Virtual Reality (2006) 10: 11–23
DOI 10.1007/s10055-006-0026-6

ORIGINAL ARTICLE

Maurizio de Pascale · Allesandro Formaglio
Domenico Prattichizzo

A mobile platform for haptic grasping in large environments

Received: 24 January 2006 / Accepted: 31 March 2006 / Published online: 9 May 2006

Abstract This paper presents methodologies and technologies that are exploited to design and implement the *mobile haptic grasper* (MHG), i.e. an integrated system consisting of a mobile robot and two grounded haptic devices (HD) fixed on it. This system features two-point contact kinaesthetic interactions while guaranteeing full user's locomotion in large virtual environment. The workspace of haptic interaction is indefinitely extended, and this is extremely relevant for applications such as virtual grasping, where the global workspace is typically reduced with respect to those of the single-point contact devices. Regarding software architecture, we present the Haptik Library, an open source library developed at the University of Siena which allows to uniformly access HD, that has been used to implement the MHG software.

Keywords Haptics · Grasping · Mobile robotics

1 Introduction

The increasing demand for new and powerful virtual reality (VR) simulations feeds the research towards designing new hardware and software solutions. Nowadays, systems are available to display contact forces to human operators for simulating kinaesthetic interactions with virtual objects, mediated by several haptic tools ranging from simple styluses to finger thimbles. Typically these tools feature one-point contact interaction with virtual environments (Massie and Salisbury 1994; The Delta and the Omega haptic devices. ForceDimension—http://www.forcedimension.com).

M. de Pascale · A. Formaglio · D. Prattichizzo (✉)
Department of Information Engineering, University of Siena,
Via Roma 56, 53100 Siena, Italy
E-mail: prattichizzo@dii.unisi.it
Tel.: +39-0577-234609
Fax: +39-0577-234602
E-mail: mdepascale@dii.unisi.it
E-mail: formaglio@dii.unisi.it

Although research in multi-point contact interaction with virtual environments is still in its infancy, see (Barbagli et al. 2005) and therein references, the use of more than one single-point contact device has been shown to be quite satisfactory to handle two or three points of contact (Harwin and Melder 2002; De Pascale et al. 2005). In Harwin and Melder (2002) authors introduce the *friction cone algorithm*, which guarantees to haptically seize a virtual object and lift it up in a stable way. Improvements have been proposed by (Salisbury et al. 2004) where the *soft-finger contact model* has been modeled as an extension of existing contact models (Zilles 1995). Such contact model exploits rotational friction in order to counteract possible rotations of the object while it is grasped by two contact points.

Another emerging research in haptics is the extension of the workspace limitation that affects common grounded haptic devices (HD) such as, for instance the Phantom (Massie and Salisbury 1994) and the Omega (The Delta and the Omega haptic devices. ForceDimension—http://www.forcedimension.com). With these devices users are typically constrained to stand still within the workspace of the device and interact in a limited volume depending on the device kinematics and dimensions. Two possible approaches have been investigated in the literature to extend the workspace limitations of HD. The first one is referred to as *workspace drift control* and consists of transparently shifting the physical workspace of the device mapped inside the virtual world thus providing the illusion of haptic interaction with an unlimited workspace (Conti and Khatib 2005). In this case the user does not move and the illusion of large workspace is only for the visual and tactile feedback and does not involve the vestibular human system. The second approach is different in nature and is more complete in the sense that it allows the user to walk in the extended workspace. Examples of this approach are, for instance, the *Cobots*, i.e. devices that allow users to track a pre-computed path by displaying virtual constraints in the real environment

(Peshkin et al. 2005), and the *mobile haptic devices* (MHD), i.e. devices obtained combining desktop HD with mobile platforms (MP). The MHD approach was first introduced by Nitzsche et al. (2003), and Barbagli et al. (2005). Mobile haptic devices are the only class of devices allowing for true unlimited workspace, thanks to the integration of force feedback with full user locomotion.

This paper builds upon previous contributions and extends the MHD idea to haptic grasping. The proposed system referred to as mobile haptic grasper (MHG) is built by combining the mobile robot Nomad XR4000 by Nomadic Tech. Inc. to track user's locomotion and two Phantom Desktop HD to provide the user with a contact point for each hand as shown in Fig. 1.

The main contribution of this work is that of showing that it is possible to extend the workspace of haptic grasping devices by mobilizing the base where the two grounded HD are fixed. Note that this is paramount in grasping because it is well known that if two HD are combined to build a grasping system, then the workspace is typically reduced with respect to those of the single devices. A special attention is devoted to the control of the overall system that integrates results obtained for virtual grasping (De Pascale et al. 2005) with the control algorithm of MHD with a single-point contact (Formaglio et al. 2005; Formaglio and Prattichizzo 2005).

The MHG is a paradigmatic example of cooperative haptics. In fact, it is a system that allows the two hands of a single user to interact with the virtual environment but it could also be used by two users to cooperatively interact with the same object.

The second contribution of our work regards the architecture of the software driving the whole system that has been designed within the Haptik Library framework. This is an open source library developed at the University of Siena for accessing HD from different manufacturers with a powerful structure of runtime-loaded plugins (de Pascale et al. 2004; The Haptik Library Developer's Manual—http://www.haptiklibrary.org). The use of this library allows to easily and elegantly integrate existing software applications and makes MHD automatically available to both old and new applications through plugin architectures.

The paper is structured as follows. In Sect. 2 the basic principles of a MHG are introduced; in Sect. 3, MHG modeling and design process are shown and discussed in detail; Sect. 4 presents the software implementation of the MHG; in Sect. 5 experimental results are reported; conclusions and future work are discussed in Sect. 6.

2 The MHG

A general purpose MHD is designed to allow the interaction between user and objects displaced in large virtual environments. A MHD is made up of two main components: a mobile platform (MP) and an impedance-type haptic device (HD) grounded to the mobile platform. The haptic device provides kinesthetic interaction with objects displaced in the virtual environment while the MP is position controlled and tracks the end-effector trajectories generated by the operator (Barbagli et al. 2005; Formaglio et al. 2005). This allows to move the HD workspace towards the area of interest of the user, according to her/his real motion and consequently to enlarge indefinitely the MHD workspace. The MP motion control is such that the HD is kept in a desired optimum kinematic configuration, for instance that

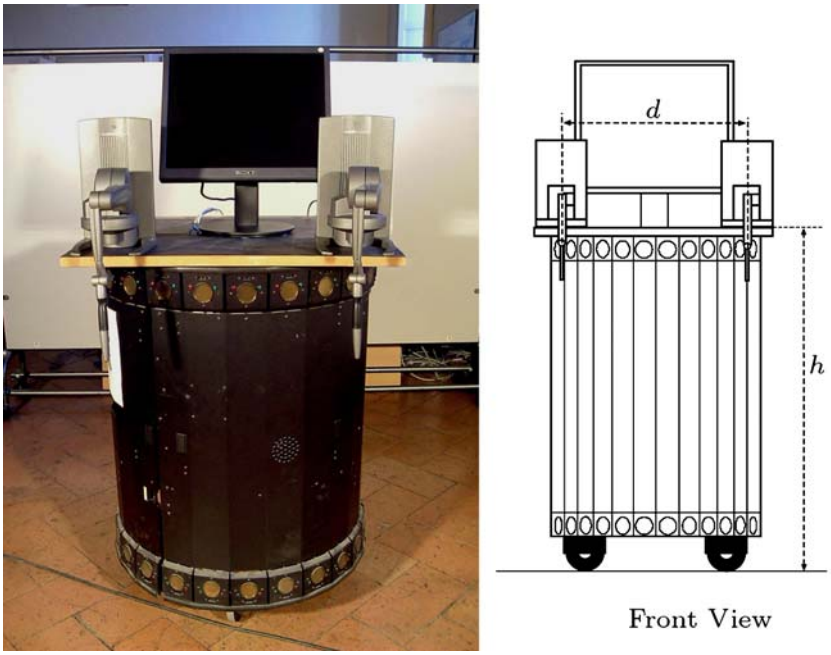

Fig. 1 The mobile haptic grasper (MHG) prototype

maximizing structural stiffness and such that the HD is kept far from reaching its workspace boundaries to avoid spurious forces felt by the operator that would compromise the overall sense of transparency.

