A Novel Framework for Closed-Loop Robotic Motion Simulation - Part II: Motion Cueing Design and Experimental Validation

P. Robuffo Giordano, C. Masone, J. Tesch, M. Breidt, L. Pollini, and H. H. Bülthoff

***Abstract*— This paper, divided in two Parts, considers the problem of realizing a 6-DOF closed-loop motion simulator by exploiting an anthropomorphic serial manipulator as motion platform. After having proposed a suitable inverse kinematics scheme in *Part I* [1], we address here the other key issue, i.e., devising a motion cueing algorithm tailored to the specific robot motion envelope. An extension of the well-known classical washout filter designed in *cylindrical* coordinates will provide an effective solution to this problem. The paper will then present a thorough experimental evaluation of the overall architecture (inverse kinematics + motion cueing) on the chosen scenario: closed-loop simulation of a Formula 1 racing car. This will prove the feasibility of our approach in fully exploiting the robot motion capabilities as a motion simulator.**

I. Introduction

In *Part I* of this work [1] we addressed the problem of achieving a 6-DOF closed-loop motion simulation through the use of an anthropomorphic serial robot arm — the CyberMotion Simulator, see Fig. 1. This manipulator, based on the commercial KUKA Robocoaster [2], is exploited as motion platform for reproducing the motion cues that would have been felt on a given (simulated) vehicle. Compared to standard Stewart platforms, a serial 6-DOF industrial manipulator offers higher dexterity, larger motion envelopes, the possibility to realize any end-effector posture within the workspace, and the ability to displace heavy loads (up to 500 [kg] in our case) with large accelerations and velocities. By attaching a cabin to the end-effector, one can then take advantage of the robot motion envelope to obtain a highly versatile tool for interactive vehicle simulation.

There are, however, some challenges that need to be addressed when adopting a manipulator arm as motion platform for closed-loop motion simulation, mainly the design of the *motion cueing* and of the *inverse kinematics* algorithms. The motion cueing subsystem is responsible for transforming the 'ideal' vehicle motion into a Cartesian trajectory compatible with the motion platform limited workspace, but still inducing a realistic motion perception onto the user. A well-know example is the classical combination of washout filters and tilt-coordination algorithms [3], [4] and all their variants. Anyway, such schemes are typically designed for Stewart-like platforms, while the motion envelope of a serial manipulator is inherently different, in particular more cylindrical than rectangular in Cartesian space. Therefore, some modifications are required to fully exploit the robot capabilities.

In addition, the filtered cabin motion is in general unpredictable and geometrically arbitrary, since it eventually depends on the (unpredictable) user's inputs to the simulated vehicle. This makes it hard to design an effective inverse kinematics scheme: the sought algorithm must realize the desired (but unknown in advance) cabin motion and, at the same time, cope with all the typical robot constraints (joint limits, actuator limitations) and avoid singularities in real-time. As discussed in *Part I*, one has to assume that the desired cabin trajectory may in general violate any or all robot constraints, and design an inverse kinematics scheme that aims at realizing online the *best feasible motion*.

Goal and contribution of this *Part II* is to conclude the discussion opened in *Part I*, and to demonstrate the feasibility and effectiveness of the proposed architecture for closed-loop motion simulation. To this end, we will first present the design of a motion cueing algorithm tailored to the specific robot motion envelope (Sect. II). This algorithm will then be used to generate the reference cabin trajectory tracked by the robot via the inverse kinematics illustrated in *Part I*. We will subsequently focus on the experimental evaluation of the whole architecture (motion cueing combined with the inverse kinematics of *Part I*) on the chosen test scenario: closed-loop simulation of a Formula 1 racing car. We will first illustrate the dynamical model adopted for the car simulation (Sect. III), and then analyze the experimental data collected while driving the car along a lap on a virtual track (Sect. IV). A video of this lap showing the robot motion is also attached to the paper. We will then conclude by summarizing the results and discussing the open points (Sect. V).

II. Design of the Motion Cueing Algorithm

As explained in the previous Section, the goal of a motion cueing algorithm is to 'filter' the ideal motion of the simulated vehicle to make it compatible with the limited workspace of a motion platform fixed to the ground, while still inducing the same *motion perception* onto the user. This is a fundamental but difficult problem that has been studied since the early '70 [5] until nowadays [6], and keeps drawing the attention of many researchers. Perhaps the most widespread motion cueing algorithm is the so-called *classical washout filter*, i.e., a combination of washout filters for

P. Robuffo Giordano, J. Tesch, and M. Breidt are with the Max Planck Institute for Biological Cybernetics, Spemannstraße 38, 72076 Tübingen, Germany {paolo.robuffo-giordano@tuebingen.mpg.de}.

C. Masone is with the Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Ariosto 25, 00185 Roma, Italy.

L. Pollini is with the Dipartimento di Sistemi Elettrici e Automazione, Università di Pisa, Via Diotisalvi 2, 56126 Pisa, Italy.

H. H. Bülthoff is with the Max Planck Institute for Biological Cybernetics, Spemannstraße 38, 72076 Tübingen, Germany, and with the Department of Brain and Cognitive Engineering, Korea University, Anam-dong, Seongbuk-gu, Seoul, 136-713 Korea.

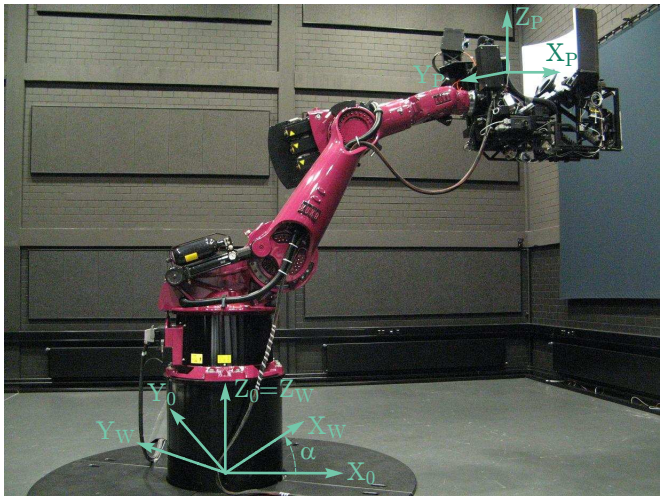


Fig. 1: A snapshot of the CyberMotion Simulator setup with some relevant reference frames

reproducing high-frequency motions, and tilt-coordination algorithms for reproducing low-frequency motions [3], [4].

