Mobile Haptic Interface for Large Immersive Virtual Environments: PoMHI v0.5

[1]Chaehyun Lee, [2]Min Sik Hong, [1]In Lee, [2]Oh Kyu Choi, [2]Kyung-Lyong Han, [2]Yoo Yeon Kim, [1]Seungmoon Choi, and [2]Jin S. Lee
[1]: Haptics and Virtual Reality Laboratory
POSTECH, Republic of Korea
[2]: Robotics and Automation Laboratory
POSTECH, Republic of Korea
{xlos, mshong, inism, hyh1004, sidabari, yooyeoni, choism, jsoo}@postech.ac.kr

Abstract - We present initial results of research toward a novel Mobile Haptic Interface (MHI) that provides an unlimited haptic workspace in large immersive virtual environments. When a user explores a large virtual environment, the MHI can sense the configuration of the user, move itself to an appropriate configuration, and provide force feedback for haptic display, thereby enabling a virtually limitless workspace. Our MHI (PoMHI v0.5) is featured with omni-directional mobility, a collision-free motion planning algorithm, and force feedback for general environment models. We also provide experimental results that show the fidelity of our mobile haptic interface.

Keywords - mobile haptic interface, mobile robot, omni-directional wheel, PID control with fuzzy logic control, haptic rendering

1. Introduction

A force-feedback haptic interface has an inherent limit on its workspaces and cannot render virtual objects larger than its workspace. This is tolerable in desktop applications where the size of a virtual environment is comparable to the workspace of a desktop haptic interface, but can be a severe limitation in large virtual environments such as the CAVE™. The traditional solution to this problem has been using a large haptic interface of a manipulator type [1] or of a string type (SPIDAR) [2][3]. However, their workspaces are still limited and cannot be easily scaled to virtual environments of different sizes.

A promising alternative is a Mobile Haptic Interface (MHI) that refers to a force-feedback haptic interface with a mobile base. The mobile base can move the haptic interface to an adequate place to render large virtual objects and thus can provide an unlimited workspace. Compared to other large haptic interfaces, the MHI is easier to be installed or removed due to its mobility, considerably smaller, and safer. Furthermore, the MHI can provide high-fidelity force feedback since a desktop haptic interface with greater precision is used for it.

The concept of the MHI was firstly proposed by Nitzsche et al [4][5]. Although their MHI, Walkii, had omni-directional mobility, a simple motion planning algorithm was used and only primitive objects could be rendered. Barbagli et al. presented two new MHIs in 2004 [6]. Both of the two MHIs used a commercial robot as its mobile base and adopted more complex motion planning algorithm than Walkii. More recently, a MHI with two desktop haptic interfaces was proposed for two-point manipulation [7].

In this paper, we present an initial version of POSTECH Mobile Haptic Interface (PoMHI v0.5) especially designed to be used with large visual environments such as the CAVE™. The PoMHI can move in any direction using three omni-directional wheels while avoiding collisions with a user, load general virtual environment models, and provide acceptable force feedback. The mobile base with the three omni-directional wheels was controlled via a PID-control with an additional fuzzy logic control. A motion planning algorithm guiding the movement of the mobile base was also developed. We also confirmed through experiments that the dynamics of the mobile base brings little interference on the forces delivered to the user.

The reminder of this paper is organized as follows. Section 2 briefly reviews the overall architecture of the PoMHI. Section 3 describes the hardware components and the control methods used in the system. In Section 4, we describe the software structure and the motion planning strategy. The effect of the mobile base dynamics on the final rendering force is also examined. Applications of the PoMHI are discussed in Section 5. Finally, we conclude this paper in Section 6.