This paper builds upon previous results (Barbagli et al. 2005; Formaglio et al. 2005) and extends the MHD with grasping capabilities. The mobile grasping system presented in this work features two points of contacts, thus the basic control principles must be partially reviewed. As regards the motion control strategy, for example, the user can independently move both end-effectors, thus in most cases the MP trajectory which prevents the devices to reach their workspace boundaries may be ambiguous. Before introducing the basic idea which extends MHD control principles to the MHG, let us introduce the main reference frames for the mobile system.

Refer to Fig. 2 and consider the world reference frame Σ_W, a mobile reference frame Σ_M fixed to the mobile robot, and finally the frames $\Sigma_{H,\,L}$ and $\Sigma_{H,\,R}$, fixed to the left and right HD, respectively. Note that the origins of the haptic device reference frames are centered at the centers of their workspace.

2.1 Key idea

Consider Fig. 2 and let $x_{H,\,L}$ and $x_{H,\,R}$ be the left and right end-effectors displacements due to motions of the user hands. The mobile platform motion control problem can be synthesized in moving the mobile base, and consequently the HD reference frames, of a displacement x_C able to reduce distance of the two end-effectors from their workspace centers. The MP motion control problem can be stated as a minimization problem over the vector x_C whose solution is the arithmetic mean of the two displacements $x_{H,\,L}$ and $x_{H,\,R}$ (Luenberger 1969). An equivalent graphical formulation of the control problem is that of considering the reference frame Σ_E whose center is the arithmetic mean of the two HD workspace centers and to control the mobile base to move the center O_E of Σ_E towards the mean of $x_{H,\,L}$ and $x_{H,\,R}$ expressed in Σ_E. In other terms, while the operator handles both end-effectors, the mobile platform point O_E is controlled to track the trajectory performed by the mean point between the two end-effectors. The main advantages of the proposed control strategy are that the distance of the two HD end-effectors fixed to the mobile platform from their respective workspace centers is minimized both in free motion, i.e. while no forces are fed back to the user's hands, and while grasping a virtual object. Consequently, the possibility of reaching the grounded HD workspace boundaries reduces.

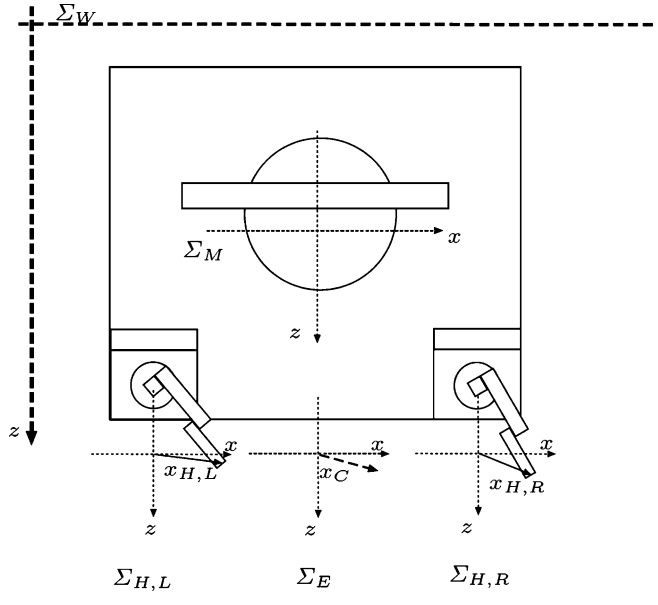

Fig. 2 The main reference frames of the MHG

2.2 Virtual grasping rendering

The proposed hardware architecture allows the user to independently operate with both hands with different objects belonging to the same virtual scenario, nevertheless the type of multi-point contact interaction we are mainly interested to simulate is grasping of virtual large-object, in which the object is seized between both hands. From the point of view of grasp modeling, this kind of two-point contact interaction is analog to the pinch grasp (Johansson and Cole 1994), in which the two contact forces splits into two components: the self-balanced grip forces F_G and the load forces F_L as shown in Fig. 3.

The criteria for a stable pinch grasp are as follows: the sum of the load forces must be equal to the object's weight, the sum of the grip forces must be equal to zero and the contacts must not be slippery, i.e. the contact force must lie within the friction cone at the virtual contact point. The grip forces are directly computed

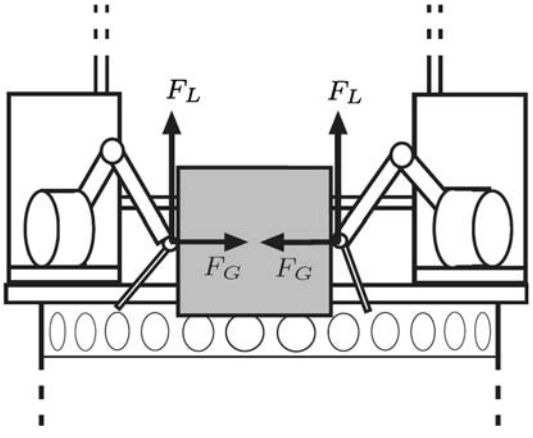

Fig. 3 Contact forces splits into two components: the self-balanced grip forces F_G and the load forces F_L

from the normal reaction applied on the HD by the force rendering algorithm at the contact points. In order to compute the load force, a non-linear local model has been extended to include tangential frictions.

Among several models for tangential friction available in the literature, the one we implemented originates from the model proposed by Zilles (1995) which has proved to be simple and effective. This model is briefly reviewed for the reader convenience. The contact is not slippery and consequently only static tangential friction is taken into account at the contact point. This leads to a model with a single state corresponding to a static friction (*stiction*) condition (De Pascale et al. 2005). In this state, the force exerted by the user lies within the friction cone thus the contact point cannot slip (see Fig. 4, side *A*). This is modeled as follows: let x_A be the current haptic end-effector avatar position, and let x_S be the current stiction point, i.e. the position in which the avatar is fixed by surface tangential friction (note that the same analysis apply for left and right HDs, $x_A = x_{H,L}$ or $x_A = x_{H,R}$.

The tangential force component F_T at the contact is obtained as:

$$\begin{cases} F_T = k\Delta x_T \\ \Delta x_T = (\Delta x - (\Delta x \cdot \hat{n}) \cdot \hat{n}) \\ \Delta x = (x_S - x_A), \end{cases} \quad (1)$$

where k is the local model stiffness and $\hat{n}$ is the surface normal unit vector in x_S.

The algorithm described so far guarantees to grasp and lift up a virtual object through two contact points, but cannot avoid object rotations about the grasp axis, i.e. the line joining the two contact points (Fig. 5). In practice, the human hands through fingertips are able to resist torsional moments about contact normals. Ability of counteracting rotations of object is modeled by the rotational friction and in particular by the soft-finger contact model first introduced by Mason and Salisbury (1985).

The soft-finger model for virtual contacts implemented in the MHG is that proposed by Salisbury et al. (2004). This model takes into account angles of rotation about the contact normal if there is an interaction with only one point or otherwise about the grasp axis if the interaction is with two contact points.

