive driving [7,8]. The scheme of a typical MPC-based MCA is shown in Figure 3.

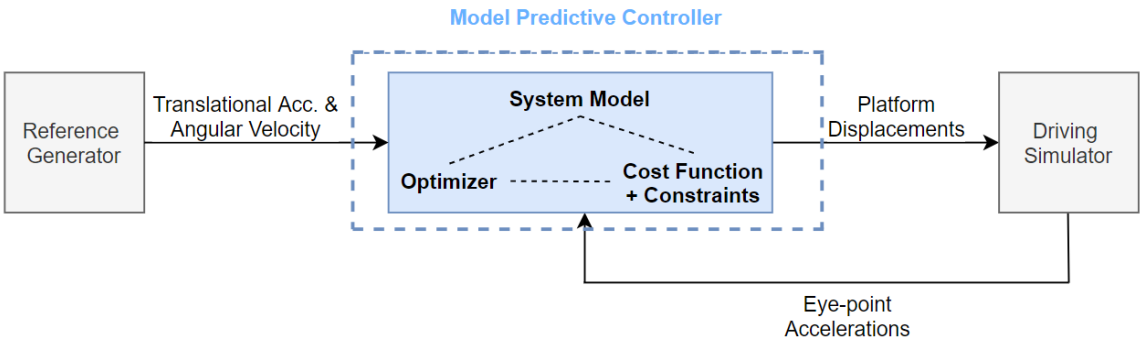

Figure 3. Scheme of MPC-based MCA.

To keep the problem linear, the algorithms designed in the past [9,10] apply the constraints on the position and velocity of the driver's eyepoint. In order to find the available workspace for the eyepoint displacement, the forward kinematic relations have to be used which is concerned with determining the displacement of the platform given the position of all the actuators. However for a six

DOF motion platform, there are many solutions to the forward kinematic problem [11] and only one of them corresponds to the actual pose of the platform. Generally, Newton-Raphson method is used to iteratively solve the forward kinematics problem. To reduce the computational effort, a conservatively chosen constant space is often used as the workspace for driver's eyepoint. However, this tends to produce sub-optimal results. An efficient alternative to manage the workspace is to use actuator-based constraints instead of the eyepoint displacements. Garret et al. [12] derived an MPC-based MCA which uses actuator-based constraints. However, linear approximations were applied to the constraints on the actuator lengths. This simplification also affected constraint handling as the inverse kinematics of the motion platform is nonlinear in nature. Degdelen et al. [13], implemented the MPC-based MCA in the Renault ULTIMATE Simulator. The study was done for a single DOF cueing problem (surge acceleration) and tilt coordination was demonstrated as an extension to the basic algorithm. Taking it further, in [14], an explicit MPC-based concept for the Renault ULTIMATE Simulator was proposed. The control problem was decoupled into four separate cases (pitch-surge, roll-sway, heave, and yaw) and a stability condition was determined. The algorithm operates in real time; however, fast degradation in the computational effort was observed as the problem extended to higher dimensions. Katliar et al. [15,16] implemented an MPC-based MCA which included the motion platform actuation for a Cable-Robot-based motion simulator. Their main finding was that with proper software and numerical methods, it is possible to run an MPC-based MCA with a complex model in real time.

In this paper, a new MPC-based MCA has been designed that incorporates the nonlinear kinematics of the Stewart platform. Inverse kinematics relations are used to calculate the length and the velocity of the actuators, which are included as states within the MPC framework. Moreover, the human vestibular system model is included within the MPC formulation to increase the fidelity of the produced motion cues. To manage the workspace efficiently, constraints are imposed on the actuator displacements and state-dependent adaptive weights are used to tune the MPC algorithm. The formulated nonlinear optimization problem is solved using the Real-Time Iteration (RTI) scheme [17] to increase the computational efficiency of the algorithm. Thus, the main contribution and a distinctive feature of the proposed approach is an efficient algorithm producing high-fidelity motion-cueing by using a nonlinear MPC-based controller with actuator-based constraints and state-dependent adaptive weights.

The rest of this paper is structured as follows. Section 2 describes the basics of Steward motion platform and the frames of references associated with it. The system model used within the MPC controller is derived in Section 3. Section 4 presents the details of the MPC formulation, including the objective function, constraints, reference generation, tuning and the optimization problem. In Section 5, several performance indicators are described. The investigated scenario, simulation setup, results and discussion are presented in Section 6. Finally, the conclusions and the recommendations for future work are presented in Section 7.

2. Motion Platform

The motion platform generally used in driving simulators is a Stewart platform, which is a parallel manipulator controlled by six linear actuators [18,19]. These actuators are attached in pairs at three positions to the triangular shaped top plate called the moving base and at three positions to the fixed base plate at the bottom. In the case of the driving simulator, the vehicle cockpit is attached on top of the moving base. Since the motion platform can have a motion in three translational directions (surge, sway and heave) and three rotational directions (roll, pitch and yaw), it can imitate the motion of a freely suspended body [20]. In this paper, the following three frames of reference with respect to the motion platform are used (shown in Figure 4).

1. Inertial Frame (IF)—is fixed to the ground and does not move with the motion platform. The origin coincides with the centroid of the fixed base of the platform (Point O in Figure 4). The positive x-axis points forward, in the direction of drive. The positive y-axis points to the right, while the positive z-axis points vertically downwards.

2. Platform Frame (PF)—is fixed to the motion platform and moves with it. The origin coincides with the centroid of the moving plate (Point P_0 in Figure 4). Similar to the IF, the positive x-axis points forward, the positive y-axis points to the right, while the positive z-axis points in the downwards direction. Since the PF is body-fixed, its axes are only aligned with that of the IF when the platform has a zero roll, pitch and yaw angle.
3. Driver Frame (DF)—is fixed to the driver's head and moves with it. The origin coincides with the eyepoint of the driver (Point D_0 in Figure 4). The positive x-axis points forward, the positive y-axis points to the right, while the positive z-axis points in the downwards direction. Since DF is fixed to the driver's eyepoint, its axes are only aligned with the IF when the platform has a zero roll, pitch and yaw angle.

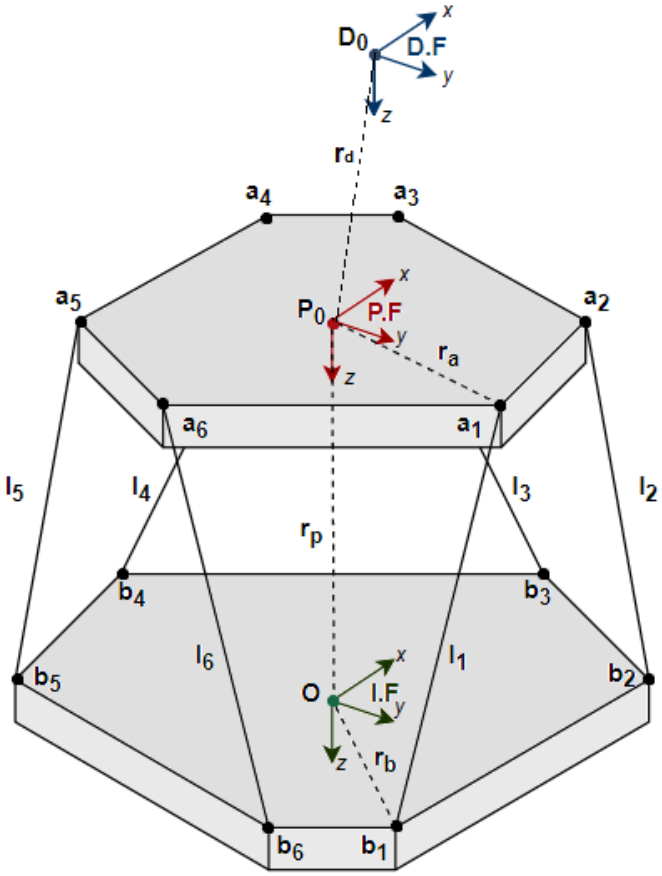

Figure 4. Motion Platform and reference frames used in the MCA.

It should be noted that throughout this paper, all the physical quantities are mentioned in IF unless specified by a superscript. Moreover, the translational acceleration vector is represented by the symbol a and consists of components in all the three canonical directions, i.e., $[a_x\ a_y\ a_z]^T$. Similarly, the translational velocity vector is represented by the symbol v and the translational displacement is represented by the symbol r. Furthermore, the angular acceleration vector is divided into rotational and tilt components. This is done to impose constraints on the tilt component without affecting the rotational component. The total angular acceleration is the sum of both the components and is represented by the symbol α. The vector consists of angular accelerations in three canonical directions, i.e., $[\alpha_\phi\ \alpha_\theta\ \alpha_\psi]^T$ (where ϕ, θ and ψ are the angles in pitch, roll and yaw directions respectively). Similarly, the angular velocity vector is represented by the symbol ω and the angular displacement is represented by the symbol β.

3. System Model

Model Predictive Control is a model-based optimal control strategy that computes the control input by solving an optimization problem. This is done to obtain the best possible reference tracking performance by predicting future states using the system model. The system model for the MPC algorithm is divided into two sub-parts.

1. Vestibular system model to provide realistic motion cues.
2. Motion platform model to manage the available motion workspace.

3.1. Vestibular System Model

The vestibular system located inside the human ear is primarily responsible to perceive motion in space. Within the vestibular system, there are two parts—the semi-circular canals, responsible for sensing the rotational accelerations and otolith organs, responsible for sensing the translations accelerations. Although extensive research had been conducted in the past for modeling the vestibular system mathematically (in [21–25]), reliable linear models have been derived only recently due to the fact that each human has slightly different perception and in general motion perception is not a linear process [26].