2. System Architecture

The overall system architecture is shown in Figure 1. The IS-900 Tracking System (a processor and two trackers; InterSense Inc., USA) is responsible for tracking the current configurations (position and orientation) of both the PoMHI and the user. Tracked information is sent to the

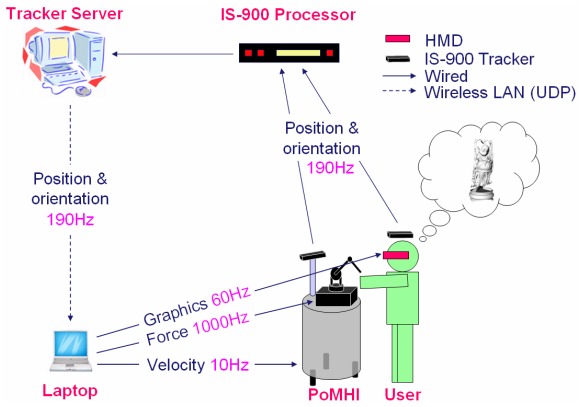

Fig. 1. Architecture of the PoMHI.

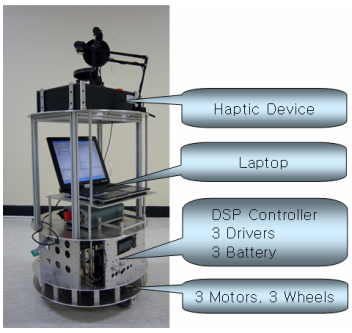

Fig. 2. Hardware structure of the PoMHI.

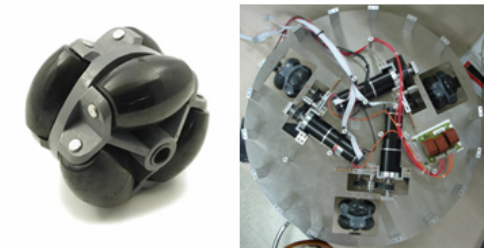

Fig. 3. An omni-directional wheel (left) and the placement of three wheels in the mobile base (right).

tracker server, and the server forwards it to the laptop inside the PoMHI via UDP protocol. Based on the user's configuration, the laptop determines an appropriate configuration of the mobile base and controls the base to place it to the configuration. Graphic and haptic rendering modules are also managed in the laptop. The user wears a head mounted display (HMD) and sees stereoscopic scenes of a virtual environment. Relevant communication and rendering rates are specified in the figure.

3. MHI Hardware

3.1 MHI Hardware Design

A. Overall Structure of the Hardware

The hardware structure of the PoMHI is shown in Figure 2. It consists of four main parts: omni-directional wheels and geared DC motors, a DSP control board and power amplifiers, a laptop, and a desktop 3 DoF haptic interface (PHANToM Premium 1.5A; SensAble inc., USA). The mobile base is responsible for moving the whole PoMHI to face the user, and the PHANToM is responsible for providing appropriate force feedback to the user.

B. Omni-directional Structure

Figure 3 shows the mobile-base actuation design using three omni-directional wheels. Our design follows the Y-shaped structure for holonomic motion of the mobile base. This is more adequate than a bidirectional mobile robot to follow possibly abrupt motions of a user. Each wheel is connected to a motor by a timing belt and a pulley.

The kinematics of the mobile base is derived from the relation of parameters represented in Figure 4. Here, the

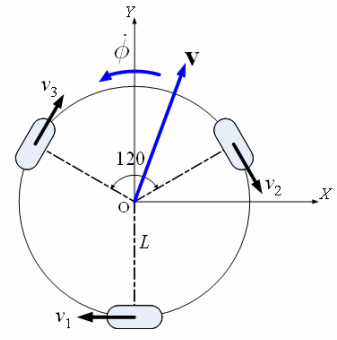

Fig. 4. Omni-directional structure on local coordinates.

origin of the local coordinate is set to the center of the base and the parameters are defined as:

v_i : Linear velocity of each wheel (i = 1, 2, 3)

$\mathbf{v} = [\dot{x}, \dot{y}]^T$: Linear velocity of the mobile base

$\dot{\phi}$: Angular velocity of the mobile base

L : Distance between a wheel and the orgin.