Again, the model features a single state corresponding to static friction. In this state, the surface rotational friction at contact points exceeds the torsional moment which acts on the object about the grasp axis, thus the object cannot rotate. This is modeled as follows: during the contact a fourth variable α_A is associated to the end-effector avatar to represent its relative orientation with respect to the virtual surface. Let α_S be the current *stiction angle*, i.e. the angular position in which the avatar is fixed due to surface rotational friction (such angles are considered to be counterclockwise measured about the grasp axis).

The torsional moment τ acting on the grasped object is obtained as:

$$\begin{cases} \tau = k_\tau \Delta\alpha \\ \Delta\alpha = \alpha_S - \alpha_A, \end{cases} \quad (2)$$

where k_τ is the haptic servo-loop gain. For further details about implementing soft-finger contact models in haptics research, the reader is referred to Salisbury et al. (2004).

3 Modeling the device

The mobile grasping system this paper deals with is built by combining the mobile robot Nomad XR4000 to track user's motion and two Phantom Desktop HD to provide the user with a contact point for each hand, (Fig. 1). The Nomad is a holonomic platform, featuring three planar degrees of freedom, i.e. it can perform translational and rotational motion. The two Phantoms are impedance-

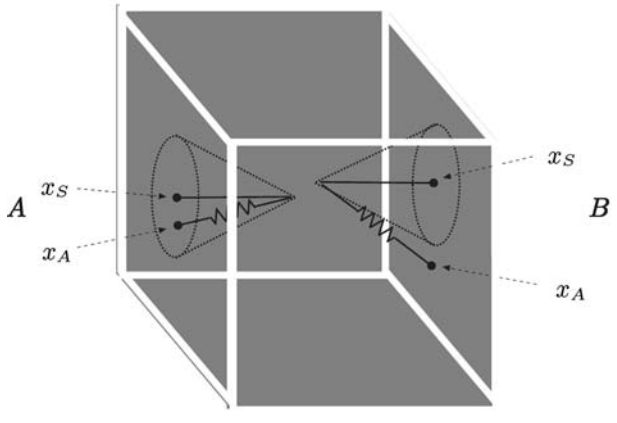

Fig. 4 A schematic representation of the friction cones at the contact points. Side *A*: the tangential force lies within the cone and the contact point cannot slip. Side *B*: the tangential force lies outside the friction cone and the contact point slips

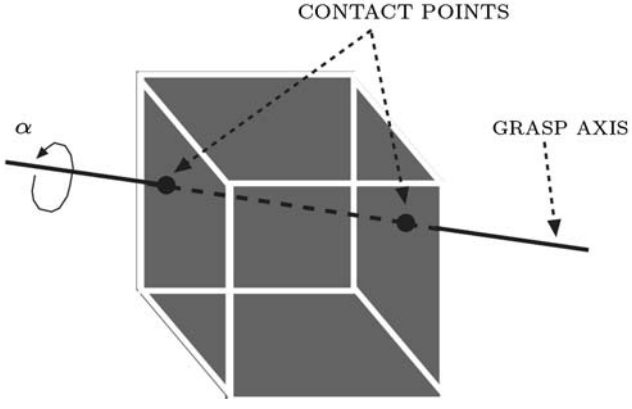

Fig. 5 The grasp axis, i.e. the line joining the two contact points

type HD with three actuated degrees of freedom, hence the MHG is a kinematically redundant system.

For the sake of simplicity, we will consider the motion of the mobile base and of the HD end-effectors only along the x-axis (Fig. 2). Dynamic model is derived similarly to the model for a single contact point (Nitzsche et al. 2003; Formaglio et al. 2005). The x-axis dynamics model for the two contact points mobile haptic system is shown in Fig. 6.

As regards the left haptic device part (Fig. 6, top), $x_{\mathrm{W,\,L}}$ and $x_{\mathrm{H,\,L}}$ represent the position of HD end-effector with respect to Σ_{W} and $\Sigma_{\mathrm{H,\,L}}$, respectively; $F_{\mathrm{O,\,L}}$ is the operator's left hand force, while $F_{\mathrm{H,\,L}}$ is the reaction force of the left device, thus yielding the total force $F_{\mathrm{T,\,L}}$ acting on end-effector; $D(s)$ models the end-effector reflected inertia felt by the operator; ZOH is a zero holder hold and finally a collision detection block is considered. The impedance model $Z_{\mathrm{E}}(z)$ for virtual objects can be chosen as a discrete spring-damper system

$$Z_{\mathrm{E}}(z) = D_{\mathrm{h}}\frac{z-1}{zT_{\mathrm{H}}} + P_{\mathrm{h}},$$

where T_{H} is the haptic servo-loop period, and P_{h} and D_{h} are the spring and damping coefficients. Note that the impedance model has been chosen equal for both interaction points. The right haptic device model (Fig. 6, bottom part) can be similarly derived. The signals $x_{\mathrm{H,\,L}}$ and $x_{\mathrm{H,\,R}}$ are employed to determine the system motion (Fig. 6, center). $C(s)$ is the transfer function matrix of the mobile platform control algorithm; x_C is the position commanded to the MP; $H(s)$ is a dynamic model for the MP; $F(z)$ is the discrete-time transfer function representing the MP localization algorithm with sampling time T_{H}, and finally x_{M} is the MP position with respect to the world frame Σ_{W}. To simplify the notation, we can assume without loss of generality that x_{M} represents the displacement of O_{E} referred to Σ_{W}. The role of controller $C(s)$, transfer function $F(z)$ and signal $\hat{x}_{\mathrm{M}}$ will be discussed in the following.

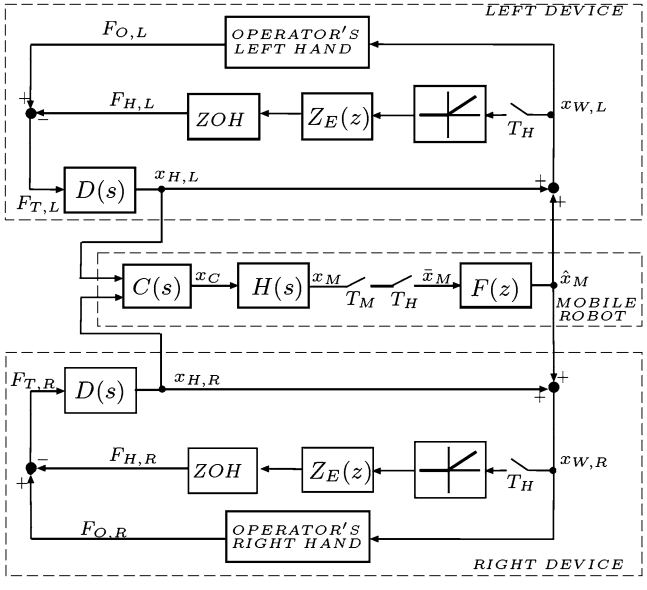

Fig. 6 Dynamic model of the MHG

Dynamics of the mobile platform are modeled through a second-order mass-spring-damper system

$$H(s) = \frac{K_{\mathrm{m}}}{s(M_{\mathrm{m}}s + B_{\mathrm{m}})}, \tag{3}$$

where K_{m} is the spring stiffness, B_{m} is the damping coefficient and M_{m} is the mass of the platform (Nitzsche et al. 2003; Formaglio et al. 2005).

While this is far from being a perfect dynamical model of a mobile robot and cannot account for nonlinear effects that are present in real world, it has been chosen for two main reasons: it is simple and is characterized by a small set of parameters that can easily be interpreted; it allows to analytically characterize the dynamical performance of the mobile haptic interface (Formaglio et al. 2005) (see Sect. 3.3).

