A Coupling Library for the *Force Dimension* Haptic Devices and the *20-sim* Modelling and Simulation Environment

F. Sanfilippo
Department of Maritime Technology and Operations, Aalesund University College
Postboks 1517, 6025 Aalesund, Norway
fisa@hials.no

P. B. T. Weustink
Controllab Products B.V.
Hengelosestraat, 500
7521 AN Enschede, Netherlands
paul.weustink@controllab.nl

K. Y. Pettersen
Department of Engineering Cybernetics, Norwegian University of Science and Technology
7491 Trondheim, Norway
kristin.y.pettersen@itk.ntnu.no

***Abstract*— A haptic feedback device is a device that establishes a kinaesthetic link between a human operator and a computer-generated environment. This paper addresses the bidirectional coupling between a commercial off-the-shelf (COTS) haptic feedback device and a general-purpose modelling and simulation environment. In particular, an open-source library is developed to couple the *Force Dimension omega.7* haptic device with the *20-sim* modelling and simulation environment. The presented coupling interface is also compatible with all the different haptic devices produced by *Force Dimension*.**

The proposed integrated haptic interface makes it possible to track the user's motion, detect collisions between the user-controlled probe and virtual objects, compute reaction forces in response to motion or contacts and exert an intuitive force feedback on the user. A real-time one-to-one correspondence between reality and virtual reality can be transparently created. This allows for a variety of possible applications. Stability issues, performance issues, design and virtual prototyping challenges can be addressed and investigated for research purposes. In addition, design and virtual prototyping are also of interest to industry. Realistic training environments can be developed for the user considering different possible operations and stressing the importance of usability and user experience. Experiments based on using haptics technology in the field of education can also be easily performed.

To demonstrate the potential of the proposed coupling, a case study is presented. Related simulations and experimental results are carried out.

***Index Terms*— Haptics, Human-computer interaction, Simulation.**

I. Introduction

Human-computer interaction covers a broad spectrum. The interface between humans and the computers they use is crucial for facilitating this interaction. From a historical perspective, this intercommunication is based on one-directional channels of information. From one side, keyboard, mouse and joystick inputs are generally adopted to transfer human instructions to computers. Visual and audio information is commonly sent back from the computer to the user. In this information loop there is no kinaesthetic energy flow to the operator because the sense of touch is not involved [1].

However, touch is one of the most reliable and robust senses, and is fundamental to our memory and in discerning.

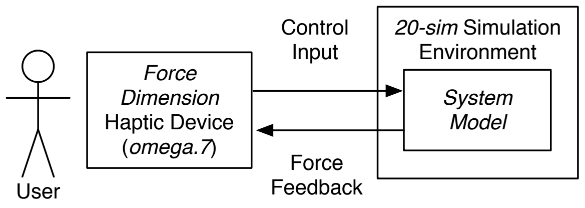

User
Force Dimension Haptic Device (*omega.7*)
Control Input
Force Feedback
20-sim Simulation Environment
System Model

Fig. 1: The idea of creating a bidirectional coupling between the *Force Dimension omega.7* haptic feedback device and the *20-sim* modelling and simulation environment.

To provide the user with additional and intuitive information, haptic technology can be employed [2]. Haptic feedback, also known as *haptics*, is the use of the sense of touch in a human-computer interface. A variety of possible applications are made possible by the use of haptics, including the possibility of expanding the abilities of humans: increasing physical strength, improving manual dexterity, augmenting the senses, and most fascinating, projecting human users into remote or abstract environments.

Nowadays, different commercial off-the-shelf (COTS) haptic feedback devices exist [3]. There are essentially two ways to access these devices: device-specific low-level application programming interfaces (APIs) written by the manufacturer and generic high-level libraries developed by either third parties or manufacturers [4]. The former are usually very simple interfaces, which users like researchers and industrial developers use to add haptic capability to their applications by partially hiding the complexity of haptic device programming. The latter are relatively larger libraries that, besides haptic rendering, typically address graphical and sound-related aspects also. However, the integration of these devices still requires significant programming skills. Therefore, this process can often be a tedious and time-consuming task, which may prevent developers from fully focusing on the application logic necessary to achieve the core task. In addition, most of the currently available APIs do not offer flexible tools for modelling, simulation and analysis of multi-domain dynamic systems. Consequently, developers are forced to develop custom-made solutions that often lack of generality.

To give researchers and other users the possibility of adding haptic capability to their applications in a more flexible and transparent fashion, the bidirectional coupling between a COTS haptic device and a general purpose modelling and simulation environment is addressed in this paper. In particular, we present an open-source library that allows for easily coupling the *Force Dimension omega.7* haptic device [5] with the *20-sim* modelling and simulation environment [6]. The underlying idea is shown in Fig. 1. It should be noted that the presented coupling interface is compatible with all the different haptic devices produced by *Force Dimension*. The library allows users to focus only on the core task to be achieved with their applications. Essential functions are provided with the library including tracking the user's motion, detecting collisions between the user-controlled probe and the virtual objects, computing reaction forces in response to motion or contacts, and exerting an intuitive force feedback on the user. It is possible to effectively establish a simultaneous interaction between reality and virtual reality. This allows for the exploration of a variety of possible applications. From a research point of view, stability and performance issues associated with haptic interaction can be addressed. Concerning both industry and research applications, design and virtual prototyping challenges can be studied. With regard to usability and user experience, realistic training environments can be developed for the user considering a variety of possible operations and stressors. Focusing of the field of education, experiments on applied haptics technology may also be easily investigated.