Such algorithm can be summarized as follows: the vehicle motion, specified in terms of linear accelerations and angular velocities[1], is split into high-frequency and low-frequency components. The high-frequency component is reproduced by actually moving the motion platform, since this component will in general generate small (thus feasible) displacements. On the other hand, the low-frequency component, comprehensive of sustained linear accelerations, is not achieved by physically accelerating the platform, but by exploiting the local gravity vector as a source of 'persistent' acceleration. Indeed, by properly orienting the gravity vector in the cabin frame, one can reproduce the illusion of persistently accelerating in a given direction. This method, of course, has several limitations: for instance, it cannot reproduce sustained accelerations larger than 1 [g], it introduces rotational motion artifacts during the coordination phase, and it reduces the amount of gravity acceleration perceived by the user because of the tilting. Still, in most cases the classical washout framework represents the only viable option as motion cueing algorithm.

Many variants of this framework have been proposed in the past years, especially in the context of flight simulators. Just to mention a few, some attempts have been done in the direction of adaptive algorithms [8], [9], [10], and optimal control approaches [11], [12]. A thorough comparison can be found in [13], [14], [15]. Despite the big variety of solutions, it is worth noting that many authors still indicate the classical washout filter as one of the most effective algorithm in terms of design simplicity, easiness of tunability, and human perception fidelity [3], [16], [17]. Therefore, we decided to base our motion cueing algorithm on the classical washout framework, and to modify it in order to take into account the particular motion envelope of our motion simulator. The next Sections will illustrate the details of our implementation.

[1]These are thought to be the motion states sensed by humans [7].

A. High-pass channels

Compared to a Stewart platform, the motion envelope of the CyberMotion Simulator is closer to a cylinder rather than a rectangular box in Cartesian space. For instance, by exploiting the rotation of the first vertical joint, one can obtain large lateral displacements along circular trajectories, considerably larger than any achievable linear trajectory. This motivated us to design the high-pass filters for the linear acceleration in cylindrical coordinates, similarly to what proposed for the spherical washout filter [18] implemented in the Desdemona motion simulator developed at TNO [19]. The idea is to keep linear forward and upwards motions unchanged, and replace linear lateral motions with circular motions (i.e., moving on the surface of a vertical cylinder).

Before proceeding, we will define some needed quantities consistently with the notation introduced in *Part I*. With reference to Fig. 1, let $\mathcal{F}_0 : \{O; \vec{X}_0, \vec{Y}_0, \vec{Z}_0\}$ be a world reference frame fixed to the robot base, with $\vec{Z}_0$ pointing upwards and $(\vec{X}_0, \vec{Y}_0)$ spanning the horizontal plane. A moving reference frame $\mathcal{F}_P : \{O_P; \vec{X}_P, \vec{Y}_P, \vec{Z}_P\}$ is attached to the the pilot's head (supposed fixed to the cabin) and has its axes aligned with the pilot's forward/left/upward direction, respectively. Let also $p = [x \ y \ z]^T \in \mathbb{R}^3$ represent the coordinates of O_P in $\mathcal{F}_0$. We can transform the Cartesian coordinates p into cylindrical ones $\xi = [R \ \alpha \ z]$ defined as

$$\begin{cases} R &= \sqrt{x^2+y^2} \\ \alpha &= \text{atan2}(y, x) \\ z &= z \end{cases},$$

and introduce a third moving frame $\mathcal{F}_W : \{O_W; \vec{X}_W, \vec{Y}_W, \vec{Z}_W\}$, denoted as the *washout frame*, with $O_W \equiv O$, $\vec{Z}_W \equiv \vec{Z}_0$, and $\vec{X}_W$ rotated of angle α w.r.t. $\vec{X}_O$. Therefore, axis $\vec{X}_W$ will always point towards the current angular position α on the cylinder.

Furthermore, let ${}^W R_P$ be the rotation matrix from frame $\mathcal{F}_W$ to frame $\mathcal{F}_P$, and $\eta = [\rho \ \theta \ \psi]^T \in \mathbb{R}^3$ the usual set of roll-pitch-yaw Euler angles parameterizing ${}^W R_P$, while the rotation matrix from frame $\mathcal{F}_0$ to frame $\mathcal{F}_W$ is just ${}^0 R_W = R_{Z_0}(\alpha)$. Let us also define g as the gravity vector, and g as the scalar value of the gravity acceleration. Finally, a pre-superscript will indicate the reference frame where a quantity is defined, e.g., in $\mathcal{F}_0$ it is ${}^0 g = [0 \ 0 \ -\text{g}]^T$.

With these settings, we can illustrate our approach with the help of the block scheme shown in Fig. 2. Let us first consider the high-pass filter for the linear accelerations (top block in Fig. 2). The input to this filter is ${}^P a$, the linear acceleration of the vehicle expressed in the frame $\mathcal{F}_P$ and without gravity components, i.e., ${}^P a = {}^P f + {}^P g$ where ${}^P f$ is the specific force acting on the vehicle[2]. By following the classical washout framework, ${}^P a$ is first scaled and limited to obtain ${}^P a_S$ which is then expressed into $\mathcal{F}_W$ as ${}^W a_S$. These Cartesian accelerations are then transformed into cylindrical

[2]We recall that the specific force ${}^P f$ is defined as ${}^P a - {}^P g$, so that during free fall (${}^P a = {}^P g$) it is ${}^P f = {}^P g - {}^P g = 0$.