3.1.1. Semi-Circular Canals

Telban et al. [26] derived the linear transfer function of the semi-circular canal as follows:

$$\frac{\hat{\omega}_i(s)}{\omega_{i,\,rot}(s)} = 5.73\frac{80s^2}{(1+80s)(1+5.73s)} \tag{1}$$

where $\omega_{i,\,rot}$ is the angular velocity to which the passenger is subjected and $\hat{\omega}_i$ is the perceived angular velocity in one of the three rotational degrees of freedom. In this study, it has been assumed that the parameters of the model used are the same in all the three rotational degrees of freedom. This assumption is based on the physiological results based on the afferent responses of the semi-circular canals according to Telban et al. [26]. Representing Equation (1) in its observable canonical state-space form, leads to:

$$\begin{aligned} \dot{x}_{scc} &= A_{scc} \cdot x_{scc} + B_{scc} \cdot u_{scc} \\ y_{scc} &= C_{scc} \cdot x_{scc} + D_{scc} \cdot u_{scc} \end{aligned} \tag{2}$$

where $y_{scc} = \hat{\omega}_i$ and $u_{scc} = \omega_{i,\,rot}$ in one of the three canonical directions. Since this model has to be adopted for each rotational degree of freedom (roll ϕ, pitch θ and yaw ψ) individually, the complete model for semi-circular canals is given as follows:

$$\begin{aligned} \dot{x}_s &= A_s \cdot x_s + B_s \cdot u_s \\ y_s &= C_s \cdot x_s + D_s \cdot u_s \end{aligned} \tag{3}$$

where

$$\begin{aligned} A_s &= \begin{bmatrix} A_{scc_\phi} & 0_{2x2} & 0_{2x2} \\ 0_{2x2} & A_{scc_\theta} & 0_{2x2} \\ 0_{2x2} & 0_{2x2} & A_{scc_\psi} \end{bmatrix} \quad B_s = \begin{bmatrix} B_{scc_\phi} & 0_{2x1} & 0_{2x1} \\ 0_{2x1} & B_{scc_\theta} & 0_{2x1} \\ 0_{2x1} & 0_{2x1} & B_{scc_\psi} \end{bmatrix} \\ C_s &= \begin{bmatrix} C_{scc_\phi} & 0_{1x2} & 0_{1x2} \\ 0_{1x2} & C_{scc_\theta} & 0_{1x2} \\ 0_{1x2} & 0_{1x2} & C_{scc_\psi} \end{bmatrix} \quad D_s = \begin{bmatrix} D_{scc_\phi} & 0 & 0 \\ 0 & D_{scc_\theta} & 0 \\ 0 & 0 & D_{scc_\psi} \end{bmatrix} \end{aligned} \tag{4}$$

and input and output signals are shown in Equations (5) and (6). It must be noted that both quantities are expressed at the driver eyepoint (point D_0) in DF.

$$u_s = \omega^{DF}_{D_0,\,rot} = [\omega_{\phi,\,rot}\ \omega_{\theta,\,rot}\ \omega_{\psi,\,rot}]^T \tag{5}$$

$$y_s = \hat{\omega}^{DF}_{D_0} = [\hat{\omega}_\phi\ \hat{\omega}_\theta\ \hat{\omega}_\psi]^T \tag{6}$$

3.1.2. Otolith Organ

Based on the results from Telban et al. [26], the linear transfer function for the otolith organ is given as follows:

$$\frac{\hat{a}_i(s)}{a_i(s)} = 0.4\frac{(1+10s)}{(1+5s)(1+0.016s)} \tag{7}$$

where a_i is the specific acceleration to which the passenger is subjected and $\hat{a}_i$ is the perceived specific acceleration in one of the three translational degrees of freedom. Similar to the semi-circular canals, it has been assumed that the parameters of the otolith model used here are also same in all the three translational degrees of freedom. Moreover, both the input and output are to be specified at the driver's eyepoint (D_0) in DF. Representing Equation (7) in its observable canonical state-space form, results in:

$$\begin{aligned} \dot{x}_{oth} &= A_{oth} \cdot x_{oth} + B_{oth} \cdot u_{oth} \\ y_{oth} &= C_{oth} \cdot x_{oth} + D_{oth} \cdot u_{oth} \end{aligned} \tag{8}$$

where $y_{oth} = \hat{a}_i$ and $u_{oth} = a_i$, in one of the three canonical directions (surge x, sway y or heave z). Therefore, the complete model for otolith organs is as follows:

$$\begin{aligned} \dot{x}_o &= A_o \cdot x_o + B_o \cdot u_o \\ y_o &= C_o \cdot x_o + D_o \cdot u_o \end{aligned} \tag{9}$$

where

$$\begin{aligned} A_o &= \begin{bmatrix} A_{oth_x} & 0_{2x2} & 0_{2x2} \\ 0_{2x2} & A_{oth_y} & 0_{2x2} \\ 0_{2x2} & 0_{2x2} & A_{oth_z} \end{bmatrix} \quad B_o = \begin{bmatrix} B_{oth_x} & 0_{2x1} & 0_{2x1} \\ 0_{2x1} & B_{oth_y} & 0_{2x1} \\ 0_{2x1} & 0_{2x1} & B_{oth_z} \end{bmatrix} \\ C_o &= \begin{bmatrix} C_{oth_x} & 0_{1x2} & 0_{1x2} \\ 0_{1x2} & C_{oth_y} & 0_{1x2} \\ 0_{1x2} & 0_{1x2} & C_{oth_z} \end{bmatrix} \quad D_o = \begin{bmatrix} D_{oth_x} & 0 & 0 \\ 0 & D_{oth_y} & 0 \\ 0 & 0 & D_{oth_z} \end{bmatrix} \end{aligned} \tag{10}$$

and the input and the output signals are shown below as:

$$u_o = a^{DF}_{D_0} = [a_x\ a_y\ a_z]^T \tag{11}$$

$$y_o = \hat{a}^{DF}_{D_0} = [\hat{a}_x\ \hat{a}_y\ \hat{a}_z]^T \tag{12}$$

The complete model of otolith organ should also include the tilt coordination effects into it. The otolith matrix is augmented to the following:

$$\begin{aligned} A_{\bar{o}} &= \left[\begin{array}{c|c} A_o & \bar{B} \\ \hline 0 & 0 \end{array}\right] \quad B_{\bar{o}} = \left[\begin{array}{c|c} B_o & 0 \\ \hline 0 & I_3 \end{array}\right] \\ C_{\bar{o}} &= \left[\begin{array}{c|c} C_o & 0 \end{array}\right] \quad D_{\bar{o}} = \left[\begin{array}{c|c} D_o & 0 \end{array}\right] \end{aligned} \tag{13}$$

where

$$\bar{B} = B_o \cdot \begin{bmatrix} 0 & g & 0 \\ -g & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{14}$$

It must be noted that small-angle approximation is assumed and the input and the output signals expressed at the eyepoint of the driver (point D_0) in DF:

$$u_{\bar{o}} = [a_{D_0}^{DF} \; ; \; \omega_{D_0,\, tilt}^{DF}] \tag{15}$$

$$y_{\bar{o}} = \hat{a}_{D_0}^{DF} \tag{16}$$

3.1.3. Complete Vestibular System Model

The complete vestibular system is further modeled by combining the state-space models of the semi-circular canals and the otolith organ, resulting in the following system:

$$\begin{aligned} \dot{x}_v &= A_v x_v + B_v u_v \\ y_v &= C_v x_v + D_v u_v \end{aligned} \tag{17}$$

where

$$\begin{aligned} A_v &= \begin{bmatrix} A_s & 0_{6x9} \\ 0_{9x6} & A_{\bar{o}} \end{bmatrix} \quad B_v = \begin{bmatrix} 0_{6x6} & B_s \\ B_{\bar{o}} & 0_{9x3} \end{bmatrix} \\ C_v &= \begin{bmatrix} C_s & 0_{3x9} \\ 0_{3x6} & C_{\bar{o}} \end{bmatrix} \quad D_v = \begin{bmatrix} 0_{3x6} & D_s \\ D_{\bar{o}} & 0_{3x6} \end{bmatrix} \end{aligned} \tag{18}$$

and the input and the output signals expressed at the eyepoint of the driver (point D_0) in DF:

$$u_v = [a_{D_0}^{DF} \; ; \; \omega_{D_0,\, tilt}^{DF} \; ; \; \omega_{D_0,\, rot}^{DF}] \tag{19}$$

$$y_v = [\hat{a}_{D_0}^{DF} \; ; \; \hat{\omega}_{D_0}^{DF}] \tag{20}$$

Since the motion platform must be controlled in the IF, the inputs of the vestibular system must be converted from DF to IF. To transform the translational acceleration from DF to IF, the following relation is used:

$$a_{D_0}^{DF} = a_{P_0} + (\omega_{P_0} \times (\omega_{P_0} \times (R_{PF}^{IF} \cdot r_d^{PF}))) + (\alpha_{P_0} \times (R_{PF}^{IF} \cdot r_d^{PF})) \tag{21}$$

where r_d^{PF} is the vector from point P_0 to the driver's eyepoint (point D_0) in PF. a_{P_0}, ω_{P_0} and α_{P_0} are the translational acceleration, total angular velocity and total angular acceleration respectively at point P_0 in IF. Furthermore, R_{PF}^{IF} is the rotation matrix for translational acceleration from PF to IF written in terms of the total inclination angles (roll (ϕ), pitch (θ) and yaw (ψ)) of the motion platform as follows:

$$R_{PF}^{IF} = \begin{bmatrix} \cos\phi \cdot \cos\psi - \sin\phi \cdot \cos\theta \cdot \sin\psi & -\cos\phi \cdot \sin\psi - \sin\phi \cdot \cos\theta \cdot \cos\psi & \sin\phi \cdot \sin\theta \\ \sin\phi \cdot \cos\psi + \cos\phi \cdot \cos\theta \cdot \sin\psi & -\sin\phi \cdot \sin\psi + \cos\phi \cdot \cos\theta \cdot \cos\psi & -\cos\phi \cdot \sin\theta \\ \sin\theta \cdot \sin\psi & \sin\theta \cdot \cos\psi & \cos\theta \end{bmatrix} \tag{22}$$