Then, the mobile base kinematics can be derived as follows [8][9]:

$$\begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} -1 & 0 & -L \\ \frac{1}{2} & -\frac{\sqrt{3}}{2} & -L \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & -L \end{pmatrix} \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{pmatrix} = A\dot{S}\,, \tag{1}$$

where $\dot{S} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{\phi} \end{bmatrix}^T = \begin{bmatrix} \mathbf{v}^T & \dot{\phi} \end{bmatrix}^T$.

3.2 MHI Control

A. Mobile Base Control Overview

The motion of the mobile base is based on PID control for the desired linear and angular velocities of the base. As shown in Figure 3, each omni-directional wheel has six free-rolling sub-wheels that may cause a slip. Moreover, it was observed that the mobile base contains inherent nonlinearity induced from the wheel and belt-pulley structure. We thus also use a fuzzy logic control (FLC) and other supplementary control techniques to overcome the problem. The overall control equation for motor i is:

$$U_i(k+1) = U_i(k) + \Delta U_i(k) + \Gamma_i(k)\,, \tag{2}$$

where $U_i(k)$ is a command to be sent to the motor amplifier and $\Gamma_i(k)$ is an output of the supplementary control. The sampling rate of the control loop is set to 50 Hz.

B. PID-based Velocity Control

Once a desired trajectory of the mobile base is determined from the motion planning algorithm (will be explained later), the desired velocity of each wheel is calculated by the kinematics in Eq. (1). Using these

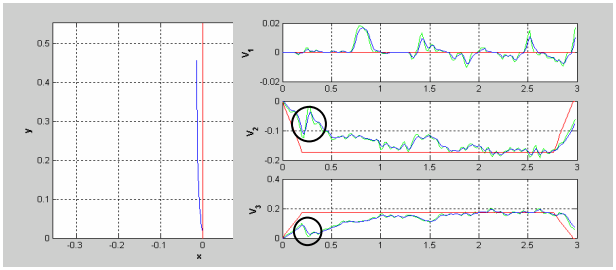

Fig. 5. Experimental results using the PID-based velocity control only ($\dot{x}$ = 0.0 m/s, $\dot{y}$ = 0.2 m/s, $\dot{\phi}$ = 0.0 rad/s).

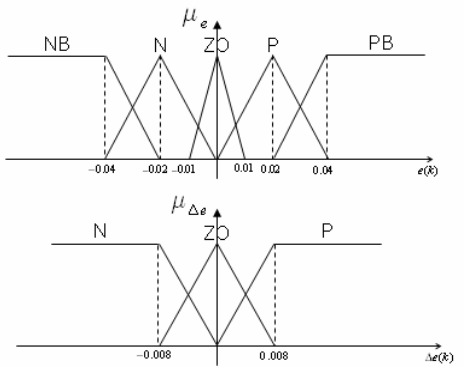

Fig. 6. Membership functions of velocity error, μ_e, and its variation, $\mu_{\Delta e}$.

velocities, each motor is controlled by the PID control method. The velocity of each wheel is estimated using the exponential filter as:

$$v_{i,\,filtered}(k) = (1-\alpha)v_{i,\,filtered}(k-1) + \alpha v_i(k)\,, \tag{3}$$

where $0 \le \alpha \le 1$ is the parameter of the filter.

To obtain the desired motor velocity, we use the PID control equation as follows:

$$U_i(k+1) = K_p e_i(k) + K_i \sum_{i=1}^{k} e_i(i) + K_d \Delta e_i(k)\,, \tag{4}$$

where $e_i(k) = v_{i,desired}(k) - v_{i,\,filtered}(k)$ is the velocity error of the motor, and K_p, K_i, and K_d are the PID gains.

