A novel framework for closed-loop robotic motion simulation - Part I: inverse kinematics design

A Novel Framework for Closed-Loop Robotic Motion Simulation - Part I: Inverse Kinematics Design

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

***Abstract*— This paper considers the problem of realizing a 6-DOF closed-loop motion simulator by exploiting an anthropomorphic serial manipulator as motion platform. Contrary to standard Stewart platforms, an industrial anthropomorphic manipulator offers a considerably larger motion envelope and higher dexterity that let envisage it as a viable and superior alternative. Our work is divided in two papers. In this *Part I*, we discuss the main challenges in adopting a serial manipulator as motion platform, and thoroughly analyze one key issue: the design of a suitable inverse kinematics scheme for online motion reproduction. Experimental results are proposed to analyze the effectiveness of our approach. *Part II* [1] will address the design of a motion cueing algorithm tailored to the robot kinematics, and will provide an experimental evaluation on the chosen scenario: closed-loop simulation of a Formula 1 racing car.**

I. Introduction

Over the last decades, the realization of realistic immersions in virtual environments has been an active research field [2]. In this context, simulators of vehicle motion represent one major application of interaction with a virtual environment for training purposes [3], but also for educating and entertaining as reported in [4]. The big challenge, in this case, is to provide the user with all the sensory cues needed to reproduce a perfect illusion of *realism*. While controlling the simulated vehicle, the user should behave exactly as if he/she was interacting with the real one.

When simulating interaction with a vehicle, visual cues play a major role in achieving good realism, also thanks to the recent developments in 3D computer graphics. Vision alone, however, may not be enough to obtain a sufficient simulation fidelity so that additional cues, such as motion cues, are also often considered. In this case, the user experiences motion by means of a suitably actuated motion platform that tries to reproduce the linear accelerations and angular velocities that would have been felt on the vehicle.

Most existing designs of motion simulators are based on fully actuated hexapods (Stewart platforms). Although with motion capabilities in 6 degrees of freedom (DOF), these platforms suffer from their limited workspace and the impossibility to achieve large linear and angular displacements and rates because of the the closed chain nature of their actuation system. As a possible improvement to this design, the idea of exploiting industrial robot manipulators as motion platforms is drawing an increasing attention in the scientific community [5], [6], [7]. Indeed, a serial 6-DOF industrial manipulator offers higher dexterity, larger motion envelope, the possibility to realize any end-effector posture within the workspace, and the ability to displace heavy loads (up to 1000 $[\mathrm{kg}]$) with large accelerations and velocities. It is then possible to attach a cabin carrying the user to the robot end-effector. The large dexterity envelope of the robot motions allows moving the cabin along complex coordinated trajectories and reaching any attitude: the cabin could even be placed upside down, thereby achieving sustained negative vertical accelerations.

However, exploiting a serial manipulator as motion platform for closed-loop vehicle simulation involves also several challenges that will be extensively discussed in the rest of the paper and its companion *Part II*. In few words, there are two main issues: first, one must conceive a suitable inverse kinematics scheme able to cope with an unpredictable and arbitrary desired cabin motion, generated online as a function of the (unpredictable) user inputs to the simulated vehicle. Second, the design of washout filters, and in general of the whole *motion cueing* block, must be tailored to the specific motion envelope of a serial manipulator.

In this paper, we address the closed-loop control for the CyberMotion Simulator, a 6-DOF anthropomorphic robot arm based on the commercial KUKA Robocoaster [8]. In previous works, we already presented some results on simplified situations where only a small subset of robot DOF were used to generate motion [9], [10], [11]. Goal and main contribution of this paper and its companion *Part II* is to propose a general and rigorous framework to exploit the *full* 6-DOF of the robot to realize closed-loop motion simulations. Although our approach can be seamlessly applied to simulate any vehicle dynamics, we selected a Formula 1 racing car as testing scenario for obtaining an experimental evaluation.

This paper is structured as follows: after a description of the CyberMotion Simulator in Sect. II, Sect. III will illustrate the architecture of a typical motion simulation loop, and give more details on the challenges inherent in adopting a serial manipulator as motion platform. Section IV will then tackle the core issue of this *Part I*, i.e., the design of the inverse kinematics algorithm whose effectiveness will then be discussed in Sect. V via experimental results. Section VI will finally summarize the paper contributions, and indicate

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.

open points and future directions. Design of the motion cueing algorithm, details of the Formula 1 car model, and experimental evaluation of the whole architecture are given in *Part II*.

II. Description of the CyberMotion Simulator

The CyberMotion Simulator consists of a standard six-joint anthropomorphic robot arm, see Fig. 1(a). It is based on the commercial KUKA Robocoaster [8] (a modified KR-500 industrial robot with a 500 [Kg] payload), which was originally designed for use in amusement parks. A cabin with an onboard projection system is rigidly attached to the robot end-effector, and can be equipped with different kinds of input devices. In the case under consideration, we mounted a force-feedback steering wheel and pedals, see Fig. 1(b–c).

The robot is equipped with a low-level controller able to realize a given joint velocity command at a fast rate, so that one can disregard any dynamical issue and consider joint velocities as actual control inputs. In particular, by letting $q \in \mathbb{R}^6$ be the joint configuration vector, the low-level controller accepts joint increment commands $\Delta q_k = q(t_{k+1}) - q(t_k)$ as inputs, and returns the measured joint configuration $q(t_k)$ as output at a rate of $T_s = 0.012$ [s]. A structural delay of 0.04 [s] between the generation and actual execution of Δq_k is also present in this control loop. We did not consider the presence of this delay in the rest of the paper, and modeled the joint motion as a pure integrator

$$\dot{q} = u, \tag{1}$$

where $u \in \mathbb{R}^6$ represents the commanded joint velocity — our control input. The effects of the delay, and possible ways to compensate for it in the control design, will be addressed in future studies. Therefore, all the following higher-level control schemes will be built upon the ideal model (1).