3.1 Mobile platform position smoothing

An important issue in mobile haptics is the influence of the position controlled part, i.e. the motion control of the mobile platform, on collision detection and reaction force computation, that for MHD rely on global end-effector displacements $x_{\mathrm{W,\,L}}$ and $x_{\mathrm{W,\,R}}$ (Formaglio and Prattichizzo 2005). These signals are retrieved by composing local measurements $x_{\mathrm{E,\,L}}$ and $x_{\mathrm{E,\,R}}$ of the two HD expressed in Σ_{E} with the position x_{M} of the mobile platform

$$\begin{cases} x_{\mathrm{W,L}} = Rx_{\mathrm{E,L}} + x_{\mathrm{M}} \\ x_{\mathrm{W,R}} = Rx_{\mathrm{E,R}} + x_{\mathrm{M}}, \end{cases} \tag{4}$$

where R is the rotation of Σ_{E} with respect to the world frame.

As far as dynamics is concerned, by Nitzsche et al. (2003) it has been proven that the impedance perceived by the user, under certain conditions, does not depend on the mobile platform dynamics. Actually, the required conditions consist of exact position measurements of the HD and of exact localization of the mobile platform, and low haptic end-effector reflected inertia. Generally such conditions are satisfied except for the accuracy of the mobile platform localization x_{M} that suffers from rough quantization and slow sampling rate with respect to the haptic servo-loop.

To refine the estimation of MP displacement we propose to use a smoothing algorithm (Formaglio and Prattichizzo 2005). The basic idea is the following: first x_{M}, originally featuring sampling time T_{M}, is oversampled at the same rate as the haptic servo-loop, thus obtaining $\bar{x}_{\mathrm{M}}$ sampled at time T_{H} (Fig. 6). Secondly, the mobile platform displacement is estimated using a discrete-time signal $\hat{x}_{\mathrm{M}}$ defined as:

$$\begin{cases} \hat{x}_{\mathrm{M}}((i+1)T_{\mathrm{H}}) = \hat{x}_{\mathrm{M}}(iT_{\mathrm{H}}) + K\Delta x(iT_{\mathrm{H}}) \\ \Delta x(iT_{\mathrm{H}}) = \bar{x}_{\mathrm{M}}(iT_{\mathrm{H}}) - \hat{x}_{\mathrm{M}}(iT_{\mathrm{H}}) \\ \hat{x}_{\mathrm{M}}(0) = 0, \end{cases} \tag{5}$$

For a given $\bar{x}_{\mathrm{M}}$, updated with sampling period T_{M} and kept constant at the inter-samples with sampling period T_{H}, the variable $\hat{x}_{\mathrm{M}}$ is updated to reduce error Δx. The proportional gain K determines the velocity whereby Δx reduces. To analytically design the parameter K, we set the following specifications:

1. $\hat{x}_{\mathrm{M}}$ must recover the current MP odometry measure within m steps, where $m = \lfloor T_{\mathrm{M}}/T_{\mathrm{H}} \rfloor$;
2. from dynamics (5) $\hat{x}_{\mathrm{M}}$ can only asymptotically tend to $\bar{x}_{\mathrm{M}}$, hence we set K such that $\hat{x}_{\mathrm{M}}(m) = \delta_{\mathrm{H}}$ i.e. such that after m steps the difference between $\hat{x}_{\mathrm{M}}$ and $\bar{x}_{\mathrm{M}}$ is equal to the HD quantization error δ_{H}.

Our worst case analysis shown that the maximum value for the error Δx is $V_{\mathrm{M}}\ T_{\mathrm{M}}$, where V_{M} is the maximum allowed translational velocity for the mobile platform. For the sake of clarity, let us suppose that at $t=0$ the current measure $\bar{x}_{\mathrm{M}}(0) = 0$ and the initial condition $\hat{x}_{\mathrm{M}}(0) = V_{\mathrm{M}}T_{\mathrm{M}}$. The free-state evolution of the discrete-time system (5) can be studied with respect to K. By straightforward computation, from specifications 1 and 2 we have

$$\hat{x}_{\mathrm{M}}(m) = (1-K)^m V_{\mathrm{M}}T_{\mathrm{M}} = \delta_{\mathrm{H}}$$

that yields to

$$K = 1 - \left(\frac{\delta_{\mathrm{H}}}{V_{\mathrm{M}}T_{\mathrm{M}}}\right)^{\frac{1}{m}}. \tag{6}$$

The gain K computed according to (6) guarantees that $\hat{x}_{\mathrm{M}}$ estimates the MP displacement with a bounded approximation error

$$\delta_{\mathrm{H}} \leq \Delta x \leq V_{\mathrm{M}}T_{\mathrm{M}}. \tag{7}$$

The lower bound represents the unrecoverable estimation error introduced by this algorithm, which is actually comparable to the quantization error of the haptic device. As regards the upper bound, typical values of V_{M} and T_{M} (see for example Table 1) yield to reasonable values for the maximum error. The smoothing algorithm is used to compute the position of the HD end-effectors in the world frame as follows:

$$\begin{cases} x_{\mathrm{W,L}} = Rx_{\mathrm{E,L}} + \hat{x}_{\mathrm{M}} \\ x_{\mathrm{W,R}} = Rx_{\mathrm{E,R}} + \hat{x}_{\mathrm{M}}. \end{cases} \tag{8}$$

The transfer function $F(z)$ in Fig. 6 is obtained analyzing the block scheme of Fig. 7, and it takes on the following form:

$$F(z) = \frac{1 - \left(\frac{\delta_{\mathrm{H}}}{V_{\mathrm{M}}T_{\mathrm{M}}}\right)^{1/m}}{z - \left(\frac{\delta_{\mathrm{H}}}{V_{\mathrm{M}}T_{\mathrm{M}}}\right)^{1/m}}. \tag{9}$$

The main advantage of this strategy is that the effects of mobile platform localization on impedance rendering are remarkably reduced (Formaglio and Prattichizzo 2005), and also in the case of two-contact-points mobile haptic system the realism of virtual simulation is preserved.

3.2 Motion control

The HD and the mobile robot are respectively force controlled and position controlled systems, thus a MHG features both force controlled and position controlled dynamics. As regards the position controlled part, the target is extending the workspace of virtual environments thanks to the motion of the mobile robot, which continuously tracks operator's locomotion as pointed out in Sect. 2.1. At each time instant the MP is commanded to track the mean point between end-effectors displacements (Fig. 8)

$$x_{\mathrm{H}} = \frac{x_{\mathrm{E,L}} + x_{\mathrm{E,R}}}{2}. \tag{10}$$

In other terms, platform motion is controlled to get x_{H} going to zero and the signal x_{H} can be seen as the mobile platform tracking error.