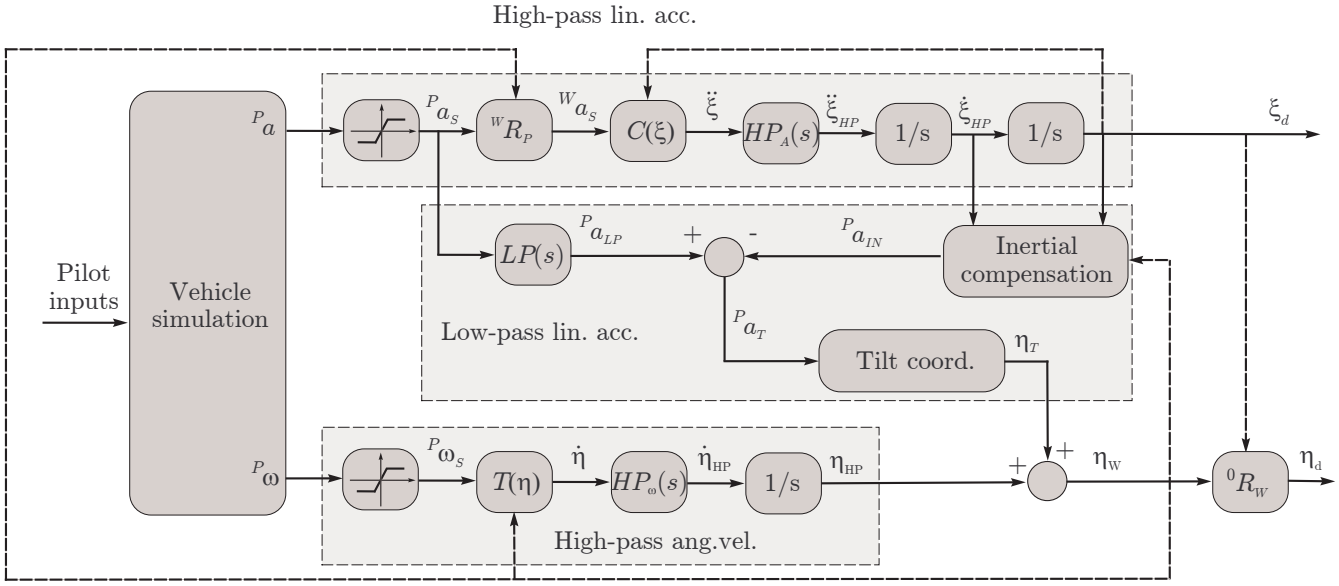


Fig. 2: A block scheme representation of the motion cueing algorithm adopted in this paper. The algorithm is based on the classical washout filter and tilt coordination [3], [4], but extended so as to take into account the cylindrical motion envelope of our motion simulator. Compared to the classical design, an additional subsystem, denoted as *Inertial compensation*, is also present. The purpose of this block is to compensate for the spurious centripetal and Coriolis accelerations due to the circular trajectories output of the filter

ones as

$$\ddot{\xi} = \begin{bmatrix} \ddot{R} \\ \ddot{\alpha} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{R} & 0 \\ 0 & 0 & 1 \end{bmatrix} {}^W a_S = C(\xi) {}^W a_S \quad (1)$$

The simple form of (1) clearly follows from the definition of $\mathcal{F}_W$. Linear forward accelerations in $\mathcal{F}_W$ correspond to radial accelerations $\ddot{R}$, and lateral accelerations in $\mathcal{F}_W$ correspond to angular accelerations $\ddot{\alpha} R$.

The accelerations $\ddot{\xi}$ are then high-pass filtered through the transfer function $HP_A(s)$ to yield the high-pass component of the linear motion $\ddot{\xi}_{HP}$, which is subsequently double-integrated into the desired platform displacement $\xi_{HP} = \xi_d$. Many choices are possible for $HP_A(s)$. The typical one is to take

$$HP_A(s) = \frac{s^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \cdot \frac{s}{s + \omega_b},$$

where the natural frequency ω_n, the damping ratio ζ, and the break frequency ω_b are in general different for each component of $\ddot{\xi}$. This choice ensures the *washout* characteristics of the filter, i.e., the fact that, at steady state and for constant inputs, the platform displacement $\xi_{HP} = \ddot{\xi}/s^2$ goes back to the initial position.

The angular high-pass channel (bottom block in Fig. 2) is designed as in the classical washout scheme. In short, the input angular velocity $^P\omega$ expressed in $\mathcal{F}_P$ is first scaled and limited, then transformed into the corresponding Euler rate $\dot{\eta}$ which is high-pass filtered through $HP_\omega(s)$ to obtain the high-pass component of the angular velocity $\dot{\eta}_{HP}$. This is finally integrated into the corresponding angular displacement η_{HP}. Here, $HP_\omega(s)$ can be chosen as a second-order high-pass filter

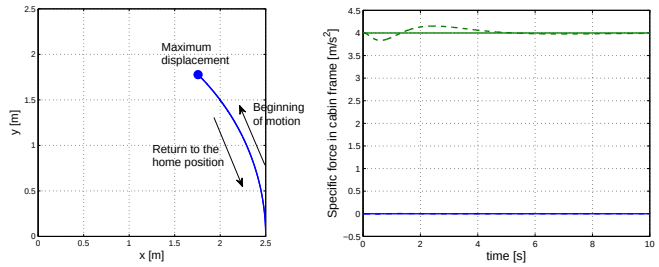
$$HP_\omega(s) = \frac{s^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

to ensure the washout characteristics.

B. Low-pass Channel and Tilt Coordination

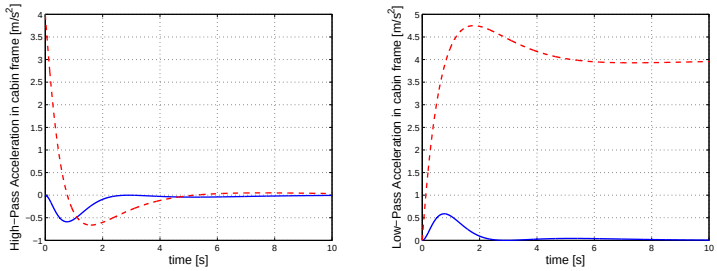
As explained at the beginning of this Section, the purpose of this step is to orient $^P g$ so as to simulate presence of sustained linear accelerations. This is achieved by matching the low-pass components of $^P a$ with the corresponding components of $^P g$ through the so-called tilt coordination algorithm. In our case, however, we extended this idea to also compensate for the undesired inertial accelerations due to the choice of working in cylindrical coordinates. Indeed, the benefits of moving along a horizontal circular trajectory instead of a straight line (i.e., increased robot workspace) come at the price of introducing spurious centripetal and Coriolis accelerations during the motion. These disturbances can be largely attenuated by using the gravity vector $^P g$ to filter them out.