The joint range is delimited by

$$q_{min} = [-130 \ -128 \ -30 \ -180 \ -58 \ -180]^T \quad [\text{deg}]$$

and

$$q_{max} = [130 \ -42 \ 78 \ 180 \ 58 \ 180]^T \quad [\text{deg}],$$

and there exist constraints on the maximum joint velocities $\dot{q}_{max} = [69 \ 57 \ 69 \ 76 \ 76 \ 120]$ [deg/s] and maximum joint acceleration $\ddot{q}_{max} = [98 \ 70 \ 128 \ 33 \ 95 \ 77]$ [deg/s^2].

III. Problem formulation

Figure 2 shows a block-scheme representation of a generic motion simulation loop. Starting from the left side, the first component is a software block that implements the dynamics of the simulated vehicle. This block accepts user's commands as inputs, and yields as outputs the rendered virtual scene and the relevant Cartesian motion profiles. These consist of those linear acceleration and angular velocities that would be experienced by a user onboard the vehicle.

The virtual scene is typically displayed 1:1 on a projection screen or in a head mounted display worn by the user. As for the motion data, some pre-filtering is usually required to make the 'ideal' vehicle motion compatible with the limited workspace of the chosen motion platform. Indeed, the motion range of a platform fixed to the ground is in general too limited to reproduce 1:1 *any* vehicle trajectory as, e.g., a car or an airplane linearly accelerating over a long time. Therefore, a so-called *motion cueing* block is inserted into the loop. Its purpose is to transform the input motion profile into a Cartesian trajectory compatible with the platform workspace, but still inducing a realistic motion perception onto the user. A classical example is the well-known tilt-coordination algorithm [12] that exploits the gravity vector in the user's body frame to simulate presence of a sustained linear acceleration. An overview of existing motion cueing algorithms, as well as all the details of our implementation, are given in *Part II*.

This filtered Cartesian trajectory must then, in our case, be fed to an *Inverse Kinematics* algorithm for its actual realization on the robot. A proper design of this component is a fundamental and nontrivial task, and is the main scope of this *Part I*. The problem lies in the fact that the classical structure of a motion cueing algorithm does not allow to take explicitly into account all the robot constraints expressed at the joint level. Therefore, one has to assume that the output of the motion cueing block can in general violate any (or all) of the robot constraints over time. Moreover, such an output trajectory is completely arbitrary (in terms of geometric path) and unknown in advance — it eventually depends on the unpredictable inputs of the user to the vehicle. Hence, the sought inverse kinematics must be able to realize at best and in real-time a Cartesian trajectory that 1) is geometrically unpredictable and unknown in advance, and 2) may violate any robot constraints over time.

The main contribution of this *Part I* is to propose a rigorous solution to this general problem. This solution will then be exploited in *Part II* to realize our closed-loop motion simulator.

IV. Design of the Inverse Kinematics

A. Preliminary definitions

With reference to Fig. 1(a) , 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.

Furthermore, let R_P be the rotation matrix from frame $\mathcal{F}_0$ 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 R_P. Let also $p = [x \ y \ z]^T \in \mathbb{R}^3$ represent the coordinates of O_P in $\mathcal{F}_0$. With these settings, we will call $J_{CE}(q) \in \mathbb{R}^{6\times 6}$ the Jacobian matrix representing the mapping from joint velocities $\dot{q}$ to Cartesian/Euler velocities

$$\begin{bmatrix} \dot{p} \\ \dot{\eta} \end{bmatrix} = J_{CE}(q)\dot{q}. \tag{2}$$

Derivation of $J_{CE}(q)$ follows from any standard robotics textbook, see, e.g., [13].

(a) (b) (c)

Fig. 1: A snapshot of the CyberMotion Simulator setup (a), and some details of the steering wheel and pedals mounted on the cabin (b–c)

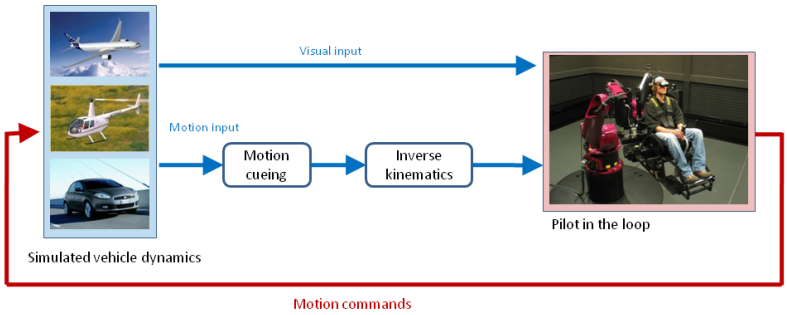


Fig. 2: A block-scheme representation of the system architecture

For reasons related to the design of the motion cueing algorithm (see *Part II*), we will 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 take $r = [\xi^T \;\; \eta^T]^T \in \mathrm{R}^6$ as *task variables* to be controlled. Being $\dot{R} = \dot{x}\cos\alpha + \dot{y}\sin\alpha$ and $\dot{\alpha} = (-\dot{x}\sin\alpha + \dot{y}\cos\alpha)/R$, one can easily obtain the mapping between $\dot{q}$ and $\dot{r}$

$$\dot{r} = \begin{bmatrix} \cos\alpha & \sin\alpha & \\ -\dfrac{\sin\alpha}{R} & \dfrac{\cos\alpha}{R} & 0 \\ 0 & & I \end{bmatrix} \begin{bmatrix} \dot{p} \\ \dot{\eta} \end{bmatrix} = T(\xi(q))J_{CE}(q)\dot{q} = J(q)\dot{q}. \tag{3}$$

Hereafter, $J(q)$ will be referred to as *task Jacobian* matrix. To simplify the notation, dependence on q will be dropped unless strictly necessary.