3.3 Object depending performance

Mobile robots are generally characterized by slow dynamics, then delays may be present when attempting to track the human operator motion. As a consequence the HD end-effector may reach the boundary of its workspace before the mobile platform can recover the lag, which in turn can create spurious forces and ultimately cause a total loss of transparency for the MHD. In our previous works (Barbagli et al. 2005; Formaglio et al. 2005) we analyzed this aspect for single-contact-point MHD, determining that dynamical performances of a MHD depend on parameters of the mobile robot,

Table 1 Hardware and software parameters for the experimental application

Mobile Platform		Controller	
T_{M}	0.01 s	k_{p}	1.2
V_{M}	0.4 m/s	k_{d}	0.2
K_{m}	8,000 N/m	d	0.4 m
B_{m}	3,000 Ns/m	$w_{\mathrm{E}} = w$	0.15 m
M_{m}	160 kg	l	0.4 m
Haptic devices		**Smoother**	
T_{H}	0.001 s	K	0.45
δ_{H}	0.01 mm	m	0.5 kg

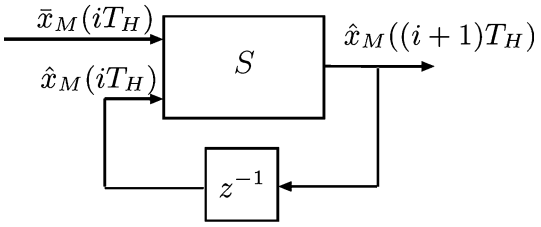

Fig. 7 Block scheme of the smoother

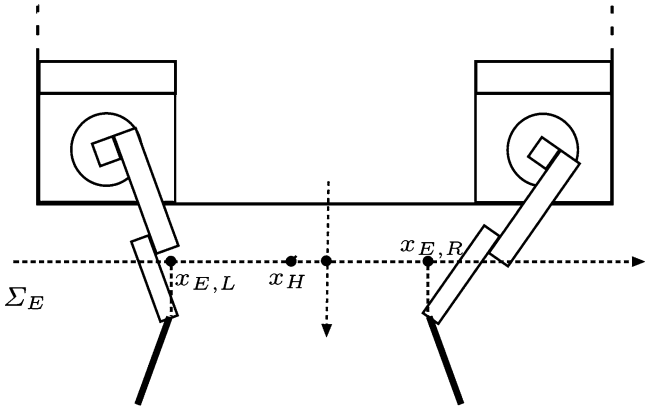

Fig. 8 A typical configuration of the end-effectors: the arithmetic mean x_{H} between $x_{\mathrm{E,\,L}}$ and $x_{\mathrm{E,\,R}}$ represents the mobile platform tracking error, and it determines an unique trajectory to be tracked by the MP

on the motion control law and also on dimensions of the haptic device original workspace (Barbagli et al. 2005; Formaglio et al. 2005).

In this section we extend performance analysis to the case of a two-contact-points mobile haptic system and in particular we focus our investigation on the case where the user moves in the large workspace while grasping a virtual object via the two HD.

Let d be the distance between the workspace centers of the two HD, l the size of the virtual object and w the original HD workspace width (Fig. 9).

In such a case, the equivalent workspace w_{E} reported in Fig. 9 (shaded regions) represents the space in which both HD can correctly render reaction forces of the grasped object without reaching their original workspace boundaries. Note that being the analysis limited to the x-axis, w_{E} can be easily determined by translating the object at the two limit cases marked by a and b in Fig. 9, where one of the two devices has reached its workspace boundary. The analytic relationship between w_{E} and the parameters depicted in Fig. 9 can be easily written as

$$w_{\mathrm{E}} = l + w - d. \tag{11}$$

The performance analysis problem can be equivalently recasted as a performance analysis problem for a MHD with a single-contact-point as in Fig. 10 where the mean point x_{H} corresponds to the displacement of this equivalent contact point from the center of the equivalent workspace whose length is w_{E}.

3.4 Controller design

Relying on the equivalent model, it is possible to exploit experimental criteria introduced by Formaglio et al. (2005) in order to design a proportional-derivative controller $C(s)$ in Fig. 6 which improves dynamical performance of the system from a transparency viewpoint. This design methodology is referred to particular sets of position input signals for the MHD, such as step inputs, ramp inputs and sinusoidal inputs. Although for each of them there exists a different design procedure, their common feature consists of determining k_p, k_d, i.e. the proportional and derivative gains of $C(s)$, respectively, which guarantee that the tracking error always holds within the equivalent workspace bound for every input signal of the related set, i.e.

$$|x_{\mathrm{H}}(t)| < \frac{w_{\mathrm{E}}}{2}, \quad \forall t. \tag{12}$$

In particular, for the MHG we consider position ramp input signals, which better resemble the typical movements that an user can perform to move the grasped object inside a large virtual environment. Given V_{M} the maximum allowed velocity for the mobile platform, as performance specification we desire the MHG to transparently track all input ramps featuring a slope $V \leq V_{\mathrm{M}}$. We recall that here for transparency we mean that the boundary of the HD workspace are not reached during the MHG motion.

According to results introduced by Formaglio et al. (2005), first the proportional gain should be chosen s.t.

$$k_{\mathrm{p}} \geq \frac{B_{\mathrm{m}}}{K_{\mathrm{m}} w_{\mathrm{E}}} V_{\mathrm{M}} = \frac{B_{\mathrm{m}}}{K_{\mathrm{m}}(l + w - d)} V_{\mathrm{M}}, \tag{13}$$

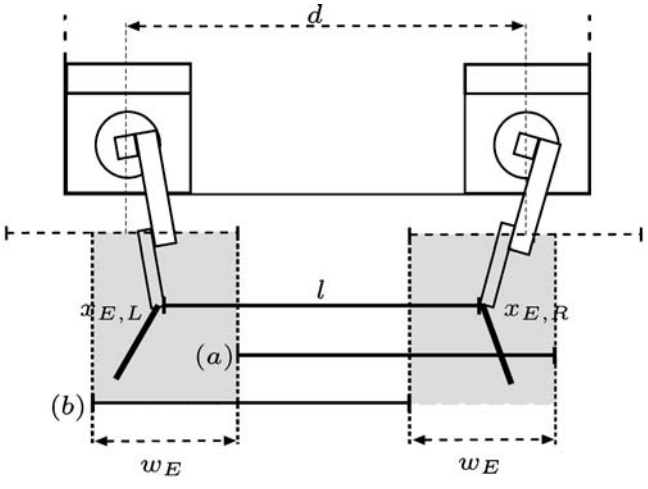

Fig. 9 The equivalent workspace (*shaded regions*) for a MHG depends on structural and software parameters

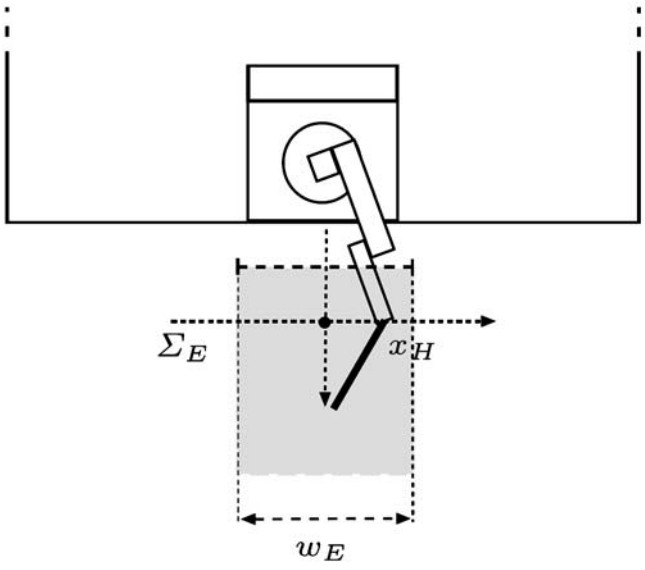

Fig. 10 The equivalent single-contact-point configuration which represents the two-contact-point system while grasping a virtual object

where B_m and K_m are the mobile platform parameters which appear in relationship (3). The condition (13) guarantees that at the steady state, (12) holds, i.e.

$$|x_H(\infty)| < \frac{w_E}{2}. \tag{14}$$

Now a proper value of the derivative gain k_d should be selected to prevent the tracking error x_H from exhibiting overshoots, which may result in violating the (12) at a certain time instant $t < \infty$. To this purpose, k_d can be graphically chosen in the shaded region depicted in Fig. 11 where the modes of the tracking error x_H are all stable and real thus satisfying the condition (12). In