Moreover, to transform the rotational velocity from DF to IF, the following relation is used:

$$\omega_{D_0}^{DF} = T_{DF}^{IF} \cdot \omega_{P_0} \tag{23}$$

where T_{PF}^{IF} is the rotation matrix for rotational velocity from DF to IF, written in terms of the total inclination angles (roll (ϕ), pitch (θ) and yaw (ψ)) of the motion platform as follows:

$$T_{DF}^{IF} = \begin{bmatrix} 0 & \cos\phi & \sin\phi \cdot \sin\theta \\ 0 & \sin\phi & -\cos\phi \cdot \sin\theta \\ 1 & 0 & \cos\theta \end{bmatrix} \tag{24}$$

3.2. Motion Platform Model

The motion platform model is used by the MPC algorithm to manage the workspace. The workspace of a 6-DOF motion platform is defined as a six-dimensional complex-shaped body where the system is free to move without violating its actuator limitations. The boundaries of the workspace are formed due to the excursion limitations of one or more actuators. The motion space of the platform is defined as space where the system is free to move in the future as per the current state of the system. It should be noted the movement in a single DOF requires contribution from all the linear actuators. As a consequence, the available motion space in one DOF depends on the excursions in other DOFs as well. The following approaches can be used to manage the workspace:

- **Limiting motion workspace** —Forward kinematics is used to calculate the motion space (as per the current actuator position) in terms of the translational and angular displacement of the point P_0. Furthermore, the constraints are applied based on the current motion state and the same should be updated at each time step. As a result, the resulting motion space is a 6-dimensional complex body.
- **Limiting actuator workspace**—Inverse kinematics is used to determine the motion space directly in terms of the actuator positions. Subsequently, fixed constraints are added based on the permissible actuator length.

Limiting the actuator workspace results in simpler relations, because the stroke limit on each actuator is independent of that of the other actuators while the degrees of freedom of the point P_0 are coupled with each other. For example, the available workspace for the surge motion of point P_0 would depend on the current state of the other degrees of freedom (sway, heave, roll, pitch and yaw). Therefore, the bound on the surge motion will be state-dependent and must be calculated at every time step. Meanwhile, the bound on each actuator will be constant (its excursion or retraction limit) and independent of the other actuators. Therefore, in this study, limiting the actuator workspace is used as the strategy for workspace management.

Actuator Kinematics

The inverse kinematic relations of the Stewart platform can be used to implement the constraints on the actuator length and velocity based on the actuator kinematic relations derived in [27]. According to Figure 4, the following relation for actuator length vector can be derived using vector arithmetic:

$$\vec{L}_i = \vec{r}_p + R_{PF}^{IF} \cdot \vec{r}_a^{PF} - \vec{r}_b \tag{25}$$

The actuator length can be formulated as:

$$l_i = \sqrt{\vec{L}_i \cdot \vec{L}_i} \tag{26}$$

Moreover, the unit vector along the length vector can be written as:

$$\vec{n}_i = \vec{L}_i / l_i \tag{27}$$

Furthermore, the actuator velocity vector can be computed by differentiating Equation (26).

$$\dot{l}_i = Q_1^{-1} \cdot Q_2^{-1} \cdot \begin{bmatrix} v_p \\ \alpha_{P_0} \end{bmatrix} \tag{28}$$

where

$$Q_1^{-1} = \begin{bmatrix} \vec{n}_1^T & ((R_{PF}^{IF} \cdot \vec{r}_{a_1}^{PF}) \times \vec{n}_1)^T \\ \vdots & \vdots \\ \vec{n}_6^T & ((R_{PF}^{IF} \cdot \vec{r}_{a_6}^{PF}) \times \vec{n}_6)^T \end{bmatrix} \quad Q_2^{-1} = \begin{bmatrix} I_{3\times3} & O_{3\times3} \\ O_{3\times3} & T \end{bmatrix} \tag{29}$$

3.3. Combined System Model

The combined system model consisting of both the vestibular system and the motion platform model. Therefore, the combined system states can be formulated as:

$$\dot{x}_c(t) = \begin{cases} \dot{\omega}_{rot,\,P_0} = \alpha_{rot,\,P_0} \\ \dot{\beta}_{rot,\,P_0} = \omega_{rot,\,P_0} \\ \dot{\omega}_{tilt,\,P_0} = \alpha_{tilt,\,P_0} \\ \dot{\beta}_{tilt,\,P_0} = \omega_{tilt,\,P_0} \\ \dot{v}_p = a_{P_0} \\ \dot{r}_p = v_p \\ \dot{x}_v = A_v x_v + B_v \cdot [a_{D_0}^{DF}\,;\, \omega_{D_0}^{DF}] \\ \dot{l}_i = Q_1^{-1} \cdot Q_2^{-1} \cdot [v_p\,;\, \alpha_{P_0}] \end{cases} \tag{30}$$

where

$$\begin{aligned} a_{D_0}^{DF} &= a_{P_0} + (\omega_{P_0} \times (\omega_{P_0} \times (R_{PF}^{IF} \cdot r_d^{PF}))) + (\alpha_{P_0} \times (R_{PF}^{IF} \cdot r_d^{PF})) \\ \omega_{D_0}^{DF} &= T \cdot \omega_{P_0} \\ \alpha_{P_0} &= \alpha_{rot,\,P_0} + \alpha_{tilt,\,P_0} \\ \omega_{P_0} &= \omega_{rot,\,P_0} + \omega_{tilt,\,P_0} \\ \beta_{P_0} &= \beta_{rot,\,P_0} + \beta_{tilt,\,P_0} \end{aligned} \tag{31}$$

Therefore, the state vector x_c is:

$$x_c = [\omega_{rot,\,P_0} \quad \beta_{rot,\,P_0} \quad \omega_{tilt,\,P_0} \quad \beta_{tilt,\,P_0} \quad v_p \quad r_p \quad x_v \quad l_i]^T \tag{32}$$

and the input vector u_c can be written as follows:

$$u_c = [a_{P_0} \quad \alpha_{tilt,\,P_0} \quad \alpha_{rot,\,P_0}]^T \tag{33}$$

This combined system can be represented as the following:

$$\dot{x}_c(t) = f\big(x_c(t), u_c(t)\big) \tag{34}$$

To discretize the system, direct multiple shooting technique is used [28]. The time horizon $[t_0, t_0 + T]$ (where $T = N_p \times Ts$) is divided into N_p sub-intervals $[t_k, t_{k+N_p}]$ and the state trajectory is computed on each sub-interval independently. Furthermore, matching constraints are added to ensure continuity of the optimal state trajectory on the whole horizon [29]. After discretization, the obtained system can be represented as follows:

$$x_c(k+1) = f_d\big(x_c(k), u_c(k)\big) \tag{35}$$

4. MPC Formulation

The MPC controller uses the system model and the current state of the system to predict the evolution of the future state over a finite prediction horizon (N_p). Using this, the optimal control action is derived over a control horizon (N_c). Then, only the first control input is applied to the real system and the same process is repeated for the next time step. Therefore, the MPC input can be regarded as a nonlinear state feedback control input, obtained online by repeatedly solving the optimization problem—minimizing an objective function while adhering to the system dynamics and fulfilling the given constraints at every time step.

4.1. Objective Function

The standard objective function for MPC consists of quadratic functions of both the tracking error and the control action along the prediction horizon. In this paper, the objective function is divided into two parts, namely the stage cost $(\ell(x_c(k), u_c(k)))$ and the terminal cost $(V(x_c(N_p)))$. The total objective function is given as:

$$J(x_c, u_c) = \sum_{k=0}^{N_p-1} \ell\big(x_c(k), u_c(k)\big) + V\big(x_c(N_p)\big) \tag{36}$$

The expression for stage cost is shown below:

$$\ell\big(x_c(k), u_c(k)\big) = \|x_{ref}(k) - x_c(k)\|_Q + \|u_{ref}(k) - u_c(k)\|_R \tag{37}$$

where Q and R are the positive semi-definite weight matrices for a penalty on tracking error and control input, respectively. The stage cost function is defined such that it satisfies the following conditions:

$$\begin{aligned} &\ell(0,0) = 0 \\ &\ell\big(x_c(k), u_c(k)\big) > 0,\ \forall x(k) \in \mathbb{X},\ x(k) \neq x_{ref}(k) \end{aligned} \tag{38}$$

The expression for the terminal cost is shown below:

$$V\big(x_c(N_p)\big) = \|x_{ref}(N_p) - x_c(N_p)\|_P \tag{39}$$

where P is the positive semi-definite weight matrix for a penalty on the tracking error at the terminal stage of the prediction horizon.

For finite prediction horizon problems, stability can be guaranteed by choosing a suitable terminal cost (V) and terminal attractive region Ω [30,31]. Even though the conditions for asymptotic stability are clearly defined, choosing V and Ω is still an open problem [32]. It is shown in [33] that stability can be guaranteed by the tuning the matrices Q, R, and P. Furthermore, a longer prediction horizon (N_p) would help the algorithm to achieve convergence at the cost of a more computational demanding problem.