B. Differential Inverse Kinematics

In this subsection we will consider the problem of realizing a given reference task trajectory $r_d(t)$ output of the motion cueing algorithm. Since the robot directly accepts joint velocities $\dot{q}$ as inputs, the problem can be naturally formulated at the kinematic level, by exploiting the classical concepts of *kinematic inversion* and *kinematic control* see, e.g., [14]. The goal is to find a control law, or inversion scheme, $u = f(q,\, r_d,\, r)$ that guarantees realization of the task $r_d(t)$. As explained before, the chosen inversion scheme must also comply with all the robot constraints. Formally,

$$\begin{cases} \text{a)} & \forall i,\, \forall t \geq 0, \quad q_{i,min} \leq q_i(t) \leq q_{i,max} \\ \text{b)} & \forall i,\, \forall t \geq 0, \quad |\dot{q}_i(t)| \leq \dot{q}_{i,max} \\ \text{c)} & \forall i,\, \forall t \geq 0, \quad |\ddot{q}_i(t)| \leq \ddot{q}_{i,max} \end{cases}. \tag{4}$$

Furthermore, the inversion scheme should avoid singularities or, when not possible, soften their negative effect by passing as 'smoothly' as possible through them.

Many past works have addressed kinematic inversion by taking into account these issues. Broadly speaking, two main approaches do exist: online (local) and offline (global) algorithms. When the whole $r_d(t)$ is known in advance, or at least the geometric path has a known structure, offline methods can be used to modify the path or the associated timing law so as to cope with the constraints. This is the case, for instance, of the classical works of Slotine [15] and Shiller [16] where an offline optimization guarantees feasibility of the motion w.r.t. the actuator constraints.
Local methods are widespread for avoiding joint limits and singularities. Joint limits can be avoided by resorting to simple parabolic potential fields [13], or more sophisticated ones [17]. The same applies to the avoidance of singularities where suitable indexes, such as the manipulability measure [18], can be optimized. Mixed solutions, i.e., concurrent avoidance of joint limits and singularities, have also been explored [19], as well as strategies to soften the effect of singularities when passing close to them [14], [20].

In our case, offline methods are not a viable option because, as explained in Sect. III, $r_d(t)$ is both geometrically arbitrary and completely unknown in advance. Therefore, we must design an online solution able to realize the *best feasible motion*. Whenever realization of $r_d(t)$ is compatible

with all the constraints, then the robot should track it exactly. For instance, proximity to a joint limit should not degrade the robot motion until strictly necessary (as opposite to the effect of many potential field methods). On the other hand, whenever a constraint is violated, the robot should move as best as it can, i.e., by minimizing the Euclidean norm of the tracking error $\|e(t)\| = \|r(t) - r_d(t)\|$.

To the best of our knowledge, no previous work has rigorously considered the problem at hand in its completeness. The next sections will illustrate the solution adopted in this paper.

1) Singularities: singularities occur at those configurations $\bar{q}$ where the task Jacobian $J(\bar{q})$ loses rank, i.e., in our case when $\operatorname{rank} J(\bar{q}) < 6$. At a singular configuration it is impossible to generate task velocities in certain directions, namely those directions associated to the zero singular values of J. It is also well-known that, apart from the loss of mobility, any method based on the (pseudo-)inversion of J will become numerically unstable close to a singularity, yielding unbounded joint velocity commands as $q(t)$ approaches $\bar{q}$.

In this respect, a convenient way to deal with the occurrence of singularities is to resort to a Task Priority (TP) inversion scheme [20]. The idea is to divide the main task into several subtasks with different priorities. Away from singularities the whole task is realized whereas, whenever J is rank deficient, the lowest priority tasks are automatically relaxed while still correctly executing those with highest priority.

The peculiar nature of our problem (reproducing the correct motion perception on the user) led us to partition the main task as $r = [r_A^T \; r_B^T]^T$, with $r_A = [\rho \; \theta]^T \in \mathbb{R}^2$ being the higher priority subtask, and $r_B = [R \; \alpha \; z \; \psi]^T \in \mathbb{R}^4$ the lower priority one. In this way, close to singularities, realization of the correct orientation of gravity in $\mathcal{F}_P$ (responsible for simulating sustained cues) is favored compared to a correct execution of the other task variables (responsible for simulating onset cues).

Among the different variants of the TP strategy, (see [20], [21] for a survey), we chose to implement the following law

$$u = J_A^* w_A + (J_B(I - J_A^* J_A))^* (w_B - J_B J_A^* w_A). \quad (5)$$

Here, $J_A \in \mathbb{R}^{2\times 6}$ and $J_B \in \mathbb{R}^{4\times 6}$ are the subJacobians of J in (3) relative to the subtasks r_i, $i \in \{A, \; B\}$, the superscript * denotes a matrix generalized inverse, and $w_i = \dot{r}_{d_i} + K_i(r_{d_i} - r_i)$, with $K_i > 0$ being a positive definite gain matrix of suitable dimensions. The choice of vectors w_i follows the well-known CLIK paradigm [14] that ensures recovery of numerical drifts or tracking errors during the motion.

To avoid ill-conditioning when implementing (5), we resorted to a singularity-robust pseudoinversion based on numerical filtering [22] and implemented as

$$J^\circ = \sum_{i=1}^{s} \frac{\sigma_i}{\sigma_i^2 + \lambda_i^2} v_i u_i^T, \quad (6)$$

where $J = \sum_{i=1}^{s} \sigma_i u_i v_i^T$ is the singular value decomposition of matrix J, and individual λ_i affect each singular value. By