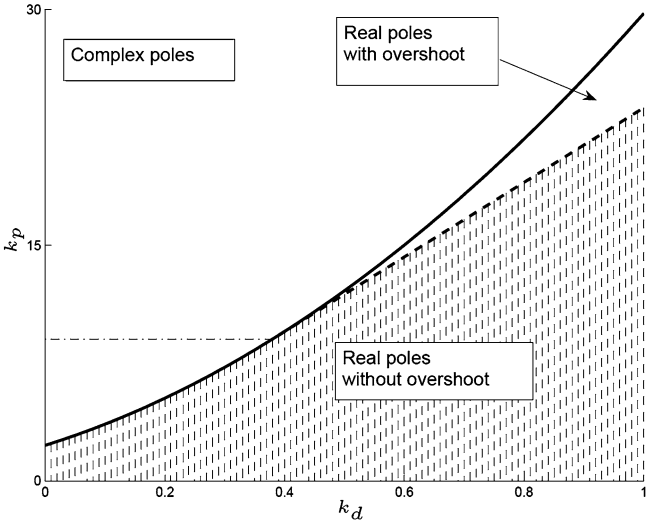

Fig. 11 Controller design for ramp response. The *shaded region* represents the values (k_p, k_d) ensuring no overshoot

Formaglio et al. (2005) we proved that boundaries of the region depicted in Fig. 11 can be easily computed and only depend on the parameters (M_m, B_m, K_m) of the mobile platform.

Summarizing, relying on the case of grasped object, a suitable PD controller is designed such that the reference signal x_C commanded to the robot is

$$x_C = (k_p + sk_d)x_H \tag{15}$$

which corresponds to choose in the block diagram of Fig. 6

$$C(s) = \left[\left(\frac{k_p}{2} + s\frac{k_d}{2}\right) \quad \left(\frac{k_p}{2} + s\frac{k_d}{2}\right)\right]. \tag{16}$$

The controller design strictly depends on hardware parameters, such as w and l, and software features like dimensions of to-be-grasped virtual objects. This means that typology and dimensions of virtual objects which can be grasped, parameters of the device and dynamical performance of the MHG are deeply related each other, thus designing hardware and virtual environments cannot be approached independently, but should be jointly developed to best exploit MHG functionalities.

4 Software design and implementation

This part of the paper presents the software design of the MHG. The software for the MHG system has been developed in the framework of the Haptik Library (de Pascale et al. 2004; The Haptik Library Developer's Manual—http://www.haptiklibrary.org). Haptik is a small open-source library with a component-based design [see Microsoft COM (Microsoft Component Object Model, COM—http://www.msdn.com) and OMG CORBA (OMG Common Object Request Broker Architecture, CORBA—http://www.corba.org)] for introductory material on the component model] which provides a hardware abstraction layer to access HD from different manufacturers in a standard way. The first contribution of using Haptik is that it allows to reuse existing software to build most of the new mobile system. The second contribution deals with the grow of the library itself. In fact, support for MHD is now provided through an additional plug-in for the library, the Mobile plugin, and is available to old and new applications, in a binary compatible way.

The rationale behind the design of the software for the MHG is the concept that every *mobile* haptic device can be seen as a *grounded* device with a huge workspace (Fig. 12). This abstraction still holds when considering more than one device, even if they actually share the same mobile base as in the case of our system. Therefore the whole MHG system, which is actually composed of a mobile platform with two grounded HD, from a software point of view can be seen simply as two MHD, which in turn are equivalent to two grounded HD with large workspace. It is worth noting that this abstraction

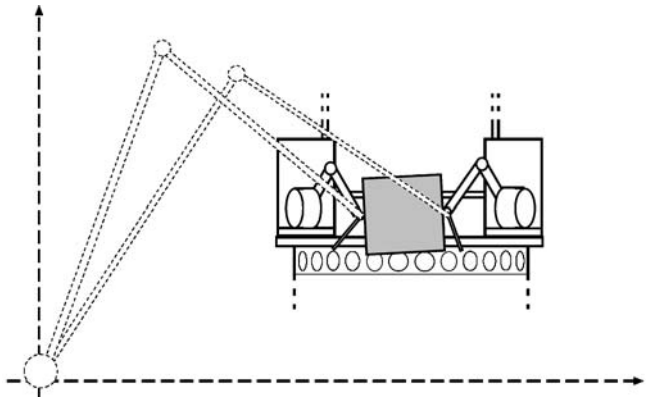

Fig. 12 A two contact point mobile haptic system can be seen as a grounded system featuring two haptic devices with a huge workspace

allows algorithms written for two or more grounded devices to work unchanged in the new mobile scenario. Therefore, existing applications can be easily re-used by extending the software that handles the grounded HD to support the new mobile haptic device abstraction.

Haptik is the natural framework for such an extension since it has powerful facilities for accessing many HD through a standard API, its runtime-loaded plugin structure allows for out-of-the-box addition of new features even on top of existing applications and finally the library has native support for building plugins that extends the capabilities of devices exposed by other plugins.

Thanks to this design, the software for the MHG has been built mostly on top of existing software. In fact the higher level virtual environment, the grasping and physical simulation as well as both graphic and haptic rendering, are all provided by the Grasp3D software, an application which has been re-used unchanged from a previous research project (De Pascale et al. 2005) focused on grasping. Furthermore, access to the two grounded devices is already provided by the Haptik Library, therefore the only new code that has actually been written regards the motion control and the communications with the mobile robot.

In what follows we provide a brief overview of the main features of the Haptik Library, with a special attention to the aspects regarding the MHG system, followed by a more detailed description of the software for the MHG and in particular of the mobile Plugin.

4.1 Overview of the Haptik Library

Haptik is a component-based library which provides easy yet powerful low-level access to haptic-enabled hardware through a robust API which hides the details and limitations (hard dependencies from particular driver version and installed DLLs/SOs, different access paradigms and coordinate systems, lack of binary compatibility, just to name a few) involved with direct usage of the many different native APIs used by each manufacturer. Interested readers are referred to online documentation (The Haptik Library Developer's Manual—http://www.haptiklibrary.org) for further details on the Haptik Library.

4.1.1 Device components and interfaces

Within Haptik each haptic device is represented by a *device component* which encapsulates the code and data needed to access and control a particular piece of hardware. Operations on these device components (e.g. reading state, sending forces, etc.) can be performed through one of the exposed *interfaces*[1], a run-time mechanism which can be thought as a collection of invokable methods.

Haptik's device components usually expose more than one interface: a standard interface (*IHaptikDevice*) is guaranteed to be always exposed by any device component, thus allowing all applications to work with any haptic hardware device, while, in addition, each device component can implement custom interfaces to expose special features of a particular hardware.

4.1.2 Plugins

Implementation of device components is not hard coded inside the library itself but is provided by a series of plugins. Usually each plugin supports all HD from a particular manufacturer, but they are not restricted to just wrapping the native API of a particular hardware and can also provide software only implementation of virtual devices, as well as extending the behavior of other components: through a mechanism referred to as *internal recursion*, plugins can use devices components from other plugins to implement their own extended devices.

Haptik's plugins are loaded at runtime by the library, therefore support for new devices can be added to old applications by simply adding the related plugin. Moreover, all dependencies on a particular hardware/software configuration are circumscribed to a single plugin, therefore this runtime loading scheme allows applications to use any supported devices without being strictly depending on any of them.

4.2 Software architecture

The software architecture of the MHG is reported in Fig. 13.