Let us again refer to the scheme in Fig. 2, in particular to the middle block. The scaled linear acceleration $^P a_S$ is first low-pass filtered through $LP(s)$ into $^P a_{LP}$. We chose to set $LP(s) = 1 - HP_A(s)$ in order to obtain a perfect coordination between the high-pass and low-pass channels. At this point, the standard tilt coordination algorithm would compute the required cabin orientation η_T to match the first two components of $^P g$ with $-{}^P a_{LP}$. By imposing $^P R_W {}^W g =$



(a) (b)

Fig. 3: Left: planar trajectory of the cabin in $\mathcal{F}_0$. The cabin starts from $(2.5,\ 0)$, travels along a circular path, and then goes back to the starting position. Right: specific force felt on the cabin during the motion (dashed lines) versus the input acceleration (solid lines). The desired lateral acceleration of 4 $[\mathrm{m/s^2}]$ (upper curve) is almost perfectly reproduced, as well as the desired (zero) forward acceleration (lower curve)



(a) (b)

Fig. 4: Left: high-pass components of the cabin acceleration. Right: low-pass components of the cabin acceleration. Note how the lateral accelerations in both plots (red dashed lines) have an almost symmetric behavior. The initial peak on the high-pass forward acceleration (solid blue line on the left) is due to the centripetal acceleration associated to the circular trajectory followed by the cabin, and is compensated by the corresponding peak on the low-pass acceleration (solid blue line on the right)

$-{}^P a_{LP}$, one obtains

$$\begin{cases} \rho_T &= \arcsin \dfrac{{}^P a_{LP_y}}{\mathrm{g}\cos\theta_T} \\ \theta_T &= -\arcsin \dfrac{{}^P a_{LP_x}}{\mathrm{g}} \\ \psi_T &= 0 \end{cases} .$$

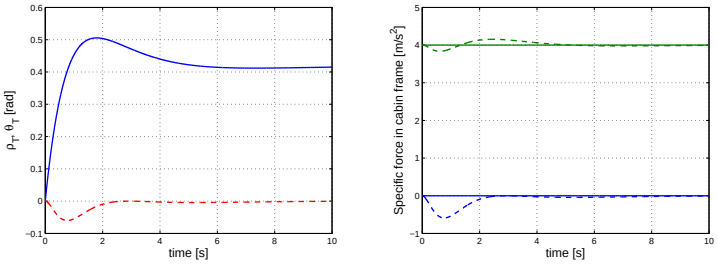
These angles are typically rate limited to avoid a strong rotational cueing on the user.

The centrifugal and Coriolis accelerations due to the cylindrical motion in $\mathcal{F}_W$ are ${}^W a_{IN} = [-R\dot{\theta}^2 \quad 2\dot{R}\dot{\theta} \quad 0]^T$. These can be transformed into the cabin frame ${}^P a_{IN} = {}^P R_W {}^W a_{IN}$, and subtracted from ${}^P a_{LP}$ to obtain the final acceleration vector ${}^P a_T = {}^P a_{LP} - {}^P a_{IN}$ to be sent to the tilt coordination algorithm. The corresponding η_T is then added to angular displacement η_{HP} from the high-pass angular velocity channel to yield the sought cabin orientation $\eta_W = \eta_T + \eta_{HP}$. The final step is to transform η_W into the corresponding world frame quantity. By recalling the definition of $\mathcal{F}_W$, this can be easily achieved by adding the angular displacement α to the third (yaw) component of η_W, thus obtaining the desired cabin orientation input to the robot inverse kinematics $\eta_d = \eta_W + [0\ 0\ \alpha]^T$.

C. Simulation results

Here, we present a simulation example that will effectively illustrate the main features of our washout implementation. The corresponding experimental data are reported in Sect. IV. We chose to simulate a lateral constant acceleration of the cabin ${}^P a = [0 \ \ 4 \ \ 0]^T \ [\mathrm{m/s^2}]$. The parameters of $HP_A(s)$ were set to $\omega_n = .5$ [rad/s], $\zeta = 1$, and $\omega_b = 1.2$ [rad/s] for the three acceleration channels, and no rate limitation was applied to the tilt coordination angles η_T.

Figure 3(a) shows the resulting planar trajectory in $\mathcal{F}_0$. The cabin starts from $(2.5,\ 0)$ on the horizontal plane, travels along a circular arc until reaching a maximum angular displacement $\alpha_{max} \simeq 0.8$ [rad], and then goes back to the starting position along the same arc (washout effect).



(a) (b)

Fig. 5: Left: the behavior of the roll (solid blue line) and pitch (dashed red line) angles (ρ_T, θ_T) needed to properly orient the gravity vector in $\mathcal{F}_P$. Right: the resulting specific force on the cabin without compensating for the spurious centripetal acceleration. Compared to Fig. 3(b), an initial peak on the forward acceleration (dashed blue line) is present

Figure 3(b) illustrates the superimposition of the input acceleration ${}^P a$ (solid lines) and the total specific force felt on the cabin (dashed lines), i.e., the vectorial sum of the actual cabin acceleration due to $\ddot{\xi}_{HP}$, and $-{}^P g$. In particular, the upper curve represents the lateral specific force and the lower curve the forward specific force. One can then verify an almost perfect reproduction of the desired cabin acceleration, thanks to this combined action.