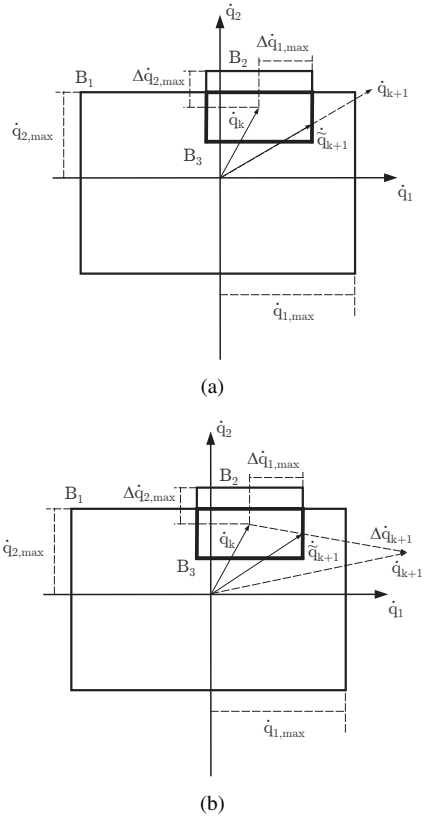


Fig. 3: Details of the scaling procedure. Top: if the supporting line of the desired joint velocity vector $\dot{q}_{k+1}$ intersects the set of feasible velocities B_3, a uniform scaling solves the problem without changing the direction of $\dot{q}_{k+1}$. Bottom: when this is not the case, a change in the direction of $\dot{q}_{k+1}$ is necessarily needed

choosing [23]

$$\lambda_i^2 = \begin{cases} 0 & \text{if } \; \sigma_i \geq \epsilon \\ \left(1 - \left(\frac{\sigma_i}{\epsilon}\right)^2\right) \lambda_{\max}^2 & \text{if } \; \sigma_i < \epsilon \end{cases} \quad (7)$$

where $\epsilon > 0$ defines the size of the singular region, and $\lambda_{\max}^2 > 0$ sets the maximum damping value, one obtains the remarkable result of introducing a tracking error only on the unfeasible task directions while keeping exact tracking on the feasible ones. Therefore, we chose to adopt (6–7) as generalized inverse to be used in the Task Priority law (5).

2) Maximum joint velocity and acceleration: the joint velocities computed from (5) have no guarantees to respect constraints b) and c) in (4). As discussed at the beginning of this section, there exist several offline methods able to modify or scale down the task reference motion $r_d(t)$ so as to make it compatible with the actuator constraints. Since, however, we assumed $r_d(t)$ to be unknown in advance and geometrically arbitrary, we chose to rely on a simpler online scaling of u with the aim of minimizing any Cartesian distortion. Figures 3(a–b) illustrate our approach in a two-dimensional case, the extension to six dimensions is straightforward. We will also reason in terms of discrete quantities sampled with a period of T_s [s] — the robot loop cycle, see Sect. II. Therefore, $\dot{q}_k = \dot{q}(t_k)$ will denote the joint velocity sent

to the robot at time t_k.

Let

$$B_1 = \{(\dot{q}_1,\, \dot{q}_2) \in \mathbb{R}^2 \,:\, |\dot{q}_1| \leq \dot{q}_{1,max},\, |\dot{q}_2| \leq \dot{q}_{2,max}\}$$

be the set of feasible joint velocities. Also, assume that a command $u_k = \dot{q}_k$ was sent to the robot, and $u_{k+1} = \dot{q}_{k+1}$ represents the next velocity command output of (5). From constraints c), we can easily determine a second set B_2 that represents the feasible velocities $\dot{q}_{k+1}$ attainable from $\dot{q}_k$ under the constraint of a maximum joint velocity increment $\Delta\dot{q}_{i,max} = T_s \cdot \ddot{q}_{i,max}$. Formally,

$$B_2 = \{(\dot{q}_1,\, \dot{q}_2) \in \mathbb{R}^2 \,:\, |\dot{q}_1 - \dot{q}_{k,1}| \leq \Delta\dot{q}_{1,max},\, |\dot{q}_2 - \dot{q}_{k,2}| \leq \Delta\dot{q}_{2,max}\}.$$

Let also $B_3 = B_1 \cap B_2$ (the thick box in the picture), and note that B_3 is always a convex set, in particular a rectangular box. If $\dot{q}_{k+1}$ lies inside B_3, no scaling is clearly needed since constraints b) and c) are met by construction.

Let us then consider the case in Fig. 3(a), in which $\dot{q}_{k+1}$ lies outside B_3 but its supporting line still intersects the region. By using standard tools of computational geometry [24], and owing to the convexity of B_3, it is easy to determine the intersection points between the supporting line of $\dot{q}_{k+1}$ and B_3. One can then apply a uniform scaling $\dot{\tilde{q}}_{k+1} = \sigma\dot{q}_{k+1}$, $\sigma > 0$, to obtain a new joint velocity $\dot{\tilde{q}}_{k+1}$ lying at the border of B_3 and possessing the *same direction* of the original $\dot{q}_{k+1}$. By doing so, the resulting task speed $\dot{\tilde{r}}_{k+1} = J\dot{\tilde{q}}_{k+1}$ will be uniformly scaled by a factor σ w.r.t. the original $\dot{r}_{k+1}$, but the task motion direction will stay the same. A straight line motion in task space will remain a straight line, but it will traveled along at a lower speed. In terms of motion perception, this will result in a lower motion cue but, at least, along the *correct* direction.

Now consider the last case depicted in Fig. 3(b). Here, no uniform scaling can solve the problem and a change in the direction of $\dot{q}_{k+1}$ is unavoidable. We implemented this change as follows: let $\Delta\dot{q}_{k+1} = \dot{q}_{k+1} - \dot{q}_k$ be the desired joint velocity increment. Like before, it is possible to determine the intersection between $\Delta\dot{q}_{k+1}$ and B_3, and uniformly scale the velocity increment $\Delta\dot{\tilde{q}}_{k+1} = \sigma\Delta\dot{q}_{k+1}$ so that the resulting joint velocity command $\dot{\tilde{q}}_{k+1} = \dot{q}_k + \Delta\dot{\tilde{q}}_{k+1}$ will lie at the border of B_3. With this strategy, the joint acceleration direction does not change but, clearly, the direction of the original velocity command $\dot{q}_{k+1}$ is lost. Therefore, a deformation in the task path will occur and, as a comparison with the previous cases, a straight line motion in task space will be transformed in a different path. We note that other choices are possible to perform the scaling in this case. For instance, one could pick the closest point of B_3 to the line supporting $\dot{q}_{k+1}$ (the lower-right vertex in the case of Fig. 3(b)), and take as $\dot{\tilde{q}}_{k+1}$ the vector joining the origin with such point. However, assessing the pro/cons of this and other possible choices in terms of motion perception quality is out of the scope of this paper, and will be evaluated in future studies.