4.2. Constraints

For motion-cueing, the following constraints are generally applied:

- Constraint on the tilt rate (ω).
- Constraints on the actuator positions (l_i).

The constraint on the tilt rate is to ensure that the tilt coordination effects are not perceived by the human-being. Therefore, the tilt rate should be limited to the threshold values for rotation. The tilt rate constraints were imposed based on the values derived in the research of Reid et. al. [34] and listed in Table 1.

Table 1. Threshold values for rotational velocities.

Degree of Freedom	Threshold Value
Roll ω_ϕ	3.0 deg/s
Pitch ω_θ	3.6 deg/s
Yaw ω_ψ	2.6 deg/s

$$\omega_{tilt,\,\min} \leq \omega_{tilt} \leq \omega_{tilt,\,\max} \tag{40}$$

Furthermore, the constraints on the actuator positions are to ensure that the platform remains within its physical limits. This is expressed as the maximum extension (l_{max}) and retraction (l_{min}) of the actuator allowed.

$$l_{min} \leq l_i \leq l_{max} \quad i = 1, \ldots, 6 \tag{41}$$

The set of states $x(k)$ satisfying the aforementioned constraints is denoted by $\mathbb{X}$. Therefore, the combined constraint equations are represented as:

$$x(k) \in \mathbb{X} \tag{42}$$

4.3. Reference Generation

The reference vector contains the following variables:

$$x_{ref} = [\omega_{rot,\,ref} \quad \beta_{rot,\,ref} \quad \omega_{tilt,\,ref} \quad \beta_{tilt,\,ref} \quad v_{p,\,ref} \quad r_{p,\,ref} \quad x_{v,\,ref} \quad l_{i,\,ref}]^T \tag{43}$$

To ensure that the platform returns to neutral position, the reference for ω_{rot}, β_{rot}, ω_{tilt}, β_{tilt}, v_p, r_p, l_i are set to zero for the entire prediction horizon. Furthermore, the reference for x_v contains the reference for $\hat{a}_{D_0}^{DF}$ and $\hat{\omega}_{D_0}^{DF}$. These are computed by translating the translational acceleration and angular velocities obtained from the vehicle model to the driver's eyepoint in DF and then passing them through the vestibular system model. Since the future reference is not available in advance, the current value of x_v is kept constant throughout the prediction horizon.

4.4. Adaptive Weight-Based Tuning

The weight matrices used in the above formulation are as follows:

$$Q = diag([\boldsymbol{W}_{\omega_{rot}}\ \boldsymbol{W}_{\beta_{rot}}\ \boldsymbol{W}_{\omega_{tilt}}\ \boldsymbol{W}_{\beta_{tilt}}\ \boldsymbol{W}_{v_p}\ \boldsymbol{W}_{r_p}\ \boldsymbol{W}_{x_v}\ \boldsymbol{W}_{l_i}]) \tag{44a}$$

$$R = diag([\boldsymbol{W}_{a_{P_0}}\ \boldsymbol{W}_{\alpha_{tilt}}\ \boldsymbol{W}_{\alpha_{rot}}]) \tag{44b}$$

$$P = diag([\boldsymbol{w}_{\omega_{rot}}\ \boldsymbol{w}_{\beta_{rot}}\ \boldsymbol{w}_{\omega_{tilt}}\ \boldsymbol{w}_{\beta_{tilt}}\ \boldsymbol{w}_{v_p}\ \boldsymbol{w}_{r_p}\ \boldsymbol{w}_{x_v}\ \boldsymbol{w}_{l_i}]) \tag{44c}$$

In the conventional MPC scheme, the weights of these tuning matrices are fixed in advance. This approach works well if the individual states are not dependent on each other or if the properties of the system do not change during the course of the simulation. However, since the motion-cueing algorithm needs to adapt to the changing workspace, an adaptive weight-based tuning approach has been implemented in this paper. The fundamental idea of this approach is to increase the weight on the position error ($\boldsymbol{W}_{r_p}$) and velocity error ($\boldsymbol{W}_{v_p}$) non-linearly as the motion platform reaches near the actuator limit.

The adaptive weights-based tuning results in two main advantages. First, for a constrained MPC problem with a short prediction horizon, the resulting output trajectories are often not smooth when the system states reach the limits of the workspace. This is because the penalty on the system states is constant irrespective of the available workspace. By varying the tuning weights based on the available workspace, a damping action is provided resulting in smooth movement of the platform near the physical limit.

Secondly, the motion-cueing algorithms tend to produce false or missing cues due to constant tendency to perform washout. By keeping the weights on v_p and r_p low and only increasing it when the platform is near its limits, the adaptive weights-based tuning would also help to reduce the production of false or missing cues. The following weight function is proposed in this study:

$$\boldsymbol{W}_{r_p} = \left(\sum_{i=1}^{6} \frac{1}{(1.1 \cdot l_{max})^2 - l_i^2} - a \right) / b \tag{45a}$$

$$\boldsymbol{W}_{v_p} = \left(\sum_{i=1}^{6} \frac{1}{(1.1 \cdot l_{max})^2 - l_i^2} - c \right) / d \tag{45b}$$

where a, b, c and d are fixed parameters. The effect of the aforementioned adaptive weight function can be seen in Figure 5. Parameters a and c determine the value of the function at point m in Figure 5, while parameters b and d determine the value of the function at point n_1 and n_2 in Figure 5.

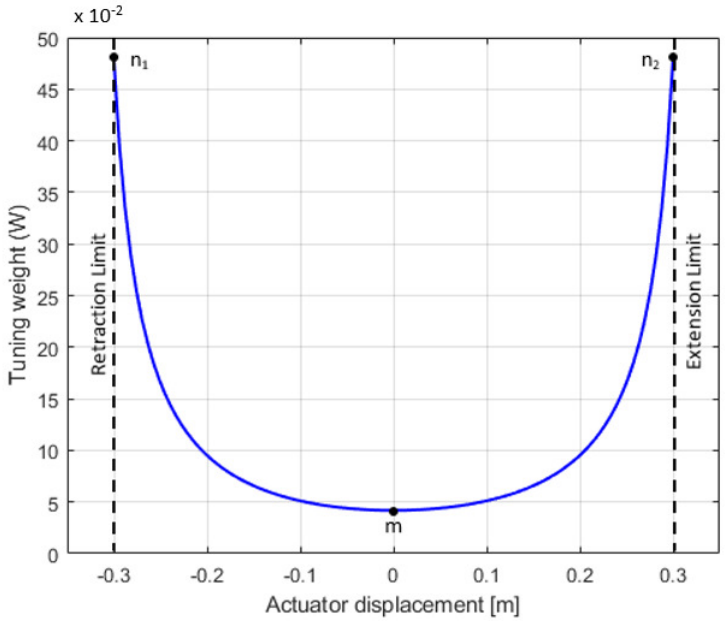

Figure 5. The effect of actuator displacement on the tuning weight.

It should be noted that the other tuning weights (except $\boldsymbol{W}_{r_p}$ and $\boldsymbol{W}_{v_p}$) in Equation (44) are kept constant and are shown in Table 2. Since the magnitude of the perceived angular velocity is much smaller than that of the perceived translational acceleration, a high weight for it is chosen (i.e., $W_{x_v,\hat{a}} \ll W_{x_v,\hat{\omega}}$). This scaling of weights is important so that the errors on both quantities are given equal weightage and the MPC algorithm seeks to track both quantities equally. Moreover, since the actuator displacements are already constrained and within the available workspace, free movement of the actuators is desired, a small weight is selected for the actuator displacement (W_{l_i}). Similarly, since the tilt rate is already constrained and free rotatory movement of the hexapod is desired, a low weight on the hexapod inclination angle ($W_{\beta_{rot}}$ and $W_{\beta_{tilt}}$) and inclination velocity ($W_{\omega_{rot}}$ and $W_{\omega_{tilt}}$) is chosen.

Table 2. Constant tuning parameters.

Parameter	$W_{\omega_{rot}}$	$W_{\beta_{rot}}$	$W_{\omega_{tilt}}$	$W_{\beta_{tilt}}$	$W_{y_v,\hat{a}}$	$W_{x_v,\hat{\omega}}$	W_{l_i}
Weight	0.1×10^{-2}	0.1×10^{-2}	0.1×10^{-2}	0.1×10^{-2}	2×10^{-2}	10×10^{-2}	0.1×10^{-2}

4.5. Optimization Problem

The optimal control input is defined by solving the following optimization problem:

$$\begin{aligned} & u(k) = argmin\ J\big(x_c(k), u_c(k)\big) \\ \text{s.t} \\ & x_c(k+1) = f_d\big(x_c(k), u_c(k)\big) \\ & x \in \mathbb{X} \end{aligned} \tag{46}$$

To solve the aforementioned optimization problem, ACADO toolkit [35] is used including a real-time iteration scheme to solve the nonlinear MPC problem. Multiple shooting technique is used to discretize the nonlinear continuous-time system. The objective function, which is arranged in the least-squares form, is solved using Sequential Quadratic Programming (SQP) technique. The RTI scheme uses the warm-start technique with shifting procedure to linearize the system, i.e., the solution of the optimization problem at the previous time step is shifted and used as the new linearization point. To reduce the number of optimization variables in the QP problem, a condensing procedure is used [28]. The resulting condensed QP problem is then passed to the qpOASES solver [36] using the active set method to evaluate the solution. In order to reduce the computational time and solve the optimization problem, the RTI method divides the optimization problem into two parts:

- Preparation step: The objective function is evaluated in the form of unknown state feedback x_0. The original QP problem is formulated and condensed into a smaller and denser QP.
- Feedback step: The state feedback x_0 is substituted and the QP is solved to obtain the control input.