This is also shown in Figs. 4(a–b) where the individual low-pass component ${}^P a_T$ and high-pass component $\ddot{\xi}_{HP}$ are reported. We note that the high-pass component starts from 4 $[\mathrm{m/s^2}]$ and then exponentially vanishes over time, while the low-pass component has an almost symmetric profile. Note also that the forward component of the high-pass motion (blue solid line in Fig. 4(a)) has a negative peak of approx. -0.6 $[\mathrm{m/s^2}]$ at time 0.76 [s]. This is the effect of the centripetal acceleration associated to the circular trajectory followed by the cabin. However, thanks to the inertial compensation action, the low-pass component shows a corresponding positive peak (blue solid line in Fig. 4(b)) that cancels this undesired acceleration in the final specific force felt by the user (blue dashed line in

Fig. 3(b)). Figure 5(a) reports the behavior of $(\rho_T,\, \theta_T)$ over time, i.e., the cabin roll and pitch angles output of the tilt coordination algorithm needed to properly orient the gravity vector in $\mathcal{F}_P$. In particular, ρ_T (solid blue line) is used to simulate the sustained acceleration, and θ_T (dashed red line) to compensate for the centripetal acceleration.

Finally, we show in Fig. 5(b) the cabin specific force obtained without compensating for the inertial effects due to the circular motion. In comparison with Fig. 3(b), we clearly see the presence of a spurious negative acceleration in the forward cabin axis (dashed blue line) consequence of the uncompensated centripetal acceleration.

III. Car Dynamical Model

In order to generate the accelerations ${}^{P}a$ and angular velocities ${}^{P}\omega$ input to the motion cueing algorithm, a fully non linear car model was developed. A detailed description of the car dynamics is outside the scope of this article; nevertheless, we will give an overview of the complex dynamic model used for our motion simulation.

The complete car was modeled as an articulated multibody system: the chassis including pilot and engine, the four suspension arms, the four spring/damper groups, the four wheel hubs, and the four wheels. The whole car is basically composed by four wheels interacting with the terrain, and connected together by the suspensions and car chassis. Figure 6(a) shows the various components of the model. For simplicity, the masses of suspension arms and spring/damper groups were included in the chassis mass, while the masses of the wheel hub and the wheel itself were considered as a unique body for generating linear motion, and separated for generating rotational motion.

The 32 system state variables are the following: car center of gravity CoG, linear velocity v_{CoG}, angular velocity ω_{CoG}, position p_{CoG}, and Euler angles: $[\phi_{CoG}, \theta_{CoG}, \psi_{CoG}]$ defining the car body reference system $\mathcal{F}_{CoG}$ w.r.t. $\mathcal{F}_W$, the position (deflection) δ and velocity $\dot{\delta}$ of the four wheel hubs along the spring/damper axis, the position p_{wheel} and velocity v_{wheel} of the four wheel hubs along their respective vertical axis of motion w.r.t. a reference terrain level, and the angular velocity ω_{wheel} of the four wheels around their respective spin axis.

The external forces acting on the system are: gravity g, contact forces and moments of each wheel (F_x, F_y, M_z), traction Q and braking B torques, and longitudinal F_{Cx}, lateral F_{Cy}, and vertical F_{Cz} aerodynamics loads; the steering angle λ is modeled as a direct user input.

According to the *Newton-Euler formulation*, the dynamic equations of the car dynamics can be written in compact form as

$$\begin{aligned} \dot{x}_1 &= \\ &M^{-1}(x_2)\left[J_1(x_2)F(x_1,\, x_2) - C(x_1,\, x_2)x_1 - S(x_1,\, x_2) - g(x_2)\right] \\ \dot{x}_2 &= J_2(x_2)x_1 \end{aligned} \tag{2}$$

where the system state vector is

$$\begin{aligned} x_1 &= \left[v_{CoG}^T\; \omega_{CoG}^T\; \dot{\delta}^T\; v_{wheel}^T\; \omega_{wheel}^T\right]^T \\ x_2 &= \left[p_{CoG}^T\; \phi_{CoG}^T\; \theta_{CoG}^T\; \psi_{CoG}^T\; \delta^T\; p_{wheel}^T\right]^T \end{aligned}$$

with δ, $\dot{\delta}$, p_{wheel}, v_{wheel} and ω_{wheel} being 4-dimensional vectors containing the corresponding variables of the four wheels.

Matrix $M(x_2)$ contains all the mass and inertia terms of the articulated body and is function of the suspension configuration. Matrix $C(x_1,\, x_2)$ takes into account centripetal and Coriolis terms, the term $S(x_1,\, x_2)$ represents the suspension related forces, and the term $g(x_2)$ the gravity forces.

The input vector F contains all the wheel/terrain interaction forces for the four wheels, as well as the aerodynamics forces. Matrix $J_1(x_2)$ determines how these forces, computed with respect to the wheel (the former) or car chassis (the latter), affect the car dynamics as a function of the current suspension state of compression and car attitude.

Matrix $J_2(x_2)$ contains the relevant rotation matrices and *integrators* needed to compute the car position and attitude, and all the wheel and suspension variables from the corresponding velocity variables in x_1.

A. Car Suspensions

Formula 1 front and rear suspensions are very complex, thus a simplified geometry was adopted. The wheel suspension is usually modeled as an articulated closed chain made of suspension arms, wheel hub, the car chassis and possibly the spring/damper group. The wheel-chassis relative motion, together with the car chassis attitude, determines how the wheel touches the ground during both lateral maneuvers/cornering and load transfers due to an acceleration or a braking.

Three main angles define how the wheel is positioned w.r.t. the terrain: toe (τ), camber (χ) and caster (γ) angles. Figure 6(b) shows the complete suspension geometry implemented in the simulation, where the subscript FL stands for *Front Left* wheel. Wheel camber and caster angles are not constant and vary with the suspension motion, and the car roll, and pitch angles. The complexity of simulating the closed kinematic chain of the suspension was overcome by using suitable kinematic constraints. The camber and caster angles values during the simulation are defined by the respective rest values χ_0 and γ_0, by the Suspension Arm rotation angle θ, which represents the current state of the suspension, and by the car attitude:

$$\chi(t) = f_\chi\left(\chi_0,\, \theta(t),\, {}^{W}R_{CoG}(t)\right) \tag{3}$$

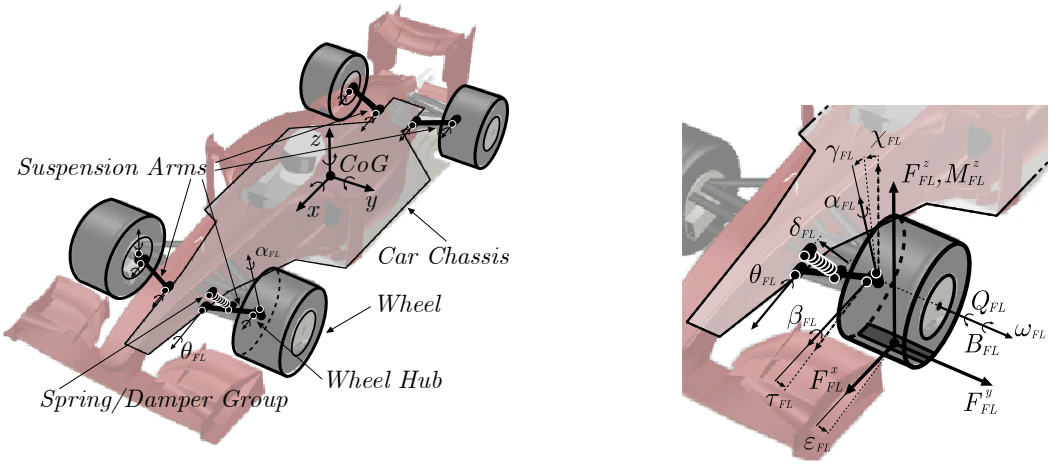
$$\gamma(t) = f_\gamma\left(\gamma_0,\, \theta(t),\, {}^{W}R_{CoG}(t)\right) \tag{4}$$

The angles θ and β are not actual state variables, but their value depend via algebraic equations from the spring/damper group compression δ, the CoG position p_{CoG}, the wheel hub position p_{wheel}, the chassis attitude ${}^{W}R_{CoG}$ and from the suspension geometry:

$$\theta(t) = f_\theta\left(\delta(t),\, p_{CoG}(t), p_{wheel}(t),\, {}^{W}R_{CoG}(t)\right) \tag{5}$$

$$\beta(t) = f_\beta\left(\delta(t),\, p_{CoG}(t), p_{wheel}(t),\, {}^{W}R_{CoG}(t)\right) \tag{6}$$

The steering angle $\lambda(t)$, together with toe τ, determines the orientation of the tyre w.r.t. the car chassis, and thus the car velocity vector. The wheel/terrain interaction produces the longitudinal reaction force $F_x(t)$, the lateral or cornering



(a) (b)

Fig. 6: Left: the various components of the multibody system modeling the Formula-1: chassis, wheels, spring/damper groups, wheel hubs, and suspension arms. Right: detailed view of the suspension geometry

force $F_y(t)$, and the aligning moment M_z which are all function of the vertical load $F_z(t)$ and other wheel variables.

The complete car dynamic equations are then constituted by (2) and by the kinematic constrains (3–6).

Modeling of the tyre(wheel)/terrain interaction poses severe challenges that are usually approached by using simplifying assumptions. For the goals of this paper, the *Pacejka's Magic Formula* [20] was adopted. Although the Magic Formula approach does not try to model the physical interaction between tyre and terrain, it represents an effective way of approximating the effects of this interaction by using interpolating functions, and is widely used in automotive simulators.

IV. Experimental evaluation

In this Section we will discuss several experimental results conducted on the CyberMotion Simulator and aimed at analyzing the performance of the whole proposed architecture, i.e., the combination of the motion cueing algorithm with the inverse kinematics presented in *Part I*. The goal is to determine how these two subsystems interact when plugged together, and to clearly assess the robot suitability of being exploited as a motion simulator. We will first present the experimental data collected during the execution of the same motion discussed in Sect. II-C, i.e., a lateral constant acceleration of the cabin ${}^P a = [0\ 4\ 0]^T\ [\mathrm{m/s^2}]$. This will allow an easy comparison of performance with the ideal simulation case. We will then focus on the data collected while driving the Formula 1 car along a lap on the virtual track of Monza [21], the Italian official track of the Formula 1 world championship. Figures 8(a–b) show a screenshot of the 3D environment displayed to the user, and an overall view of the Monza track. The complex and sudden motions experienced during this lap will constitute a solid benchmark for our motion simulator.

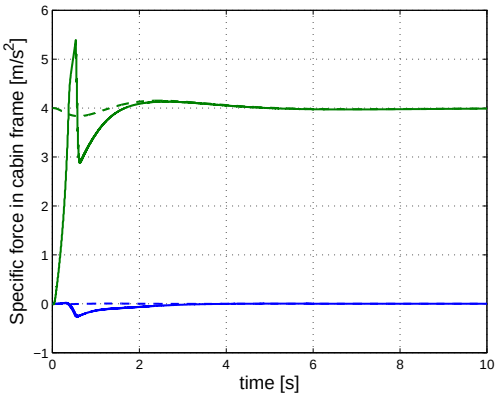
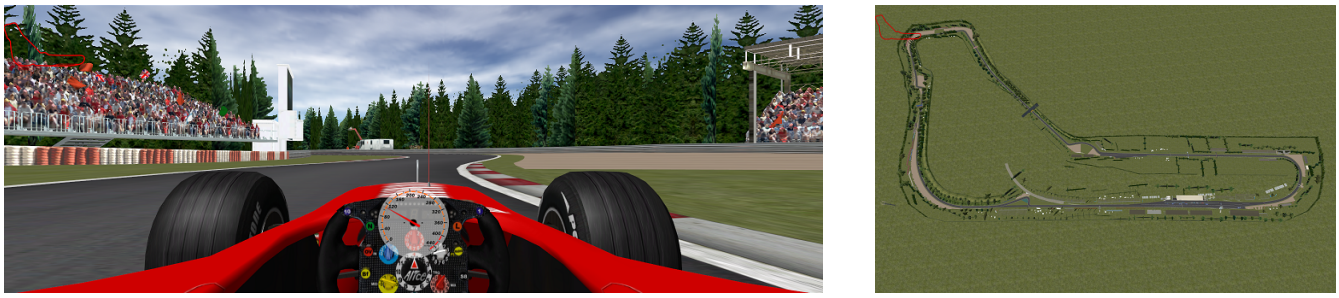


Fig. 7: Actual specific force felt on the cabin during the motion (solid lines) and desired specific force output of the motion cueing algorithm (dashed lines). Compared to the corresponding 'ideal' case of Fig 3(b), some lag is present in the robot motion due to the limited maximum joint accelerations. Nevertheless, the desired specific force is reasonably reproduced

A. Reproduction of a sustained acceleration

Analogously to Fig 3(b) in Sect. II-C, Fig. 7 shows the specific force felt on the cabin during the actual motion of the robot. This was evaluated by plugging the measured joint angles into the robot forward differential kinematics so as to compute the corresponding cabin accelerations. The resulting specific force (solid lines) is superimposed to the output of the washout filter (dashed lines) illustrated in Fig 3(b).