Throughout the rest of the paper we will denote as $\tilde{u} = \dot{\tilde{q}}$ the joint velocity command computed from (5) and (when needed) scaled according to the strategy presented in this section.

3) Avoiding joint limits: the last problem of avoiding joint limits is crucial for a successful design of our inverse kinematics. To maximize the motion capabilities of the robot and, thus, its ability to reproduce a generic acceleration on the user, one should exploit the joint range at its maximum. Intuitively,

1) joint motion should not be altered by the proximity of a joint limit unless strictly necessary, i.e., until *the very last moment* before being too late to stop the joint without hitting the limit;
2) at the same time, each joint should deterministically stop within its range, without any possibility of overshooting the limit.

Most classical approaches based on repulsive potential fields are not a suitable choice. Usually, the distorting effects of the potential (from the point of view of the desired task motion) start well before the actual joint limits to ensure safety in all conditions. One can try to shape the potential so as to reduce its side-effects (see, e.g., [17]), but then it may become tricky to find the right set of parameters for deterministically halting the joint motion in any situation. Therefore, we adopted a different point of view and did not rely on potential field methods for avoiding joint limits.

The main insight is that each joint has a bounded acceleration limit $\ddot{q}_{i,max}$ by which it can stop. Hence, it is possible to rigorously determine the *very last moment* to stop a joint by exploiting the theory of bang-bang optimal control [25]. Consider the acceleration-level version of model (1) for the i-th joint $\ddot{q}_i = a_i$, where $a_i = \dot{u}_i$, $|a_i| \leq \ddot{q}_{i,max}$, is the i-th bounded acceleration command, and $|u_i| = |\dot{q}_i| \leq \dot{q}_{i,max}$ the bounded joint velocity taken as a state. Without dwelling on well-known details, in order to stop *at the very last moment* the motion of $q_i(t)$, one has to check for the intersection with the switching curves

$$\begin{cases} \gamma^- : \quad q_i = -\dfrac{\dot{q}_i^2}{2\ddot{q}_{i,max}} & \text{if } \dot{q}_i > 0, \\ \gamma^+ : \quad q_i = \dfrac{\dot{q}_i^2}{2\ddot{q}_{i,max}} & \text{if } \dot{q}_i < 0, \end{cases} \tag{8}$$

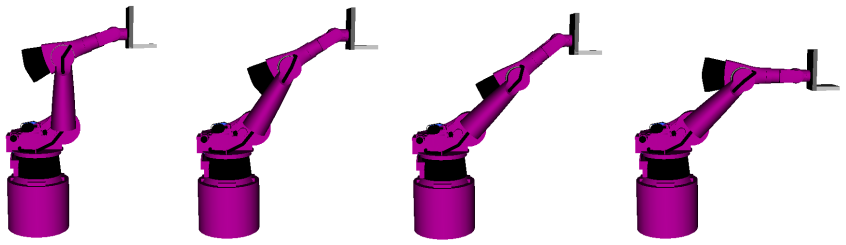
and then impose a maximum acceleration/deceleration command

$$a_i = \pm\ddot{q}_{i,max} \tag{9}$$

until the origin is reached. All this can be straightforwardly generalized for different set-points than the origin, i.e., in our case the joint minimum and maximum values $q_{i,min}$ and $q_{i,max}$.

Let us then call $\dot{u}_i^{BB} = f(q_i,\, \dot{q}_i)$ the (acceleration-level) bang-bang law able to stop the i-th joint in minimum-time[1], with the understanding that the velocity command

[1]Since implementation of the classical bang-bang controller has some drawbacks in practical cases, namely overshooting and chattering around the origin because of noise, delays, and discretization, we chose to implement a slightly modified version. Indeed, as proposed in [26], one can avoid these issues by suitably modifying the bang-bang controller behavior in a small neighborhood of the origin, and keeping the same bang-bang characteristics in large. We omit the details here for lack of space, see [26] and references therein for a complete analysis.



(a) (b) (c) (d)

Fig. 4: Snapshots of the robot motion during the experiment

u_i^{BB} actually sent to the robot is recovered by numerical integration. The idea is to apply, for each joint i,

C1: the normal control action $\tilde{u}_i$ from (5) when (8) does not hold (*unlocked joint*);

C2: the bang-bang control action $\dot{u}_i^{BB}$ as soon as (8) holds (*locked joint*).

It is clear that, contrarily to standard potential fields, this strategy will minimize the negative effects of approaching a joint limit. Any joint will be free to move unless strictly necessary, i.e., until the very last moment before being too late to stop. At the same time, each joint will systematically stop within its limit, thus ensuring a safe operation. In the last part of this section we will analyze the details of this strategy.

We will first focus on the issue of continuity of joint velocity during a switching. Assume that, at some time t_1, condition C2 is met for the i-th joint that will become *locked*. Since the control law $\dot{u}_i^{BB}$ acts at the acceleration level, it will automatically ensure continuity in the commanded joint velocity (although with a possible discontinuity in the joint acceleration). Hence, switching from C1 to C2 will preserve continuity.

Let us then consider the other switching direction from C2 to C1. We first note that, when a joint becomes locked, no control action $\tilde{u}_i$ can unlock it before coming to a full stop under $\dot{u}_i^{BB}$. Indeed, the velocity command $\tilde{u}_i$ is designed so that it will always meet the maximum acceleration constraints in (4), and thus it cannot change its sign before $\dot{u}_i^{BB}$ stops the joint at some $t_2 > t_1$, being $|\dot{\tilde{u}}_i| \leq |\dot{u}_i^{BB}|$ by construction during the halting phase.