Using only the PID control did not bring enough performance in our system. This is illustrated in Figure 5 where the red lines represent references and blue lines measured values. The left graph shows the position of the PoMHI, and the right three graphs show the velocities of the wheels. We can observe the system nonlinearity in the black circles where the measured velocity rather decreases in spite of the increase of the reference velocity. The nonlinearity seems to be caused by the increase of friction when the contact point of a wheel between a sub-wheel and the floor is changed. Because this nonlinearity is inherent characteristics of the wheel structure, eliminating this phenomenon perfectly may not be possible. We thus minimize the effect of the nonlinearity through additional control methods that will be described in the next subsections.

Table 1. Control rule set.

$\Delta e(k)$ \ $e(k)$	NB	N	ZO	P	NP
N	$\bar{y}^{(1,1)}$	$\bar{y}^{(2,1)}$	$\bar{y}^{(3,1)}$	$\bar{y}^{(4,1)}$	$\bar{y}^{(5,1)}$
ZO	$\bar{y}^{(1,2)}$	$\bar{y}^{(2,2)}$	$\bar{y}^{(3,2)}$	$\bar{y}^{(4,2)}$	$\bar{y}^{(5,2)}$
P	$\bar{y}^{(1,3)}$	$\bar{y}^{(2,3)}$	$\bar{y}^{(3,3)}$	$\bar{y}^{(4,3)}$	$\bar{y}^{(5,3)}$

C. Fuzzy Logic Control

We use a singleton fuzzifier, a product inference engine, and a center average defuzzifier in FLC of the mobile base. The inputs are the velocity error $e_i(k)$ and its increment $\Delta e_i(k)$. We use triangular membership functions (μ_e and $\mu_{\Delta e}$) as shown in Figure 6. Membership functions consist of five sections; negative big (NB), negative (N), zero (ZO), positive (P), and positive big (PB). These sections are determined experimentally.

Table 1 shows the rule set of the FLC derived from the input membership functions. Each element of the table, $\bar{y}^{(i,j)}$, represents the output of each rule. Using the rules and input variables, the updating output of FLC is [3]:

$$\Delta U_i(k) = \frac{\sum_{m=1}^{5}\sum_{n=1}^{3} \bar{y}^{(m,n)} \mu_e^m(k) \mu_{\Delta e}^n(k)}{\sum_{m=1}^{5}\sum_{n=1}^{3} \mu_e^m(k) \mu_{\Delta e}^n(k)}\,. \tag{5}$$

D. Supplementary Control

We adopt additional supplementary control when the following condition is satisfied:

$$|U_i(k)| > |U_i(k-1)| \;\;\&\;\; |v_{i,\,filtered}(k)| < |v_{i,\,filtered}(k-1)|\,. \tag{6}$$

This condition detects when the contact point of a wheel between a sub-wheel and the floor is changed. While this occurs, excessive errors are accumulated in the PID control which causes an overshoot in position control. An algorithm to minimize this behavior is described below.

If Eq. (6) is satisfied at time index k, the controller stores the sum of updating outputs $\Delta U_i(k)$ and the current velocity $v_e = v_{filtered}(k)$. In each control time l after k, the controller checks whether a condition, $|v_e| < |v_{filtered}(l)|$, is met. If it is, the controller compensates the excessive error by:

$$U_{i,sum} = \sum_{i=k}^{l} \Delta U_i(k) \;\text{ and} \tag{7}$$

$$U_i(l) = U(l) - \gamma \cdot U_{i,sum}\,, \tag{8}$$

where $0 \le \gamma \le 1$ is a proportional constant. Without the use of this control process, the system can be unstable. The final output of the supplementary control is as follows:

$$\Gamma(k) = \begin{Bmatrix} -\gamma \cdot U_{i,sum}(l), & k = l \\ 0 \quad , & k \neq l \end{Bmatrix}. \tag{9}$$

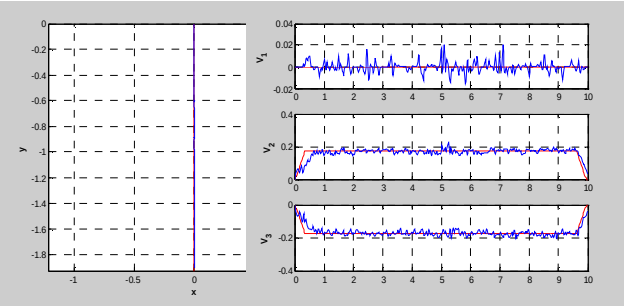

Fig. 7. Experimental results with the modified velocity control ($\dot{x}$ = 0.0 m/s, $\dot{y}$ = 0.2 m/s, $\dot{\phi}$ = 0.0 rad/s).