Apart from some distortions, we can essentially note some lag between the desired specific force and the actual one. The lateral component of the specific force (upper curve) reaches the nominal value of $4\ [\mathrm{m/s^2}]$ after about $0.3\ [\mathrm{s}]$, but then overshoots and settles back after a couple of seconds. This discrepancy is mainly due to the limitations in the robot actuators, in particular to the constraints on maximum joint accelerations that limit the responsiveness of the whole system. Indeed, as explained in *Part I*, the chosen design of



(a) (b)

Fig. 8: Left: a screenshot of the 3D environment displayed to the user on the onboard projection screen. Right: a bird's eye view of the Monza track

the motion cueing algorithm does not allow to take explicitly into account all the robot constraints expressed at the joint level. This is a clear example where the output trajectory of the motion cueing *violates* some robot constraints (max. joint acceleration), but the inverse kinematics block still tries to achieve the best feasible robot motion, i.e., a slightly 'delayed' response.

B. Driving a lap

For this closed-loop experiment, we tuned the parameters of the motion cueing algorithm by trial and error, and no rate limitation was applied to the angles η_T. Indeed, proper tuning of this action, as well as of all the other washout parameters, is a relevant but difficult problem that will be extensively addressed in future studies with the help of experienced drivers. At this step, we focused more on the motion reproduction capabilities of our system rather than on the quality of human motion perception. For the linear acceleration high-pass channel, we chose $\zeta = 1$, $\omega_n = .5$, $\omega_b = 1.2$ for the lateral acceleration, and $\zeta = 1$, $\omega_n = .6$, $\omega_b = 20$ for the forward and upward accelerations. As for the angular velocity high-pass channel, we set $\zeta = 1$ and $\omega_n = 1$ for all components. To produce ${}^{P}a_S$, we first scaled the forward/lateral accelerations by a factor 0.8 and 0.2, respectively, and then saturated all the components to a maximum value of ± 4 $[\mathrm{m/s^2}]$. The angular velocity was neither scaled nor saturated.

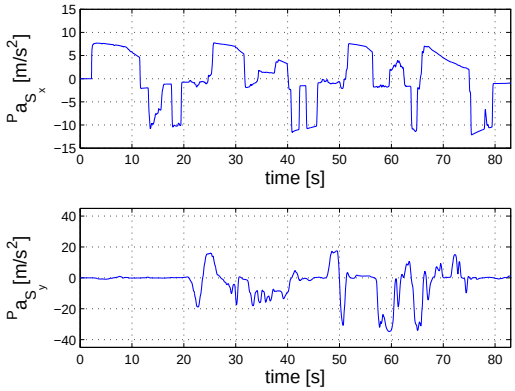
Figures 9(a–c) show the forward (top plots) and lateral (bottom plots) accelerations during the whole lap. In particular, Fig. 9(a) reports the accelerations ${}^{P}a$, direct output of the car dynamical model. We can notice that the car can achieve quite large accelerations: the peak values are 8 $[\mathrm{m/s^2}]$ as forward acceleration, -12 $[\mathrm{m/s^2}]$ during braking, and about 34 $[\mathrm{m/s^2}]$ during the roughest turns. Of course, no tilt coordination algorithm could reproduce such motions. After the scaling and saturation, we obtain the profiles shown in Fig. 9(b) which are more feasible for our robot, and must be compared with Fig. 9(c) where the actual specific force of the cabin is reported. Apart from some unavoidable noise, we can notice a substantial agreement among the two plots, thus confirming the good performance of our simulator also in reproducing such complex motions. We also encourage the reader to examine the attached video where the whole lap is shown from an external and onboard (camera-car) viewpoints. The robot response to the car motion, consequence of the user inputs, can then be fully appreciated.

V. Conclusions and Future Work

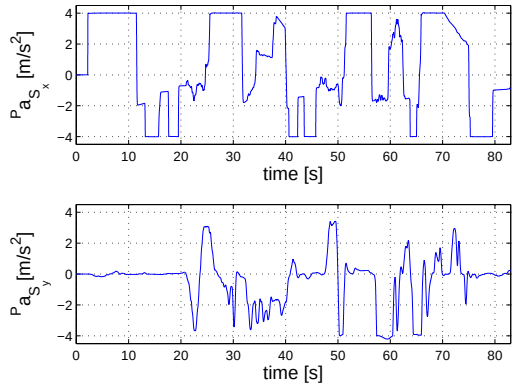
In this paper and its companion *Part I* we presented a complete architecture for realizing a 6-DOF closed-loop motion simulator based on an anthropomorphic serial manipulator — the CyberMotion Simulator. The main motivation behind this work lies in the fact that, compared to standard Stewart platforms, an industrial anthropomorphic manipulator offers a considerably larger motion envelope and higher dexterity, the possibility to realize any cabin posture within the workspace, and the ability to displace heavy loads (up to 500 $[\mathrm{kg}]$ in our case) with large accelerations and velocities. This clearly indicates a promising perspective for applications of motion simulation. However, in order to fully exploit a serial manipulator as motion platform, some specific issues must be addressed. In particular, the need of a special inverse kinematics scheme was tackled in *Part I*, while in this *Part II* we focused on designing a motion cueing algorithm tailored to the particular motion envelope of the robot (more cylindrical rather than rectangular).