After a locked joint stops its motion at time t_2, only two situations are possible for $t \geq t_2$. On one hand, $\tilde{u}_i(t)$ can keep the same sign of $\tilde{u}_i(t_1)$, implying that controller (5) is still trying to move the joint in the direction of the joint limit. In this case, condition C2 will hold and the joint will stay locked (it will not move). On the other hand, $\tilde{u}_i(t)$ can change sign so that the locked joint would move away from its limit. This will cause a switch from condition C2 to condition C1 but still preserving continuity in velocity. In fact, owing to the intrinsic continuity of controller (5), the change of sign of $\tilde{u}_i(t)$ will occur smoothly and the locked joint will start to gradually move from being at rest.

Having analyzed the continuity issues, this final part is devoted to assess how one can still preserve, as much as possible, realization of the desired task $r_d(t)$ with a possibly reduced set of unlocked joints. For simplicity, we will consider the case of a single locked joint, being the generalization straightforward. Let us then assume again that, at some t_1, the i-th joint switches to the locked condition. We will use the subscript $\mathcal{L}$ to indicate a property belonging to the set of locked joints, and subscript $\mathcal{U}$ for the unlocked joints. Therefore, in this case, it is $q_\mathcal{L} = q_i$ and $q_\mathcal{U} = [q_1 \dots q_{i-1}\ q_{i+1} \dots q_n]^T$, while $u_\mathcal{L}$ and $u_\mathcal{U}$ will indicate the generic locked and unlocked joint velocity commands.

Motion of $q_\mathcal{L}$ is governed by $u_\mathcal{L} = u_i^{BB}$, while $q_\mathcal{U}$ is driven by $u_\mathcal{U} = [\tilde{u}_1 \dots \tilde{u}_{i-1}\ \tilde{u}_{i+1} \dots \tilde{u}_n]^T$. Obviously, since the i-th component of $\tilde{u}$ is now replaced by $u_\mathcal{L}$, all the nice properties of controller (5) are in general lost. A subset of joints will still move accordingly to (5), but the effect of $u_\mathcal{L}$ will cause a degradation of performance. One can then try to redesign $u_\mathcal{U}$ so as to dampen the disrupting effects of $u_\mathcal{L}$.

Rearrange (3) as $\dot{r} = J_\mathcal{L} u_\mathcal{L} + J_\mathcal{U} u_\mathcal{U}$ and bring the contribution of $u_\mathcal{L}$ to the l.h.s. to obtain

$$\dot{r} - J_\mathcal{L} u_\mathcal{L} = \dot{r} - \dot{r}_\mathcal{L} = J_\mathcal{U} u_\mathcal{U}. \tag{10}$$

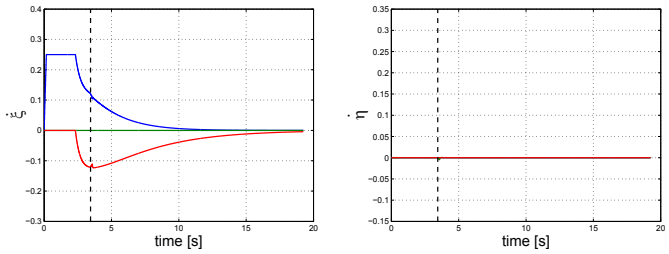
Here $\dot{r}_\mathcal{L}$ represents the distorting effects of $u_\mathcal{L}$ on the task evolution. One can still solve (10) for a given $\dot{r}_d(t)$ by applying the same TP strategy used in (5), with J_A and J_B now being the subtask Jacobian matrices of $J_\mathcal{U}$, and $w_i = \dot{r}_{d_i} + K_i(r_{d_i} - r_i) - \dot{r}_{\mathcal{L}_i}$. Inversion of (10) will then generate a new joint velocity command $u_\mathcal{U}^*$, in general different from the original $u_\mathcal{U}$ obtained from (5), that will try to realize task r_d (in a least-square sense) despite the presence of the spurious term $\dot{r}_\mathcal{L}$. In other words, while the locked joints $q_\mathcal{L}$ are stopping or at rest, the unlocked joints $q_\mathcal{U}$ will move at best to compensate for the effect of $\dot{r}_\mathcal{L}$.

Finally, we also note that at the switching time t_1 it is $u_\mathcal{U}^*(t_1) = u_\mathcal{U}(t_1)$ so that, again, continuity in the commanded joint velocity is ensured. This fact is a direct consequence of the continuous switch between $\tilde{u}_i$ and u_i^{BB} as proven before. Indeed, it is $u_\mathcal{L}(t_1) = u_i^{BB}(t_1) = \tilde{u}_i(t_1)$. Hence, solving (10) at t_1 in the locked case is equivalent to solve (5) in the full unlocked case, thus resulting in $u_\mathcal{U}^*(t_1) = u_\mathcal{U}(t_1)$.

V. Experimental results

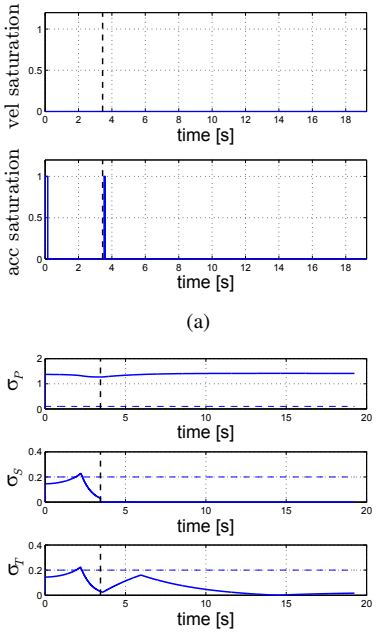
In this section, we will analyze the behavior of the proposed inverse kinematics through an illustrative example chosen to pinpoint all the relevant features of our approach. This and other motion examples are also included in the video attached to the paper. The interested reader can find additional experimental data of the algorithm in *Part II*.