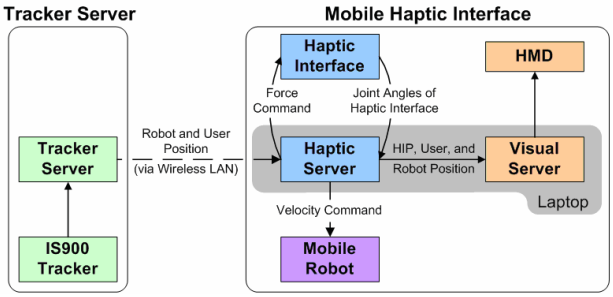

Fig. 8. Software structure of PoMHI.

E. Experimental Results

Figure 7 presents the results with the adapted FLC and the supplementary control. In the velocity graph of each wheel, fine trembles due to the change of contact points still show up, but we can see that the effect of the inherent nonlinearity has significantly decreased. Moreover, in the position graph, measured position values almost perfectly followed the reference values (a straight line).

4. MHI Software

4.1 MHI Software Design

The software in the MHI system consists of three parts. The tracker server program which is executed in the tracker server receives the configuration of each IS900 tracker and sends this information to the MHI via wireless LAN. Using this information, the software running on the laptop in the MHI controls all devices in the MHI (except for the motors controlled by the DSP board) and renders visual and haptic information.

The tracker server program and the MHI communicate through wireless LAN. Due to the update rate of the IS900 tracker, the network update rate is set to 190 Hz. To maintain this speed, the UDP protocol is used instead of the TCP protocol. TCP, which is more reliable than UDP, is sometimes too slow and even exhibits severe jitters. Whenever a packet is missed in TCP, the protocol re-transmits all packets after the missed packet and causes a long delay which is unacceptable in our application. The UDP is not as complex as the TCP, and is therefore faster. Although some packets can be missed and reordered during transmission in UDP, its effect can be made transparent to a user.

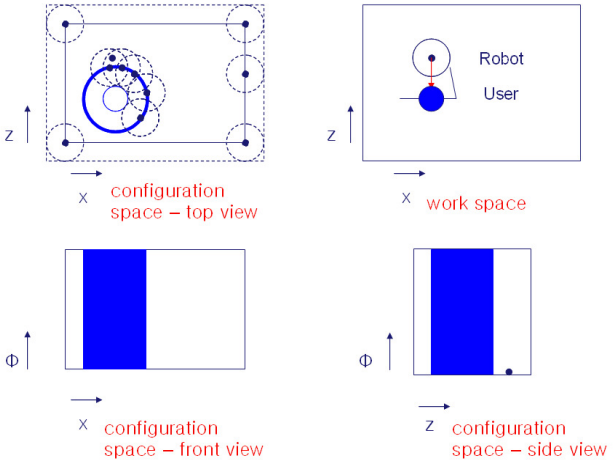

Fig. 9. Configuration space of the MHI.

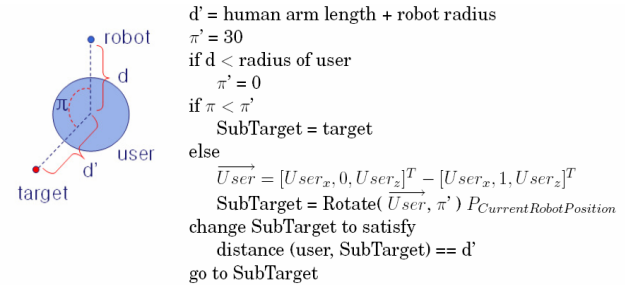

Fig. 10. Motion planning algorithm.