The simulation of physics along with the graphic and force rendering are all performed by the Grasp3D application (De Pascale et al. 2005). This is a C++

[1]Note that in order to avoid confusion throughout this work the term "interface" refers only to the software primitive of the component model and has never been used to refer to a haptic device.

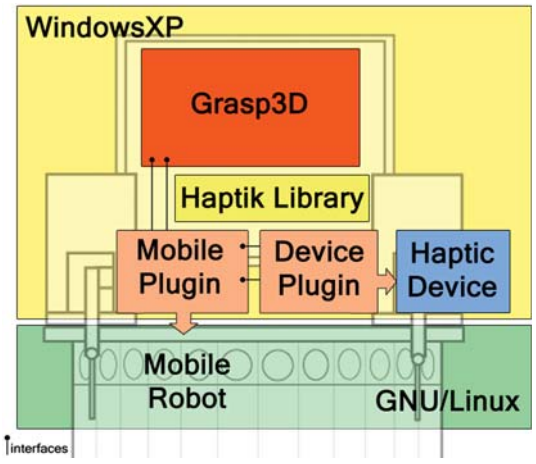

Fig. 13 Software architecture of the MHG

program [based on DirectX (Microsoft DirectX Multimedia Libraries—http://www.msdn.com/directx) for accessing graphic hardware, Haptik for accessing the HD, and Novodex SDK (The NovodeX physics engine—http://www.ageia.com) for simulating rigid body dynamics] which allows to grasp virtual objects using two grounded HD as shown in Sect. 2.2 (Fig. 14). The use of Grasp3D for the MHG has been achieved by simply choosing, on startup, devices exposed by the mobile plugin.

The mobile plugin provides the implementation for the MHD abstraction, by extending the behavior of the two grounded devices, and controls the motion of the mobile platform. Haptik acts as an initialization-time glue to establish connections between the Grasp3D application and the mobile plugin as well as the provider of access to grounded HD.

The PC hosting the Grasp3D application runs the WindowsXP operating system while the mobile robot hosts a GNU/Linux box. Communication between the two computers is provided on a wired lan connection through standard socket-based network connectivity.

Fig. 14 A screen shot of the Grasp3D application with the user lifting two boxes

4.3 The mobile plugin

The mobile plugin is an *extension* plugin (i.e. it does not access haptic hardware directly) for Haptik which performs the following actions:

- Access grounded HD through Haptik
- Communicate with the mobile robot through its API
- Implement the new mobile haptic device abstraction by combining state information from the grounded HD and the mobile platform.
- Drive the mobile robot using the control algorithms described in Sect. 3.2.

Access to HD is achieved through the Haptik *internal recursion* facility previously described. In this way the mobile plugin, and therefore the MHG, works independently from the HD that are mounted on its mobile base.

This allows the implementation of the device components providing the MHD abstraction to be pretty straightforward as it only performs the very few operations needed to change between the various coordinate systems involved. In more details, the reference frame returned to the upper layer is a composition of the reference frame reported by the grounded device with the current estimated reference frame of the mobile platform. Similarly forces are transformed from the world frame back to the reference frame of the grounded device before being applied. All other operations are simply forwarded on the interface exposed by the corresponding grounded device component.

The core of the plugin is the implementation of the control algorithm described in the Sect. 3.2: the mean position of the two end-effectors is used to drive the position of the mobile platform.

Data exchange between the controller and the device components is performed through a globally shared structure. Each component uses the platform reference frame stored by the controller to compute its world frame, and stores the position of its corresponding grounded device for use by the controller.

This design allows the asynchronous communications required for multi-rate execution, and also allows for easily extending the plugin to work with any number of grounded HD.

Reading and controlling the mobile robot position is performed through a simple C++ class which encapsulates details on using the Nomad XR4000 own API. It is worth noting that the device components exposed by the mobile plugin implements both the standard IHaptikDevice interface and a new *IHaptikMobileDevice* interface. This extended interface exposes the same operations available through the IHaptikDevice inter-

face but in addition it reports the current reference frame of both the device end-effector and its base.

This allows old applications using the IHaptikDevice interface to run unchanged with mobile devices, while at the same time allows new applications to take advantage of the extended capabilities of the new IHaptikMobileDevice interface. In particular the base reference frame reported by this interface can be easily used to move the user point of view in the virtual environment thus allowing for a more immersive experience.

5 Experimental application

In this section, we describe a simple experiment aiming to qualitatively evaluating the overall functionality of the MHG. To this purpose, an experimental application has been implemented by re-using the existing Grasp3D application (De Pascale et al. 2005) with the mobile plugin presented in this work.

The target is twofold: verifying (1) the tracking of user's locomotion by the mobile platform, and (2) user's ability to perform virtual object grasping and carrying exploiting visuo-haptic feedback provided by the system.

As already mentioned in Sect. 3, the prototype device is realized combining the mobile robot Nomad XR4000 to track user's motion and two Phantom Desktop HD to provide the user with a contact point for each hand, (Fig. 1). The user was both visually and haptically immersed in a virtual environment represented by a room with side length 3 m. Graphic feedback is provided via a LCD screen. The experimental task consists in three phases:

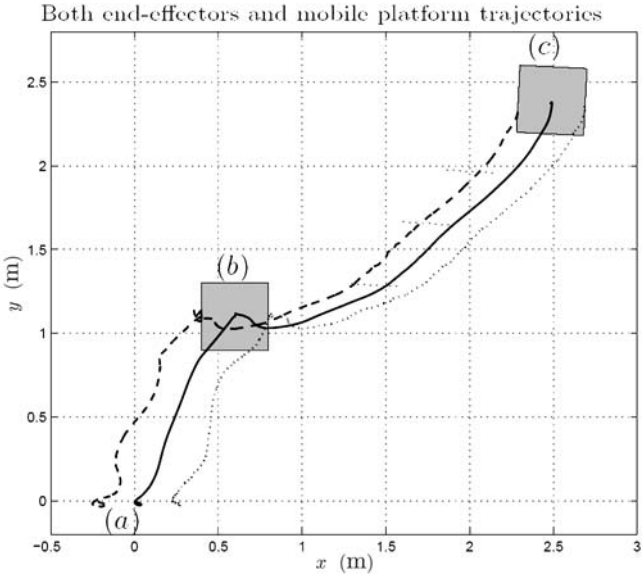

Fig. 15 Typical motion trajectories. The *dashed line* and the *dotted line* represent left and right end-effector trajectories, respectively; the *solid line* is the trajectory of the mobile robot. The *grey boxes* depicts the virtual object in its initial (*b*) and final (*c*) positions

1. Reaching a virtual object placed at a distance of about 1 m from the starting point
2. Grasping and lifting up the virtual object using both hands
3. Carrying the virtual object and placing it in another position placed at a distance of about 2 m from the initial object position.

The virtual object is a cube with side length $l = 0.4$ m and mass $m = 0.5$ kg. The values of object, controller and MHG parameters are those reported in Table 1. Typical motion trajectories of one experimental run are shown in Fig. 15.

The plots represent left (dashed line) and right (dotted line) end-effector trajectories; the solid line is the trajectory of the mobile robot, represented by the origin of the equivalent frame Σ_E. *a* represents the starting point of the simulation; *b* marks the position of the object to grasp and *c* is the target destination, where the object has been released by the operator. The whole experiment took about 50 s.

During the first phase, the user approaches the object to grasp and freely moves his hands in the left and right HD workspace. In Fig. 15 this corresponds to visible differences between the left and right end-effector trajectories from *a* to *b*. In spite of this asymmetry, the motion control algorithm was able to filter end-effectors trajectories thus smoothly tracking user's locomotion.