In this example, we chose as desired task a horizontal linear trajectory with constant speed $\dot{\xi}_d = [0.25\ 0\ 0]^T$ [m/s]



(a) (b)

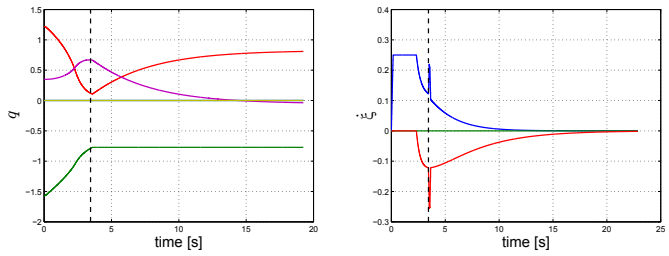
Fig. 5: Left: task velocities $\dot{\xi}$. Right: task velocities $\dot{\eta}$. Note the small perturbation in the task velocities at time $T_2 = 3.4$ [s], represented by a vertical dashed line, due to the action of the bang-bang controller



(a)

(b)

Fig. 6: Top: saturation action of the joint velocities and accelerations during the motion. Bottom: behavior of σ_P, σ_S and σ_T, i.e., the smallest singular value of J_A, $J_B(I - J_A^\circ J_A)$, and J, respectively



(a) (b)

Fig. 7: Left: behavior of $q(t)$ over time. At time $T_2 = 3.4$, joint 2 (the lower curve) stops its motion and remains locked for the whole motion. Right: task velocities $\dot{\xi}$ when not compensating for the spurious term $\dot{r}_{\mathcal{L}}$. Note the larger disturbance peaks compared to Fig. 5(a)

with an associated zero angular motion $\dot{\eta}_d = [0\ 0\ 0]^T$ [rad/s]. The robot starts from the configuration $q(t_0) = [0\ \ -90\ \ 70\ \ 0\ \ 20\ \ 0]^T$ [deg] corresponding to an initial task value $r(t_0) = [1.73\ \ 0\ \ 3.73\ \ 0\ \ 0\ \ 0]^T$ with the cabin being perfectly vertical ($\vec{Z}_P = \vec{Z}_0$). This reference motion, though artificially simple, is a good prototype of a partially unfeasible task: it starts with a discontinuity in acceleration that violates constraint c) in (4), and it travels along an infinite path that will eventually lead the robot to reach singularities and to hit joint limits.

In view of the forthcoming discussion, we will let σ_P be the smallest singular value of the primary task Jacobian J_A in (5). Similarly, σ_S will denote the smallest singular value of the secondary task 'coupling' matrix $J_B(I - J_A^\circ J_A)$, and σ_T the smallest singular value of the whole J. We also note that the inversion parameters in (7) were set to $\epsilon = \lambda_{max} = 0.2$.

The behavior of the robot during the task execution can be understood with the help of Figs. 4(a–d) and Figs. 5–7. From its starting position (Fig. 4(a)), the cabin travels along the desired horizontal straight line until realization of the secondary task starts conflicting with the primary task (algorithmic singularity, Fig. 4(b)). This event occurs at approximately time $T_1 = 2.3$ [s]. The cabin then keeps moving forward until joint 2 hits its limit and becomes locked (Fig. 4(c)). This second event occurs at approximately time $T_2 = 3.4$ [s] and is indicated by a dashed vertical line in Figs. 5–7. From this moment on, joint 2 stays locked and the cabin moves in a forward/downward direction until reaching full stop (Fig. 4(d)).

Let us analyze more in detail the motion in terms of the quantities reported in Figs. 5–7. First of all, note that during the first phase of the motion an initial acceleration saturation is present (Fig. 6(a)), and σ_S starts and remains below the threshold ϵ before T_1 (Fig. 6(b)). Nevertheless, the realized task velocity $\dot{R}$ correctly reaches the nominal value of 0.25 [m/s] at the end of the saturation phase and keeps this value until T_1 (Fig. 5(a)). This proves that the initial 'singularity' of $J_B(I - J_A^\circ J_A)$ (being $\sigma_S < \epsilon$) does not affect the correct task execution thanks to the numerical filtering strategy. Matrix $J_B(I - J_A^\circ J_A)$ is close to singularity, but the desired task direction belongs to the set of feasible ones.

After time T_1, the singularity of $J_B(I - J_A^\circ J_A)$ starts affecting the cabin motion yielding a decrease of task velocity $\dot{R}$ and the appearance of a spurious negative task velocity $\dot{z}$. The secondary task path results distorted (the cabin moves forward and downwards), but the primary task is still correctly executed as one can see from Fig. 5(b). Indeed, σ_P remains well above the threshold ϵ during the whole robot motion.

At time T_2, joint 2 becomes locked and the bang-bang controller $\dot{u}_2^{BB}$ stops its motion (Fig. 7(a)). This causes a perturbation in the task velocities (Fig. 5(a)) that disappears as soon as joint 2 is at rest. From this time on, only the 5 unlocked joints contribute to the cabin motion. In particular, the cabin keeps moving forwards and downwards until the robot reaches a configuration where no additional forward motion is possible and then stops. In other words, the least-

square solution of $r_d(t)$ yields a cabin motion that tries to keep a desired forward motion component by accepting undesired motions in other (secondary) task directions. Note that, as expected, the primary task is still correctly realized also during this phase, see Fig. 5(b).

Finally, we show in Fig. 7(b) the behavior of $r(t)$ obtained by neglecting the correction term $\dot{r}_{\mathcal{L}}$ in (10). The distorting effects of the bang-bang controller $\dot{u}_2^{BB}$ are more evident compared to Fig. 5(a), thus confirming the importance of compensating for the effect of $\dot{r}_{\mathcal{L}}$.

VI. Conclusions and future work

In this *Part I*, we considered the problem of designing an online inverse kinematics scheme able to reproduce as best as possible an arbitrary and unpredictable desired task motion by coping with the occurrence of singularities and several robot constraints, namely joint limits, maximum joint velocities and accelerations. Such an inverse kinematics scheme is a key component for realizing a 6-DOF closed-loop motion simulation based on our anthropomorphic robot manipulator: the CyberMotion Simulator. *Part II* will complete the picture by introducing a motion cueing algorithm tailored to the robot kinematics, and by presenting experimental results on the chosen test scenario, simulation of a Formula 1 car, to assess the performance of the whole architecture.