The haptic server program is comprised of networking, motion planning and haptic rendering parts. First, the networking part receives the tracker information from the tracker server and transforms the coordinate system from IS-900 to the world coordinate frame. This updates the configuration of the user, the mobile base and the haptic interface point (HIP; a point modeling the haptic tool tip) and sends the new information to the visual server through network. Note that since the haptic and visual server programs communicate with each other via network, the two servers can operate on separate machines for purposes such as higher performance or improved convenience. Second, the motion planning part calculates the next proper configuration of the mobile base and commands angular and linear velocities to the mobile base. Finally, the haptic rendering part detects a collision between the HIP and virtual objects and calculates haptic feedback force. All of these parts run at different rates, so the haptic server is designed as a multithreaded program.

The other server program which also runs on the laptop is the visual server. The visual server receives the configuration information from the haptic server and renders visual scenes. In our current system, a HMD is used for visual rendering, so the visual server also runs on the laptop with the haptic server. If other visual systems are used (e.g. CAVE™), the visual sever can be easily moved to another machine that controls the visual display and communicate with the haptic server through network, as is done in [11].

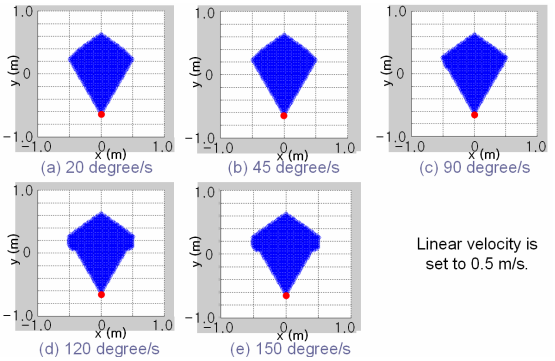

Fig. 11. Regions that the mobile base can move in 1 second with different maximum angular velocities.

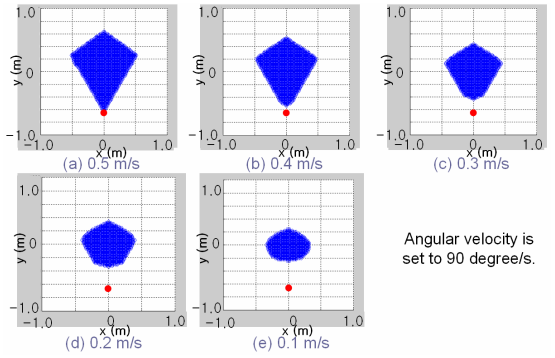

Fig. 12. Regions that the mobile base can move in 1 second with different maximum angular velocities.

4.2 Motion Planning Algorithm

Basically, the mobile base has to place the haptic device in a proper position where the device can give force feedback to a user most effectively while avoiding collisions with a user. Considering these constraints, we set the goal position of the MHI on a line which passes the user and the user's hand grasping the haptic tool. The distance between the user and the MHI is maintained within the typical arm length of an adult. Moreover, the MHI always faces the user so that the haptic interface tool is positioned in front of the user.

To represent the motion of the MHI, we introduce a 3D configuration space where its *x* and *z* axes represent the 2D position of the mobile base and its *y* axis the direction of the MHI in the workspace. In the configuration space, a configuration of the MHI is represented as a point, and a user is represented as a cylindrical obstacle (see Figure 9).

To move the MHI to a goal configuration without colliding with the user, we developed the following motion planning strategy consisting of three cases. First, when the MHI is too close to the user, it moves away from the user as soon as possible to avoid possible collisions with the user. Second, when the goal position is close to the current position, the MHI moves toward the goal. Finally, when the goal position cannot be reached directly (e.g., when the user is on the path), the MHI sets sub-goals between the goal and current position of the MHI and moves towards the nearest sub-goal. An implemented algorithm for this strategy is shown in Figure 10.