We then presented experimental results collected during a closed-loop motion simulation on our testing scenario: driving a Formula-1 car along a lap on a virtual track. Although the proposed framework has a general validity, we chose to test it on this particular scenario also because of the challenges it involves, namely high and abrupt accelerations. The obtained performance was quite satisfactory, as can be also appreciated in the attached video, thus confirming the effectiveness of our approach.

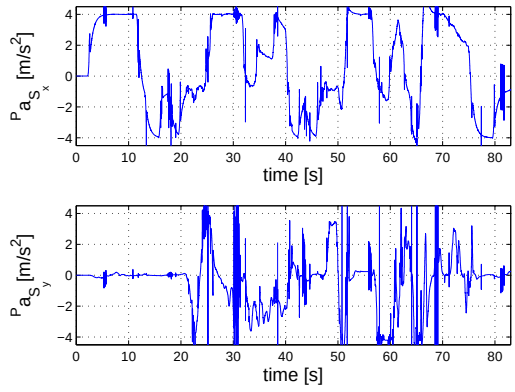
The work presented in this paper is, of course, still subject to many improvements, and should be considered as a solid and rigorous basis to lay down any future development. These will include, for instance, a careful tuning of the motion cueing algorithm to improve the simulation realism, also with the feedback of experienced pilots. More in general, our aim is to exploit the CyberMotion Simulator as a versatile tool for simulating a variety of vehicles, ranging from cars to airplanes, helicopters, and ships, for training and telepresence purposes. This will require a specific tuning, or even re-design, of the motion cueing algorithm case by case. The planned modification of the first robot joint (enabling continuous rotation), and the construction of a new closed



(a)



(b)



(c)

Fig. 9: Top: the forward (top) and lateral (bottom) linear accelerations $^P a$ of the car during the lap. Middle: the scaled accelerations $^P a_S$. Bottom: the actual specific force felt on the cabin as a consequence of the robot motion. Note the substantial agreement between plots (b) and (c). This confirms the good performance of our whole architecture, i.e., the motion cueing algorithm combined with the inverse kinematics of *Part I*

cabin will also contribute to improve the overall fidelity of the simulation.

ACKNOWLEDGMENTS

This research was supported the Max Planck Society and by the WCU (World Class University) program through the National Research Foundation of Korea funded by the Ministry of Education, Science and Technology (R31-2008-000-10008-0). The authors wish also to thank Michael Kerger and Dr. Harald Teufel for their intensive technical support.

REFERENCES

[1] P. Robuffo Giordano, C. Masone, J. Tesch, M. Breidt, L. Pollini, and H. H. Bülthoff, "A novel framework for closed-loop robotic motion simulation - Part I: Inverse kinematics design," *2010 IEEE Int. Conf. on Robotics and Automation*, 2010.

[2] Robocoaster, "www.robocoaster.com."

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

[4] P. R. Grant and L. D. Reid, "Motion washout filter tuning: Rules and requirements," *J. of Aircraft*, vol. 34, no. 2, pp. 145–151, 1997.

[5] S. F. Schmidt and B. Conrad, "Motion drive signals for piloted flight simulators," NASA, Tech. Rep. CR-1601, 1970.

[6] C.-I. Huang and L.-C. Fu, "Human vestibular based (HVB) senseless maneuver optimal washout filter design for VR-based motion simulator," *Proc. of the 2006 IEEE Int. Conf. on Systems, Man, and Cybernetics*, pp. 4451–4458, 2006.

[7] G. L. Greig, "Masking of motion cues by random motion: Comparison of human performance with a signal detection model," Univ. of Toronto, Tech. Rep. 313, 1988.

[8] R. V. Parrish, J. E. Dieudonne, R. L. Bowles, and D. J. Martin, "Coordinated adaptive washout for motion simulators," *J. of Aircraft*, vol. 12, no. 1, pp. 44–50, 1975.

[9] D. Ariel and R. Sivan, "False cue reduction in moving flight simulators," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 14, no. 4, pp. 665–671, 1984.

[10] M. A. Nahon, L. D. Reid, and J. Kirdeikist, "Adaptive simulator motion software with supervisory control," *J. of Guidance and Dynamics*, vol. 15, no. 2, pp. 376–383, 1992.

[11] R. Sivan, J. Ish-Shalom, and J. Huang, "An optimal approach to the design of moving flight simulators," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 12, no. 6, pp. 818–827, 1982.

[12] R. J. Telban, F. M. Cardullo, and J. A. Houck, "A nonlinear, human-centered approach to motion cueing with a neurocomputing solver," NASA, Tech. Rep. 2002-4692, 2002.

[13] L. D. Reid and M. A. Nahon, "Flight simulator motion-base drive algorithms: Part 1 – developing and testing the equations," Univ. of Toronto, Tech. Rep. 296, 1985.

[14] ——, "Flight simulator motion-base drive algorithms: Part 2 – selecting the system parameters," Univ. of Toronto, Tech. Rep. 307, 1986.

[15] ——, "Flight simulator motion-base drive algorithms: Part 3 – pilot evaluations," Univ. of Toronto, Tech. Rep. 319, 1986.

[16] L. Nehaoua, H. Arioui, H. Mohellebi, and S. Espie, "Restitution movement for a low cost driving simulator," *Proc. of the 2006 American Control Conference*, pp. 2599–2604, 2006.

[17] L. Nehaoua, H. Arioui, S. Espie, and H. Mohellebi, "Motion cueing algorithms for small driving simulator," *Proc. of the 2006 IEEE Int. Conf. on Robotics and Automation*, pp. 3189–3194, 2006.

[18] M. Wentink, W. Bles, R. Hosman, and M. Mayrhofer, "Design & evaluation of spherical washout algorithm for Desdemona simulator," *Proc. of AIAA Modeling and Simulation Technologies*, 2005.

[19] W. Bles and E. Groen, *The DESDEMONA Motion Facility: Applications for Space Research*. Springer, 2009.

[20] H. B. Pacejka and E. Bakker, "The magic formula tyre model," *Vehicle System Dynamics,*, vol. 21, no. 1, pp. 1–18, 1992.

[21] Monza racetrack, "www.monzanet.it."