In the future, we plan to address several points to improve the proposed approach. First of all, we will explicitly consider the presence of the delay of 0.04 [s] in the control loop (see Sect. II) in order to compensate for it at the design level. Since the delay is known and constant, a Smith predictor [27] or similar techniques could provide a successful solution. We will also investigate variations to the scaling procedure presented in Sect.IV-B with the aim of identifying the best one in terms of motion perception induced to the user. More in general, we also plan to run an extensive validation campaign of the whole proposed approach (inverse kinematics and motion cueing) to improve the realism of the simulation, and clearly assess the capabilities and limitations of our CyberMotion Simulator.

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 II: Motion cueing design and experimental validation," *2010 IEEE Int. Conf. on Robotics and Automation*, 2010.

[2] L. J. Hettinger and M. W. Haas, *Virtual and Adaptive Environments: Applications, Implications, and Human Performance Issues*. LEA, 2009.

[3] J. Burki-Cohen, T. H. Go, and T. Longridge, "Flight simulator fidelity considerations for total air line pilot training and evaluation," *Proc. of the AIAA Modeling and Simulation Technologies Conference*, 2001.

[4] Robotic Vision - To 2020 and Beyond, *The Strategic Research Agenda for Robotics in Europe*. EUROP, 2009.

[5] J. Heindl, M. Otter, H. Hirschmüller, M. Frommberger, N. Sporer, F. Siegert, and H. Heinrich, "The robocoaster as simulation platform – Experiences from the "first authentic mars flight simulation"," *Proc. of the 1nd Motion Simulator Conference*, 2005.

[6] T. Bellmann, M. Otter, J. Heindl, and G. Hirzinger, "Real-time path planning for an interactive and industrial robot-based motion simulator," *Proc. of the 2nd Motion Simulator Conference*, 2007.

[7] L. Pollini, M. Innocenti, and A. Petrone, "Novel motion platform for flight simulators using an anthropomorphic robot," *J. of Aerospace Computing, Information, and Communication*, vol. 5, pp. 175–196, 2008.

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

[9] K. Beykirch, F. M. Nieuwenhuizen, H. J. Teufel, H.-G. G. Nusseck, and H. H. Bülthoff, "A roll-lateral helicopter side-step maneuver on the MPI motion simulator," *AHS 64th Annual Forum*, pp. 1–8, 2008.

[10] H.-G. G. Nusseck, H. J. Teufel, F. M. Nieuwenhuizen, and H. H. Bülthoff, "Learning system dynamics: Transfer of training in a helicopter hover simulator," *Proc. of the AIAA Modeling and Simulation Technologies Conference*, pp. 1–11, 2008.

[11] P. Pretto, H.-G. G. Nusseck, H. J. Teufel, and H. H. Bülthoff, "Effect of lateral motion on drivers' performance in the MPI motion simulator," *Proc. of the Driving Simulation Conference (DSC-Europe 2009)*, 2009.

[12] E. L. Groen and W. Bles, "How to use body tilt for the simulation of linear self motion," *J. of Vestibular Research*, vol. 14, no. 5, pp. 375–385, 2004.

[13] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer, 2009.

[14] P. Chiacchio, S. Chiaverini, L. Sciavicco, and B. Siciliano, "Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy," *Int. J. of Robotics Research*, vol. 10, no. 4, pp. 410–425, 1991.

[15] J.-J. E. Slotine and H. S. Yang, "Improving the efficiency of time-optimal path-following algorithms," *IEEE Trans. on Robotics and Automation*, vol. 5, no. 1, pp. 118–124, 1989.

[16] Z. Shiller, "On singular time-optimal control along specified paths," *IEEE Trans. on Robotics and Automation*, vol. 10, no. 4, pp. 561–566, 1994.

[17] T. F. Chan and R. V. Dubey, "A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators," *IEEE Trans. on Robotics and Automation*, vol. 11, no. 2, pp. 286–292, 1995.

[18] K. L. Doty, C. Melchiorri, E. M. Schwartz, and C. Bonivento, "Robot manipulability," *IEEE Trans. on Robotics and Automation*, vol. 11, no. 3, pp. 462–468, 1995.

[19] B. J. Nelson and P. K. Khosla, "Strategies for increasing the tracking region of an eye-in-hand system by singularity and joint limit avoidance," *Int. J. of Robotics Research*, vol. 14, no. 3, pp. 255–269, 1995.

[20] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators," *IEEE Trans. on Robotics and Automation*, vol. 13, no. 4, pp. 398–410, 1997.

[21] G. Antonelli, "Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems," *IEEE Trans. on Robotics*, vol. 25, no. 5, pp. 985–994, 2009.

[22] A. A. Maciejewski and C. A. Klein, "Numerical filtering for the operation of robotic manipulators through kinematically singular configurations," *J. of Robotic Systems*, vol. 5, no. 6, pp. 527–552, 1988.

[23] S. Chiaverini, O. Egeland, and R. K. Kanestrøm, "Achieving user-defined accuracy with damped least-squares inverse kinematics," *Proc. of the 5th Int. Conf. on Advanced Robotics*, pp. 672–677, 1991.

[24] M. De Berg, O. Cheong, M. Van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*. Springer, 2008.

[25] F. L. Lewis, *Optimal Control*. Wiley–Interscience, 1995.

[26] C. Guarino Lo Bianco and R. Zanasi, "Smooth profile generation for a tile printing machine," *IEEE Trans. on Industrial Electronics*, vol. 50, no. 3, pp. 471–477, 2003.

[27] A. C. Smith and K. Hashtrudi-Zaad, "Smith predictor type control architectures for time delayed teleoperation," *Int. J. of Robotics Research*, vol. 25, no. 8, pp. 797–818, 2006.