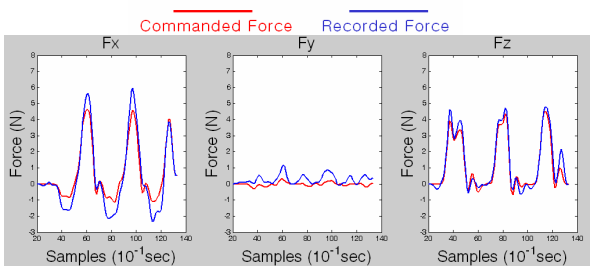

Fig. 13. Comparison between commanded and measured forces when the mobile base was stationary.

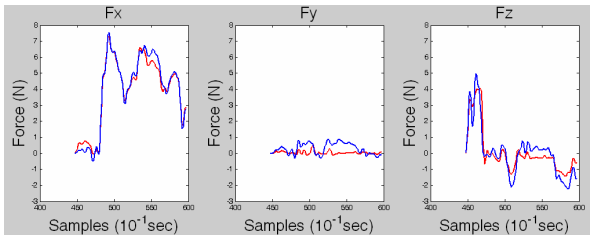

Fig. 14. Comparison between commanded and measured forces when the mobile base was moving.

To evaluate the performance of the motion planning algorithm, we simulated the motion of the mobile base and the user and represented the results with graphs in Figures 11 and 12. To simplify the simulation process, we assumed the mobile base could always move with maximum velocity. In each graph, the red point represents the current position of the MHI and the blue region represents the next position of the user where the mobile base can catch up with the user in 1 second. The results indicate that the change of maximum angular velocity does not significantly matter to the MHI while the change of the maximum linear velocity does.

4.3 Force Rendering Algorithm

To calculate feedback force for a user given the HIP position and a virtual environment model, we use the virtual proxy algorithm which is widely used for haptic rendering with static objects. In a MHI, the movement of a mobile base can cause unwanted forces that may be perceived by the user. However, if the mobile base is tightly position-controlled and significantly heavier than a haptic interface on top of it, we can ignore the effect of mobile base dynamics. To verify this fact, we conducted an experiment with a force sensor attached between the lank link of the PHANToM and the puck held by the user.

The results of the experiment are shown in Figures 13 and 14 where the red lines represent commanded force to the haptic device and the blue lines recorded force by the force sensor. According to the results, the differences between the commanded and recorded force in the case of the moving mobile base are not greater than those in the case of the stationary mobile base. This means that the mobile base dynamics does not significantly affect the feedback force delivered to the user. Therefore, we calculate torque commands sent to the haptic interface only considering the static torque-force relationship of the desktop haptic interface.

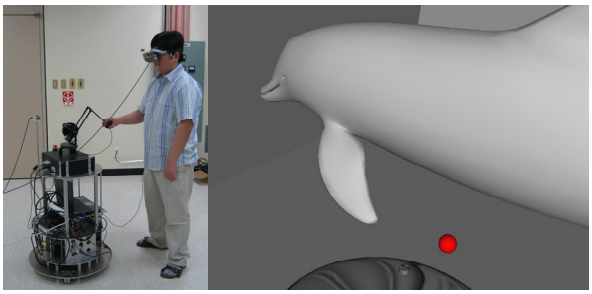

Fig. 15. A user playing with the virtual dolphin using the PoMHI (left) and the visual scenes displayed to the user via the HMD (right). The red sphere represents the HIP.

4.4 Virtual Environment Model

Not only primitive models such as a plane and a cylinder but also complex triangular mesh models can be loaded and rendered in the PoMHI. The visual server also shows the current position of the HIP to let the user know where s/he interacts with and the wall for a warning about the physical workspace limit in a room. The model of the mobile base is also loaded and visually rendered via a HMD in order to help the user avoid colliding with the mobile base for the safety purpose.