As regards the second task, i.e. grasping the virtual object (position *b*), the user exploited both visual and haptic feedback to decide the best way to approach the grasp. The grasping rendering has been implemented as described in Sect. 2.2 with friction coefficient $\mu = 1$ and local model stiffness k in (1) equal to 1 mm^{-1}.

Note that from a user perspective this type of grasp is slightly different from the pinch grasp, since it involves two hands instead of two fingers of the same hand. Hence, the user first tried several contacts in order to familiarize with the object and then definitively lifted it up (Fig. 16). However, during this transient phase the motion controller kept the platform almost still.

The final task consisted of carrying the virtual box to the target destination, and the corresponding trajectories are those from *b* to *c* in Fig. 15. The object trajectory during the motion is mainly translational and the distance between the two end-effectors is almost constant as shown in figure. During this phase, force rendering is performed while the mobile platform is moving, thus taking advantage from the smoother (5) to correctly render the virtual impedance at the two contacts. Note that for the experimental prototype, the maximum approximation error which may affect signal $\hat{x}_M$ is $V_M T_M = 4$ mm.

Despite the dynamics of mobile platform may affect the quality of haptic rendering and the overall performance of the MHG (Formaglio et al. 2005), the test users, whose previous experience with VR applications ranged from naive to expert, never had any troubles in

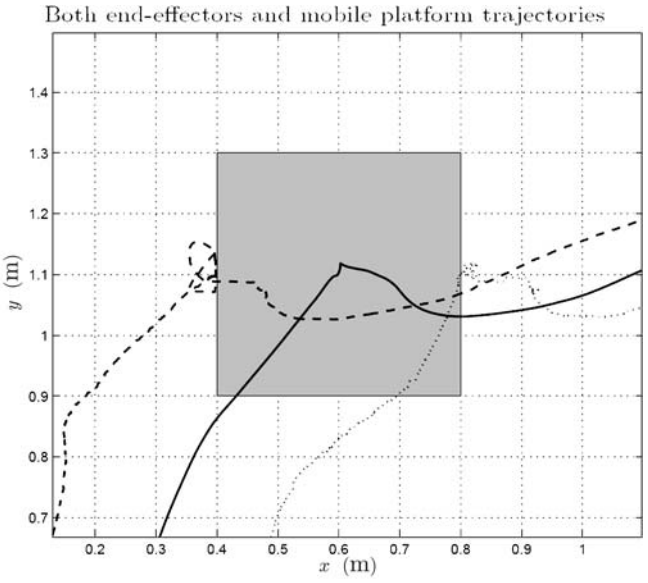

Fig. 16 A detail of the end-effectors (*dashed* and *dotted lines*) and mobile platform (*solid line*) trajectories: user tried several contacts before seizing the virtual box and lift it up

performing the task. They all reported that the contacts with the virtual object felt convincing and stable, whereas during the carrying phase the box mass and inertia rendered by the mobile device were rather realistic, thus globally perceiving a satisfying level of realism of immersion.

6 Conclusions and future perspectives

In this paper, a new haptic system which allows to simulate multi-contact grasping tasks in large environments, through the use of two standard HD grounded on a mobile robot has been presented. The main feature of the MHG is that of extending the workspace for visio-haptic simulation of grasping that usually suffers from workspace limitations. The MHG is a robotic platform that deals with two emerging research lines in haptics, namely grasping and mobile haptics. The Haptik Library allowed to exploit synergies with existing software to set up the overall system. In this framework, some basic guidelines to design a plugin for mobile haptic platforms have been provided, and finally an experiment has been reported to show the viability of this approach.

The proposed architecture features two main advantages: first the MHG provides the user with an operating workspace that is potentially unlimited on the horizontal plane. Hence, the perception of VR can be augmented by allowing full user's locomotion inside the simulated scenarios, thus improving the overall feeling of immersion. Secondly, from a software point of view, the implementation of such a system is simplified by the use of Haptik Library, which allows to easily re-use applications originally designed for desktop devices.

On the other hand, the main limitation of our current approach is that we studied and used only two translational degrees of freedom of the mobile platform without taking into account rotations.

While this simplified analysis provided satisfying results we think that better performances could be achieved by considering all degrees of freedom in an integrated setting. This will be object of our future investigations. A further limitation of our work is that we are referring to a mobile base with holonomic kinematics. Work is in progress to extend results to non-holonomic MP.

Regarding software, many enhancements are possible. The mobile plugin currently works only with the Nomad XR4000. However, it has been designed to be easily extended to other robotics MP by encapsulating all details on a specific mobile hardware in a C++ class. Finally, one of the most exciting results we aim to is that of implementing a virtual environment where many MHGs are involved to perform cooperative tasks within large virtual environments.

References

Barbagli F, Formaglio A, Franzini M, Giannitrapani A, Prattichizzo D (2005) An experimental study of the limitations of mobile haptic interfaces. In: Experimental Robotics IX. STAR, Springer Tracks in Advanced Robotics, Springer, Berlin Heidelberg New York

Barbagli F, Prattichizzo D, Salisbury JK (2005) Multi-point physical interaction with real and virtual objects. In: STAR, Springer Tracks in Advanced Robotics. Springer, Berlin Heidelberg New York

Conti F, Khatib O (2005) Spanning large workspaces using small haptic devices. In: Proceedings of the 1st joint Eurohaptics conference and symposium on haptic interfaces for virtual environment and teleoperator systems, WHC2005, Pisa

De Pascale M, Sarcuni G, Prattichizzo D (2005) Real-time soft-finger grasping of physically based quasi-rigid objects. In: Proceedings of world haptics conference, Pisa

Formaglio A, Prattichizzo D (2005) A smooth approximation of mobile platform displacement for mobile haptic interfaces. In: Proceedings of 2nd international conference on enactive interfaces, Genoa

Formaglio A, Giannitrapani A, Barbagli F, Franzini M, Prattichizzo D (2005) Performance of mobile haptic interfaces. In: Proceedings of IEEE conference on decision and control (IEEE CDC/ECC2005), Seville

Harwin WS, Melder N (2002) Improved haptic rendering for multi-finger manipulation using friction cone based god-objects. In: Proceedings of Eurohaptics conference

Johansson RS, Cole KJ (1994) Grasp stability during manipulative actions. Can J Physiol Pharmacol 72:511–524

Luenberger DG (1969) Optimization by vector space methods. Wiley, New York

Mason MT, Salisbury JK (1985) Robot hands and the mechanics of manipulation. MIT, Cambridge

Massie T, Salisbury J (1994) The PHANTOM haptic interface: a device for probing virtual objects. In: Proceedings of ASME winter annual meeting. Symposium of haptic interfaces for virtual environment and teleoperator system, pp 295–301

Nitzsche N, Hanebeck UD, Schmidt G (2003) Design issues of mobile haptic interfaces. J Rob Syst 20(9):549–556

de Pascale M, de Pascale G, Prattichizzo D, Barbagli F (2004) The Haptik Library, a component based architecture for haptic devices access. In: Proceedings of EuroHaptics 2004, Munich, Germany

Peshkin M, Colgate JE, Wannasuphoprasit W, Moore C, Gillespie B, Akella P (2005) Cobot architecture. IEEE Trans Rob Automat 17(4):377–390

Salisbury JK, Barbagli F, Frisoli A, Bergamasco M (2004) Simulating human fingers: a soft finger proxy model and algorithm. In: Proceedings of haptic symposium 2004, pp 9–17

Zilles CB (1995) Haptic rendering with the tool-handle haptic interface. Master Thesis, MIT Department of Mechanical Engineering