The preparation step is performed at the previous time step. As soon as the state feedback x_0 is obtained, it is substituted in the QP and the solution is obtained. In this paper, a sampling time of 0.01 s was used with a prediction horizon $N_P = 50$ and the control horizon N_C equal to N_P.

5. Performance Indicators

To compare different motion-cueing algorithms, specific performance indicators should be specified. Furthermore, these indicators must be chosen to compare both the reference tracking performance and workspace use of the MCA.

5.1. Indicators for Reference Tracking Performance

Root Mean Square Error (RMSE) calculates for each time step is added and the result is normalized so that the indicator can compare short and long signals fairly. RMSE is presented in Equation (47) and has the range of $[0, +\infty]$.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(x_{ref,\,i} - x_i\right)^2} \tag{47}$$

Correlation Coefficient (CC) is the shape correlation between the reference and the actual signal given in Equation (48) with the range of $[0, +1]$. If the two signals are similar in shape, then the CC should be close to one, while it should be close to zero when there is low correlation [37]. This indicator can be particularly useful to signify if there are many missing or false cues.

$$CC(x_{ref}, x) = \frac{\sum_{i=1}^{n}\left(x_{ref,\,i} - \bar{x}_{ref}\right)\cdot\left(x_i - \bar{x}\right)}{\sqrt{\sum_{i=1}^{n}\left(x_{ref,\,i} - \bar{x}_{ref}\right)^2}\cdot\sqrt{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2}} \tag{48}$$

where

$$\bar{x}_{ref} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_{ref,\,i}$$
$$\bar{x} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i \tag{49}$$

Estimated Delay (ED) calculates the magnitude of delay between the reference and the actual signal. Since both the signals are not exactly equal, actual delay cannot be calculated. Therefore, it is estimated as the offset applied to the reference signal which maximizes the correlation coefficient. The range of ED indicator is $[0, +\infty]$ and an ideal signal with no delay with respect to the reference should have ED equal to zero.

5.2. Indicators for Workspace Use

Interquartile range (IQR) of the actuator length can be used to analyze how an MCA uses the available actuator workspace [38]. It is a measure of variability and is defined as the difference between the 75th and 25th percentile of the given sample. A high interquartile range denotes high usage of the actuator workspace.

6. Results and Discussion

6.1. Simulation Setup

The control scheme of the experiment is shown in Figure 6.

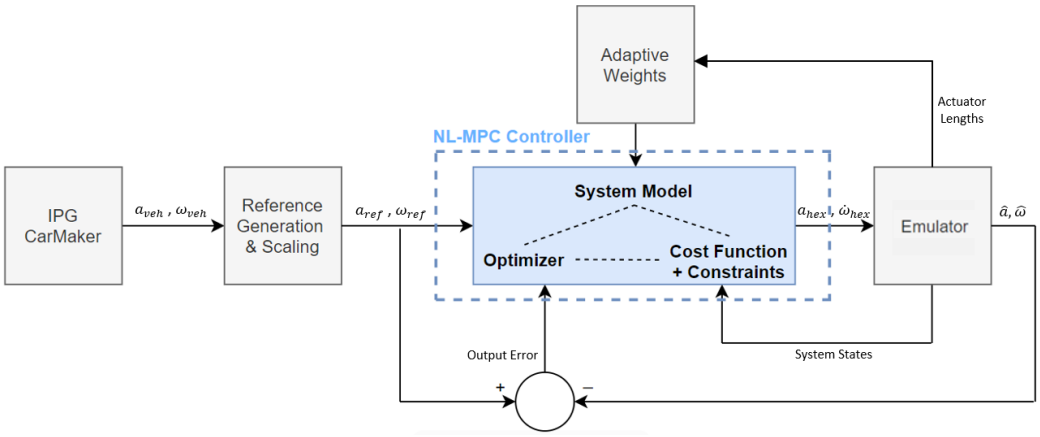

Figure 6. The control scheme used for full-track simulations.

A hatchback car was simulated using experimentally validated simulation model on the Hockenheim ring (Germany) in the IPG Carmaker software. The maneuver was simulated for a single lap on the circuit. A virtual sensor was placed on the eyepoint of the driver to record the accelerations and angular velocities. The resulting signal was recorded, passed on to the vestibular system model and the output was fed to the reference generator. Since a 1:1 reproduction of these quantities is not possible due to limited workspace of the motion platform, a scaling factor was used. Moreover, Grácio et al. [39] established that a 1:1 ratio of the inertial and visual cues are reported as too strong by the participants and thus, not preferred. Based on the mentioned study, the optimal scaling factor, known as optimal gain depends on the amplitude and the frequency of the stimuli, and the preferred motion gain decreases with the increase of the stimuli amplitude. Taking this into account and the capabilities of the motion platform, a scaling factor of 0.5 applied to the reference quantities and the resulting signals are passed to the controller as the reference signal. At every time step, the adaptive weights are calculated based on the actuator lengths. Furthermore, the controller receives the system states and the output error from the motion platform calculating the control input

using the nonlinear MPC scheme. The calculated control input, i.e., the translational and rotational acceleration of the moving base centroid of the platform is passed to the industrial motion platform emulator [40] representing the performance of the Delft Advanced Vehicle Simulator (DAVSi), which is shown in Figure 7. The dynamic threshold values for the latency of platform motion are in the range of 10 to 20 ms depending on the direction of motion [41].

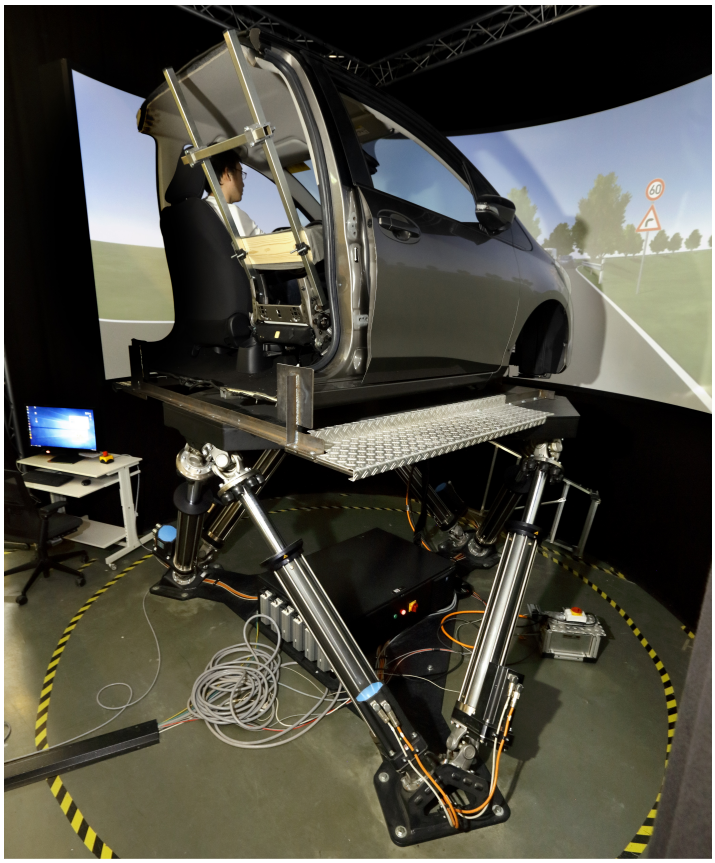

Figure 7. The Delft Advanced Vehicle Simulator.

The system performance of the motion platform is summarized in Table 3.

Table 3. System performance of the motion platform.

Motion	Excursion [m]	Velocity [m/s]	Acceleration [m/s^2]
Surge x	$-0.51 \ldots 0.63$	± 0.81	± 7.1
Sway y	$-0.51 \ldots 0.51$	± 0.81	± 7.1
Heave z	$-0.42 \ldots 0.42$	± 0.61	± 10.0
Roll ϕ	± 24.3	± 35.0	± 260.0
Pitch θ	$-25.4 \ldots 28.4$	± 38.0	± 260.0
Yaw ψ	± 25.0	± 41.0	± 510.0
Actuator	$1.297 \ldots 1.937$	-	-

The algorithm performance is analyzed based on the performance indicators mentioned in Section 5. The results of the proposed nonlinear MPC-based MCA (NLMPC) are compared with Linear MPC (LMPC) and the classical washout filter (CWF)-based MCAs. All the algorithms were tuned to maximize the reference tracking performance while keeping the actuator positions within the physical limits.

The simulations were performed on a standard Intel Core i7 2.6 GHz system with 16 GB RAM and x 64-bit Windows 10 operating system and later dSPACE Scalexio system to check hard real-time

feasibility. Furthermore, to test the real-time capabilities of the algorithm, the execution time required by the ACADO solver to solve the optimal control problem (OCP) was collected and shown in Figure 8.