5. Applications

The current MHI system can provide the user with visual and haptic feedback about large virtual environments. One example is shown in Figure 15. A complex dolphin model consisting of 4488 faces is loaded, and the user sees and touches the real-sized dolphin model (1258 mm wide, 426 mm high, and 352 mm deep). For the stability of the mobile base, we set the maximum linear and angular velocity to 0.2 m/s and 40 degree/s, respectively. With this maximum velocity values, the MHI can follow the user in most cases, unless the user moves abruptly.

6. Conclusions and Future Work

We have develop an initial version of new mobile haptic interface named PoMHI and its applications. The PoMHI can place itself to an appropriate configuration to provide boundaryless haptic feedback while avoiding collisions with the user, and handle general virtual environment models. We adopted several motion control methods to stably and correctly control the motion of the mobile base. We also examined the fidelity of force display in the PoMHI by comparing actual force outputs with desired values.

We are currently working on a next version of the PoMHI. This version has four omni-directional wheels with advanced design and a lift for the desktop haptic interface for the extension of its workspace in the height direction. We are also upgrading the software in terms of more sophisticated motion planning algorithm, more precise kinematics calibration, and force computation algorithm considering the effect of the mobile base dynamics. Once all of these are completed, we will integrate the PoMHI into the CAVE™ that is the most immersive large virtual environment platform among the present.

Acknowledgements

This work was supported by a research grant No. R01-2006-000-10808-0 from the Korea Science and Engineering Foundation (KOSEF) funded by the Korea government (MOST).

References

[1] F. P. Brooks, M. Ouh-Young, J. J. Batter, and P. J. Kilpatrick, “Project GROPE - Haptic Displays for Scientific Visualization,” In *Proc. SIGGRAPH 90*, pp. 177-185, 1990.

[2] L. Buoguila, M. Ishii, and M. Sato, “Multi-Modal Haptic Device for Large-Scale Virtual Environment,” In *Proc. 8th ACM International Conference on Multimedia*, pp. 277-283, 2000.

[3] N. Hashimoto, S. Jeong, Y. Takeyama, and M. Sato, “Immersive Multi-Projector Display on Hybrid Screens with Human-Scale Haptic and Locomotion Interfaces,” In *Proc. International Conference on Cyberworlds*, pp. 361-368, 2004.

[4] N. Nitzsche, U. D. Hanebeck, and G. Schmidt, “Mobile haptic interaction with extended real or virtual environments,” In *Proc. RO-MAN 2001*, pp. 313-318, 2001.

[5] N. Nitzsche, U. D. Hanebeck, and G. Schmidt, “Design Issues of Mobile Haptic Interfaces,” *Journal of Robotics Systems*, Vol. 20, No. 9, pp. 549-556, 2003.

[6] F. Barbagli, A. Formaglio, M. Franzini, A. Giannitrapani, and D. Prattichizzo, “An experimental study of the limitations of mobile haptic interfaces,” In *Proc. ISER 2004*, 2004.

[7] Formaglio, M. D. Pascale, and D. Prattichizzo, “A mobile platform for haptic grasping in large environments,” *Virtual Reality*, Vol. 10, No. 1, pp. 11-23, 2006.

[8] Leow Y.P., Low K.H., Loh W.K., “Kinematic modeling and analysis of mobile robots with omni-directional wheels,” In *Proc. ICARCV 2002*. Vol. 2, pp. 820-825, 2002.

[9] Yong Liu, Xiaofei Wu, J Jim Zhu, Jae Lew, “Omni-Directional Mobile Robot Controller Design by Trajectory Linearization,” In *Proc. American Control Conference*, 2003.

[10] L. X. Wang, A *Course in Fuzzy Systems and Control*, Prentice-Hall, Inc., 1997.

[11] E. Dorjgotov, S. Choi, S. R. Dunlop, and G. R. Bertoline, “Portable Haptic Display for Large Immersive Virtual Environments,” In *Proc. HAPTICS 2006*, pp. 321-327, 2006.