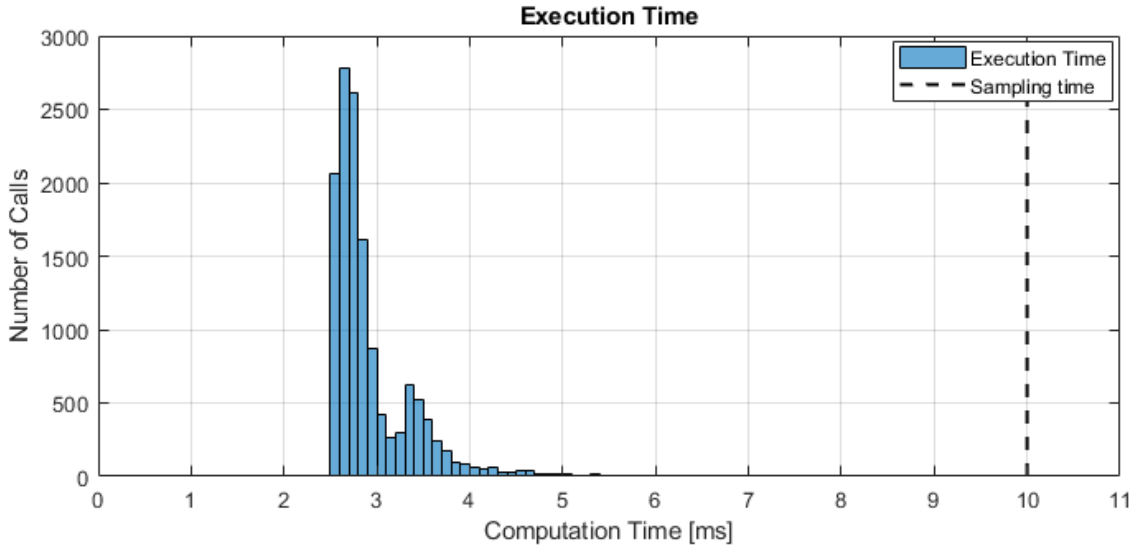

Figure 8. The execution time required by ACADO to solve the OCP.

It can be inferred from the Figure 8 that the time taken by the solver at each time step throughout the simulation is less than the sampling time (0.01 s), making it feasible to implement in real time.

6.2. Simulation Results

The reference tracking performance for perceived acceleration is shown in Figures 9–11 while the reference tracking performance for the angular velocities is shown in Figures 12–14. To further analyze the workspace use, actuator lengths spanned during the course of this simulation are shown in Figure 15.

6.2.1. Reference Tracking Performance: Translational Acceleration

From Figure 9, it can be inferred that the NLMPC algorithm results in a lower RMSE value compared to the benchmarked algorithms. Moreover, the high CC value indicates a high shape correlation. The RMSE and CC values for the CWF and LMPC algorithms indicate a comparatively inferior performance. Although the LMPC algorithm produces better results than CWF, both the algorithms produce false or missing cues. It can also be seen that both the MPC-based algorithms result in higher ED value compared to the CWF algorithm. This behavior of the MPC-based algorithms can be improved if the reference is known a priori.

Similar conclusions can be drawn from Figure 10. The NLMPC algorithm produces a superior reference tracking performance, which is reflected in the low RMSE and high CC values. Although the LMPC algorithm results in significant RMSE value, its CC value is high. This is because it produces correct but scaled-down cues resulting in high shape correlation; however, the high error as well. Moreover, the performance of CWF is again inferior compared to the other two algorithms.

From Figure 11, it can be inferred that since the value of the reference signal is small, all the three algorithms result in comparatively lower RMSE values. The NLMPC algorithm produces a high CC value compared to the other algorithms. As mentioned before, the ED value is high for both the MPC-based algorithms in all the above cases meaning that the produced cues are delayed. A reference prediction strategy can be used to improve this behavior.

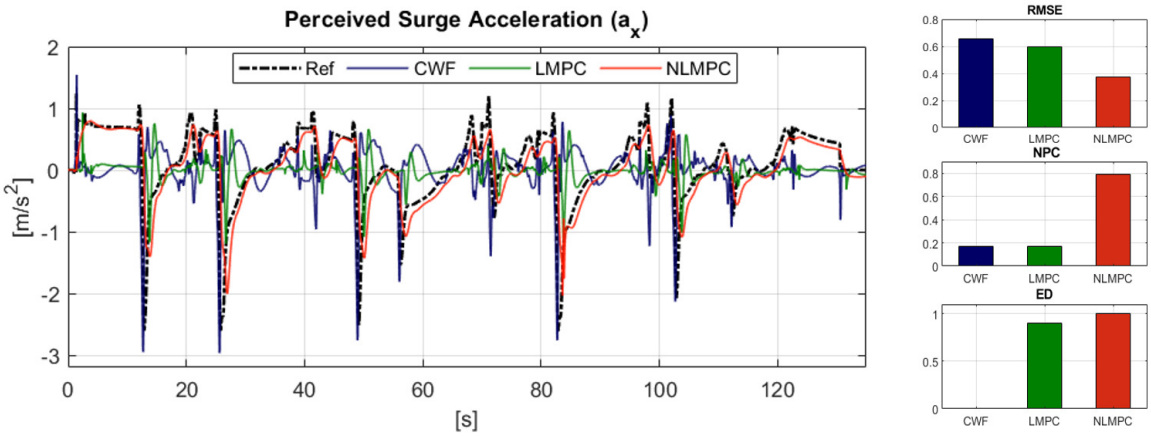

Figure 9. Reference Tracking performance: Perceived Surge Acceleration.

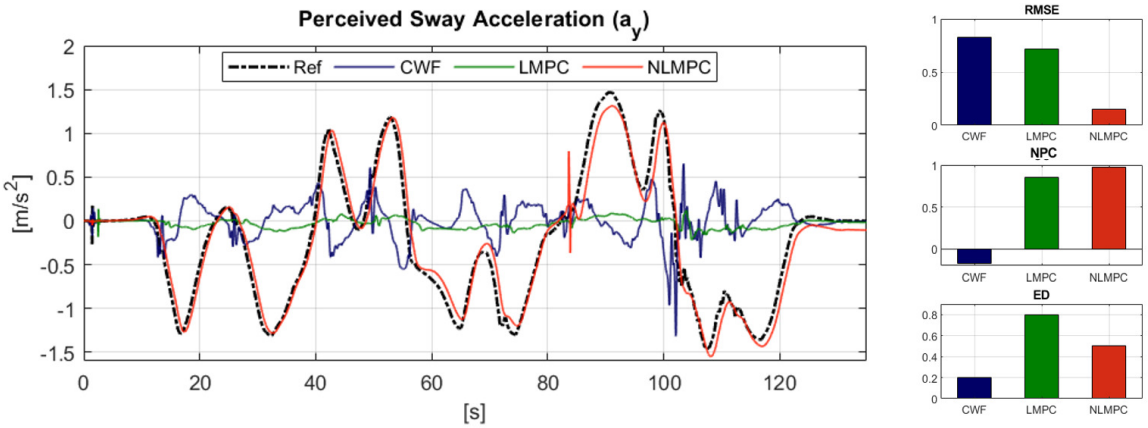

Figure 10. Reference Tracking performance: Perceived Sway Acceleration.

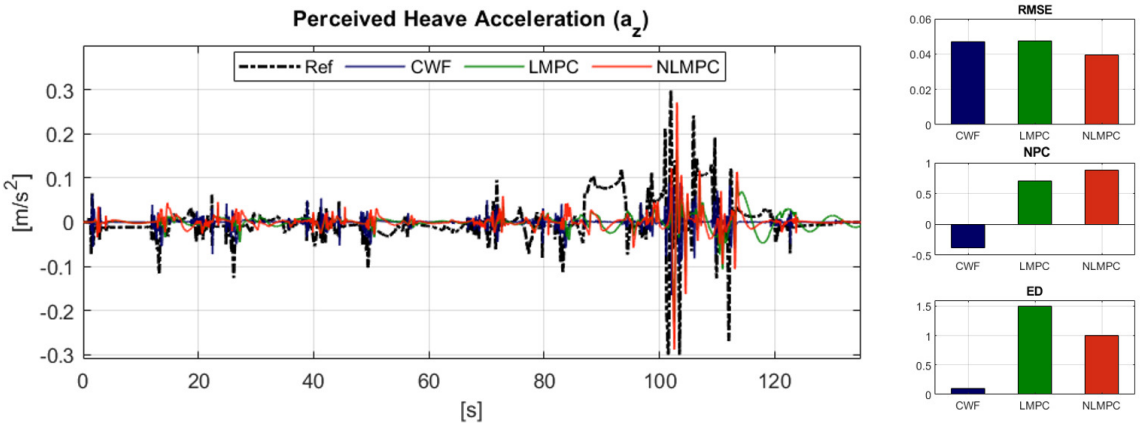

Figure 11. Reference Tracking performance: Perceived Heave Acceleration.

6.2.2. Reference Tracking Performance: Rotational Velocity

The reference tracking performance for the perceived angular velocity is shown in Figures 12–14. Since most of the reference cues are below the perception threshold, the quality of the produced cues in such cases does not matter as long as it is below the threshold. Therefore, the MCA performance should be judged based on the quality of the cues above the perception threshold.

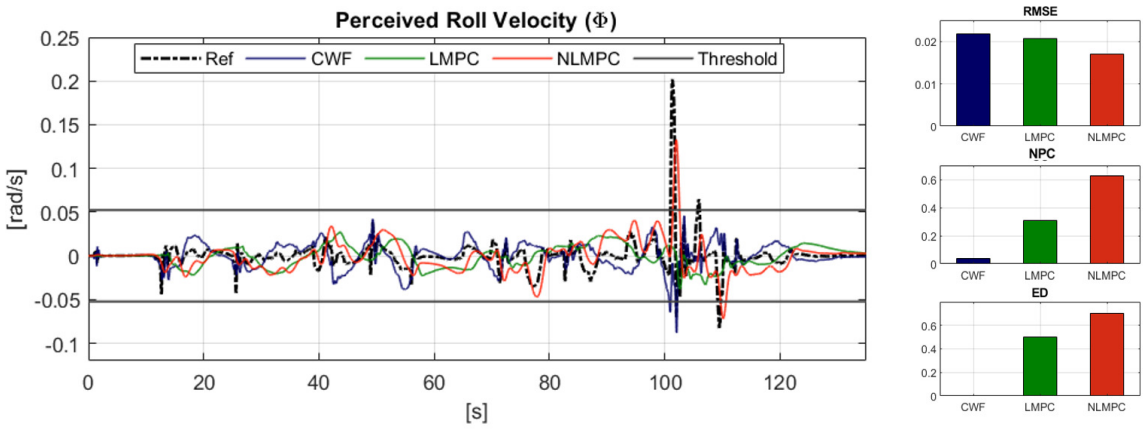

Figure 12. Reference Tracking performance: Perceived Roll Velocity.

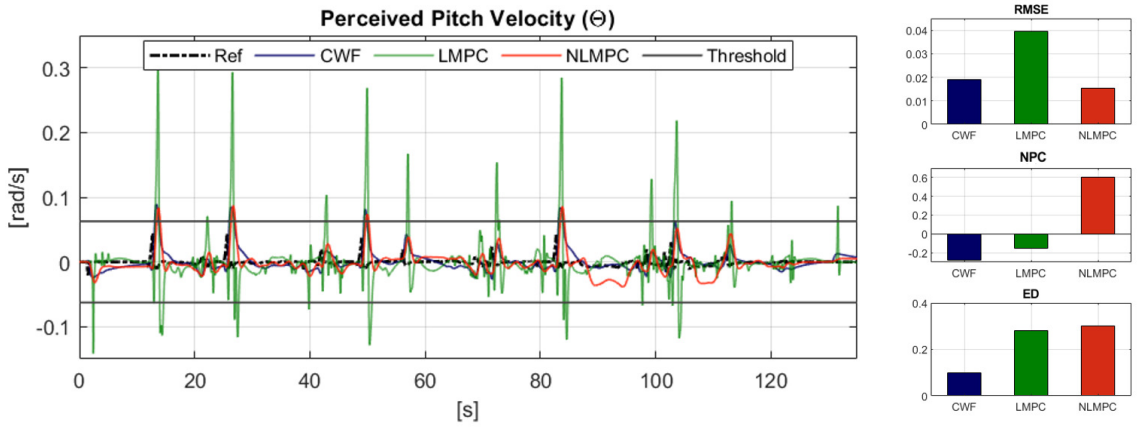

Figure 13. Reference Tracking performance: Perceived Pitch Velocity.

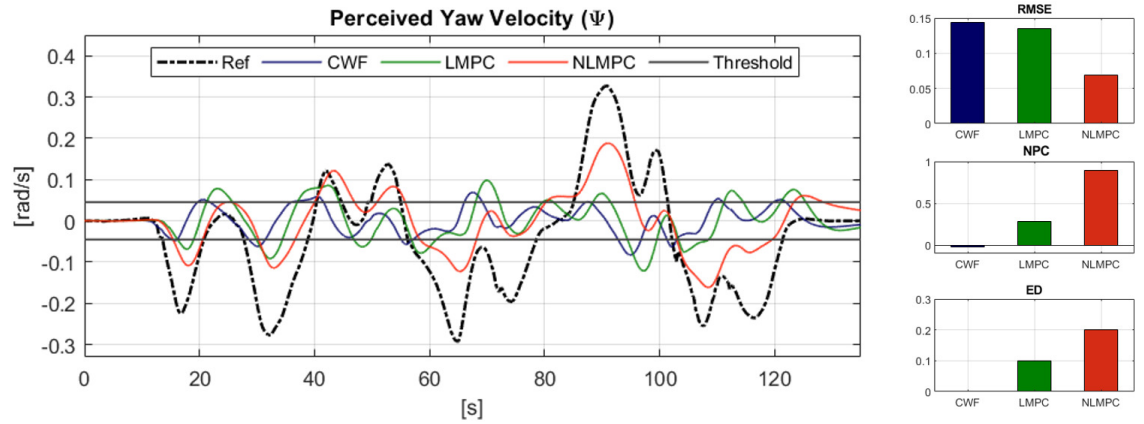

Figure 14. Reference Tracking performance: Perceived Yaw Velocity.

From the presented results, it can be inferred that the NLMPC algorithm outperforms the LMPC and CWF algorithm in terms of both the RMSE and the CC values. Moreover, a high ED value is observed for both the MPC-based algorithms.

From Figure 13, it can be seen that the LMPC algorithm produces perceivable false cues. For every peak that the algorithm tracks on the positive side, it produces an opposite peak in the negative side, resulting in a false cue. This is because after producing the cue from the high peak, the algorithm quickly tries to bring back the platform to the neutral position (washout effect), resulting in the production of the false cue. In NLMPC algorithm, this behavior is governed by the adaptive

weight-based tuning scheme. The weights on the position and velocity of the hexapod is only increased when the platform is near its limits. Therefore, the washout process becomes effective only when the platform is near the limits which reduces the tendency of the algorithm to produce false cues.

6.2.3. Workspace Use: Actuator displacement

From Figure 15, it can be inferred that the CWF algorithm uses the available workspace conservatively, as reflected by the lower IQR value. This can be attributed to the fact that the algorithm was tuned as per the limiting excursion (excursion of actuator 1 at 102 s). Therefore, during the other parts of the simulation, the algorithm uses the workspace conservatively. The LMPC scheme allows overcoming this limitation as the algorithm has a better knowledge of the platform limits and the same is taken into account while optimizing at each time step to obtain the control action. This results in a higher IQR value for the LMPC algorithm. Meanwhile, the adaptive tuning scheme further allows the NLMPC algorithm to span more workspace as the washout effect becomes effective only when the platform is near its physical limits resulting in a higher IQR value.

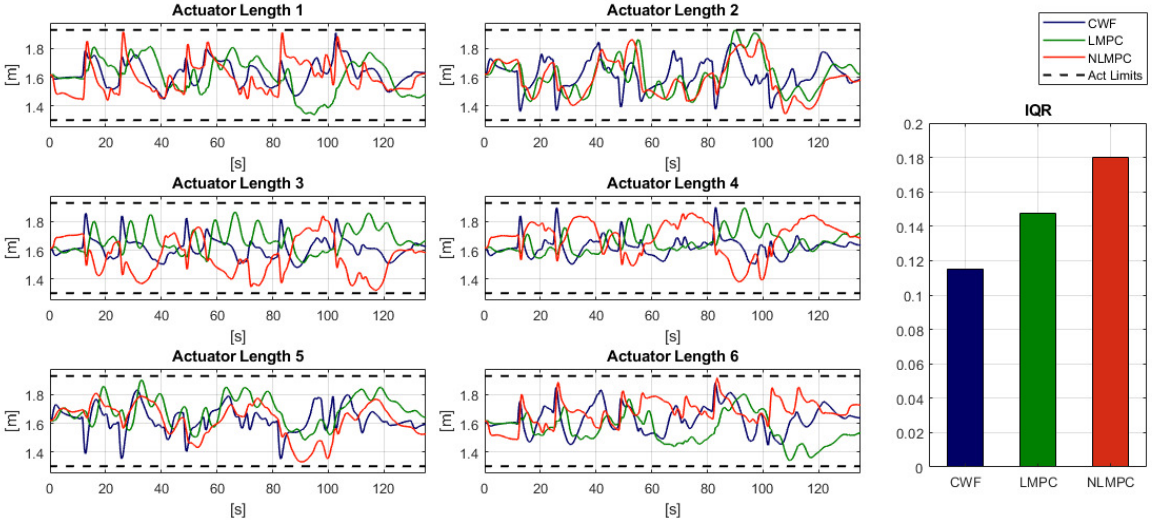

Figure 15. Workspace Use: Actuator Displacement and Mean IQR.

7. Conclusions

This paper aimed to propose a new nonlinear MPC-based motion-cueing algorithm incorporating the dynamic response of the vestibular system and the nonlinear kinematic model of the Stewart platform. The tilt coordination scheme is captured within the vestibular system model. To incorporate constraints on the rate of g-tilting without affecting the production of roll, pitch or yaw cues, the rotational velocity states are decoupled into separate rotational states (for actual rotational motion) and tilt states (for tilt coordination). Constraints are imposed on the tilt states. In the motion platform model, the actuator positions and velocities are calculated by using the nonlinear inverse kinematics of the Stewart platform and included as states within the MPC framework. Furthermore, the actuator displacements are constrained. Lastly, to manage the actuator workspace efficiently and attain smoother movement of the platform, an adaptive weight-based tuning methodology has been proposed changing the tuning weights on the platform displacement and velocities as per the available actuator motion space.

The proposed algorithm is evaluated in full-track simulations and its performance is compared to the classical washout filter and linear MPC-based MCA. Based on the literature, several performance indicators are defined to objectively evaluate and compare the reference tracking and workspace use performance of different MCAs.

The results showed superior performance of the proposed algorithm in terms of reference tracking over the linear MPC and CWF-based algorithms. The proposed algorithm produced less false or missing cues compared to the classical washout filter and linear MPC-based MCA which might reduce simulator sickness. Lastly, the proposed algorithm showed better workspace use compared to the linear MPC and CWF-based algorithms.

The further work is focused on hard real-time testing using a physical hexapod to validate the findings of this paper. Other extensions include a subjective evaluation of the MCA in active and passive driving conditions.

Author Contributions: Conceptualization, Y.R.K., M.G. and B.S.; data curation, Y.Z.; formal analysis, M.G.; investigation, Y.R.K.; project administration, B.S.; supervision, B.S.; writing—original draft, Y.R.K.; writing—review and editing, Y.Z., M.G. and B.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. De Winter, J.; van Leeuwen, P.M.; Happee, R. Advantages and disadvantages of driving simulators: A discussion. In Proceedings of the Measuring Behavior, Utrecht, The Netherlands, 28–31 August 2012; pp. 47–50.
2. Shyrokau, B.; De Winter, J.; Stroosma, O.; Dijksterhuis, C.; Loof, J.; van Paassen, R.; Happee, R. The effect of steering-system linearity, simulator motion, and truck driving experience on steering of an articulated tractor-semitrailer combination. *Appl. Ergon.* **2018**, *71*, 17–28. [CrossRef]
3. Nahon, M.; Reid, L. Simulator motion-drive algorithms—A designer's perspective. *J. Guid. Control. Dyn.* **1990**, *13*, 356–362. [CrossRef]
4. Sivan, R.; Ish-Shalom, J.; Huang, J. An optimal Control Approach to the Design of Moving Flight Simulators. *IEEE Trans. Syst. Man Cybern.* **1982**, *12*, 818–827. [CrossRef]
5. Lamprecht, A.; Steffen, D.; Haecker, J.; Graichen, K. Comparision between a filter and an MPC-based MCA in an offline simulator study. In Proceedings of the Driving Simulation Conference and Exhibition, Strasbourg, France, 4–6 September 2019.
6. Cleij, D.; Venrooij, J.; Pretto, P.; Katliar, M.; Bülthoff, H.H.; Steffen, D.; Hoffmeyer, F.W.; Schöner, H.-P. Comparison between filter- and optimization-based motion cueing algorithms for driving simulation. *Transp. Res. Part Traffic Psychol. Behav.* **2019**, *61*, 53–68. [CrossRef]
7. Cleij, D.; Pool, D.M.; Mulder, M.; Bülthoff, H.H. Optimizing an Optimization-Based MCA using Perceived Motion Incongruence Models. In Proceedings of the 19th Driving Simulation and Virtual Reality Conference, Antibes, France, 9–11 September 2020.
8. van der Ploeg, J.R.; Cleij, D.; Pool, D.M.; Mulder, M.; Bülthoff, H.H. Sensitivity Analysis of an MPC-based Motion Cueing Algorithm for a Curve Driving Scenario. In Proceedings of the 19th Driving Simulation and Virtual Reality Conference, Antibes, France, 9–11 September 2020.
9. Baseggio, M.; Bruschetta, M.; Maran, F.; Beghi, A. An MPC approach to the design of motion cueing algorithms for driving simulators. In Proceedings of the 14th IEEE international conference on Intelligent Transportation Systems, Washington, DC, USA, 5–7 October 2011; pp. 692–697.
10. Bruschetta, M.; Maran, F.; Beghi, A. A fast implementation of MPC based motion cueing algorithms for mid-size road vehicle motion simulators. *Veh. Syst. Dyn.* **2017**, *51*, 802–826. [CrossRef]
11. Husty, M. An Algorithm for Solving the Direct Kinematics of General Stewart-Gough Platform. *Mech. Mach. Theory* **1996**, *4*, 365–380. [CrossRef]
12. Garrett, N.; Best, M. Model predictive driving simulator motion cueing algorithm with actuator-based constraints. *Veh. Syst. Dyn.* **2013**, *51*, 1151–1172. [CrossRef]
13. Dagdelen, M.; Reymond, G.; Kemeny, A. Model-based predictive motion cueing strategy for vehicle driving simulators. *Control. Eng. Pract.* **2009**, *17*, 995–1003. [CrossRef]
14. Fang, Z.; Kemeny, A. Motion cueing algorithms for a real-time automobile driving simulator. In Proceedings of the Driving Simulation Conference, Paris, France, 6–7 September 2012.

15. Katliar, M.; Fischer, J.; Frison, G.; Diehl, M.; Teufel, H.; Bülthoff, H. Nonlinear Model Predictive Control of a Cable-Robot-Based Motion simulator. *IFAC-PapersOnLine* **2017**, *50*, 9833–9839. [CrossRef]
16. Katliar, M.; Olivari, M.; Drop, F.; Nooij, S.; Diehl, M.; Bülthoff, H. Offline motion simulation framework: Optimizing motion simulator trajectories and parameters. *Transp. Res. Part Traffic Psychol. Behav.* **2019**, *66*, 29–46. [CrossRef]
17. Gros, S.; Zanon, M.; Quirynen, R.; Bemporad, A.; Diehl, M. From linear to nonlinear MPC: Bridging the gap via the real-time iteration. *Int. J. Control.* **2020**, *93*, 62–80. [CrossRef]
18. Fichter, E.F. A Stewart platform-based manipulator: General theory and practical construction. *Int. J. Robot. Res.* **1986**, *5*, 157–182. [CrossRef]
19. Dasgupta, B.; Mruthyunjaya, T.S. The Stewart platform manipulator: A review. *Mech. Mach. Theory* **2000**, *35*, 15–40. [CrossRef]
20. Stewart, D. A Platform with Six Degrees of Freedom. *Proc. Inst. Mech. Eng.* **1965**, *180*, 371–386. [CrossRef]
21. Fernandez, C.; Goldberg, J. Physiology of peripheral neurons innervating otolith organs of the squirrel monkey. III. response dynamics. *J. Neurophysiol.* **1976**, *39*, 996–1008. [CrossRef] [PubMed]
22. Mayne, R. A Systems Concept of the Vestibular Organs. In *Vestibular System Part 2: Psychophysics, Applied Aspects and General Interpretations. Handbook of Sensory Physiology*; Kornhuber, H.H., Ed.; Springer: Berlin/Heidelberg, Germany, 1974.
23. Young, L.; Oman, C. Model for vestibular adaptation to horizontal rotation. *J. Aerosp. Med.* **1969**, *40*, 1076–1077.
24. Grant, W.; Best, W. Otolith-organ mechanics: Lumped parameter model and dynamic response. *Aviat. Space Environ. Med.* **1987**, *58*, 970–976.
25. Ormsby, C. Model of Human Dynamic Orientation. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1974.
26. Telban, R.J.; Cardullo, F.M. Motion Cueing Algorithm Development: Human-Centered Linear and Nonlinear Approaches. NASA Tech Report CR-2005-213747. 2005. Available online: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20050180246.pdf (accessed on 11 November 2020).
27. Harib, K.; Srinivasan, K. Kinematic and dynamic analysis of Stewart platform-based machine tool structures. *Robotica* **2003**, *21*, 541–554. [CrossRef]
28. Bock, H.; Plitt, K. A multiple shooting algorithm for direct solution of optimal control problems. In Proceedings of the 9th IFAC World Congress, Budapest, Hungary, 2–6 July 1984; pp. 243–247.
29. Vukov, M.; Domahidi, A.; Ferreau, H.; Morari, M.; Diehl, M. Auto-generated algorithms for nonlinear model predictive control on long and on short horizons. In Proceedings of the 52nd IEEE Conference on Decision and Control, Florence, Italy, 10–13 December 2013; pp. 5113–5118.
30. Magni, L.; Nicolao, G.; Magnani, L.; Scattolini, R. A stabilizing model-based predictive control algorithm for nonlinear systems. *Automatica* **2001**, *37*, 1351–1362. [CrossRef]
31. Mayne, D.; Rawlings, J.; Rao, C.; Scokaert, P. Constrained model predictive control: Stability and optimality. *Automatica* **2000**, 789–814. [CrossRef]
32. Abdelaal, M.; Franzle, M.; Hahn, A. Nonlinear Model Predictive Control for Tracking of Underactuated Vessels under Input Constraints. In Proceedings of the 2015 IEEE European Modelling Symposium, Madrid, Spain, 6–8 October 2015; pp. 313–318.
33. Grune, L.; Pannek, J. Stability and Suboptimality Without Stabilizing Terminal Conditions. In *Nonlinear Model Predictive Control: Theory and Algorithms*; Springer: Cham, Switzerland, 2011.
34. Reid, L.; Nahon, M. *Flight Simulation Motion-Base Drive Algorithms: Part 1—Developing and Testing the Equations*; UTIAS Report No. 296, CN ISSN0082-5255; Institute for Aerospace Studies, University of Toronto: Toronto, ON, Canada, 1985.
35. Houska, B.; Ferreau, H.; Diehl, M. ACADO Toolkit—An Open Source Framework for Automatic Control and Dynamic Optimization. *Optim. Control. Appl. Methods* **2011**, *32*, 298–312. [CrossRef]
36. qpOASES Homepage. Available online: http://www.qpoases.org (accessed on 11 November 2020).
37. Casas, S.; Coma, I.; Riera, J.V.; Fernández, M. Motion-cueing algorithms: Characterization of users' perception. *Hum. Factors* **2015**, *57*, 144–162. [CrossRef]
38. Grottoli, M.; Cleij, D.; Pretto, P.; Lemmens, Y.; Happee, R.; Bülthoff, H.H. Objective evaluation of prediction strategies for optimization-based motion cueing. *Simulation* **2018**, *95*, 707–724. [CrossRef]
39. Grácio, B.; van Paassen, M.; Mulder, M.; Wentink, M. Tuning of the lateral specific force gain based on human motion perception in the Desdemona simulator. In Proceedings of the AIAA Modeling and Simulation Technologies Conference, Toronto, ON, Canada, 2–5 August 2010.

40. Veltena, M.C. Movement Simulator. U.S. Patent No. 8,996,179, 31 March 2015.
41. Van Doornik, J.; Brems, W.; de Vries, E.; Uhlmann, R. Driving Simulator with High Platform Performance and Low Latency. *ATZ Worldw.* **2018**, *120*, 48–53. [CrossRef]