The proposed coupling library is an open-source project and it is available on-line at `https://github.com/filipposanfilippo/SimHaptics`, along with different examples and documentation.

The paper is organised as follows. A review of the related research work is given in Section II. In Section III, we focus on the description of the coupling architecture. A case study together with the related simulations and results are shown in Section IV. In Section V, conclusions and future works are outlined.

II. Related Research Works

A brief review of different currently available COTS haptic devices is given in this section, highlighting the possibilities of integration with third party applications or with modelling and simulation environments. At present, there is a large variety of devices available ranging from the conventional mouse with added tactile or force feedback to haptic gloves with limited movement. More complex devices that exhibit a higher number of degrees of freedom (DOFs) are also available.

A common typology of haptic devices concerns devices that can be controlled by the user with a tool of some kind. This category includes pen-based devices or devices that allow the user to attach a specific type of tool corresponding to the desired application scenario. The advantages of the pen-based devices lie in the fact that they are very intuitive for the user and can easily be used to represent the hand-held tools. From a kinematic point of view, these force feedback devices are usually constructed by either using a serial or a parallel mechanism. Serial mechanisms are normally more compact and relatively faster. The drawback is that as each kinematic link is added to the chain, the total inertia increases and the total stiffness decreases. On the contrary, parallel mechanisms do not exhibit the above problem and have a much higher stiffness. The disadvantage over serial mechanisms is that the device's elements can physically interfere [3].

One example of serial pen-based devices is represented by the *PHANTOM* series of devices produced by *SensAble Technologies* [7]. These devices feature from 3 to 6 DOFs. The *PHANTOM* series can be controlled using the *OpenHaptics* toolkit [8], which is a haptics library developed by *SensAble Technologies* to facilitate the adoption of haptics technology. This library is based on *OpenGL* [9] APIs, making it familiar to graphics programmers and facilitating the integration with existing *OpenGL* applications. The *OpenHaptics* toolkit has a two-layer hierarchical organisation including the Haptic Device APIs (HDAPIs) and Haptic Library APIs (HLAPIs). The HDAPIs layer provides low-level access to the haptic device, while the HLAPIs layer provides advanced haptic rendering tools. Both HDAPIs and HLAPIs layers are built on top of the *PHANTOM* Device Drivers (PDD), the device drivers for THE *PHANTOM* series of haptic devices. However, the *OpenHaptics* toolkit does not provide any abstraction layer specifically designed for system modelling and analysis.

One example of haptic devices with a parallel mechanism is the *Falcon* produced by *Novint* [10]. This 3-DOFs device is one of the first low-cost commercial haptic devices primarily intended as a game controller. *Novint* provides developers with a software development kit (SDK) called Haptic Device Abstraction Layer (HDAL) [10]. This SDK allows for the creation of custom applications and interfaces with the *Falcon* device. However, the SDK does not provide any specific tools for system modelling, thereby forcing developers to adopt third party applications or custom-built solutions.

Another example of haptic devices with a parallel mechanism is the *Force Dimension omega.7* [5]. The *omega.7* features 7 DOFs, allowing for high precision active grasping capabilities and orientation sensing. The *omega.7* is currently one of the high-end expensive COTS devices. It is used in the aerospace and medical industries for different safety-critical applications. This device can be controlled by using a powerful SDK [5], which is the software interface provided by *Force Dimension*. The SDK enables users to programmatically add haptic capability to their applications and offers compatibility with most existing haptic visualisation packages. However, the integration with external modelling and simulation environments still requires custom-built solutions and can be a time-consuming process for developers.

Apart from the already mentioned API sets that are specifically designed for particular classes of devices from the same manufacturers, there are different libraries that can also work with devices from different manufacturers. For instance, the *Haptik* library [4] is an open source library with a component based architecture that acts as a hardware abstraction layer to provide uniform access to a variety of different haptic devices. This library includes a set of interfaces that hide differences between devices from user applications. However, the proposed framework does not contain graphic primitives, physics related algorithms or complex class hierarchies. Another example is the *CHAI3D* library [11], developed at the Stanford University. This library is designed for visualisation applications and interactive real-time simulations. Some of its underlying algorithms can be easily tweaked, including collision detection and interaction paradigms. The *CHAI3D* library takes an important step toward developer-friendly creation of multimodal virtual realities, by tightly integrating the haptic and visual representations of objects and by abstracting away the complexities of individual haptic devices. However, advanced tools for control applications are not included in the library.

To the best of our knowledge, no haptic library exists yet that makes it possible to completely hide the complexity of haptic device programming by offering advanced modelling, simulation and analysis tools. The main contribution of this paper is to propose such a library.

III. Bidirectional Coupling Architecture

The underlying idea of this work is to develop a bidirectional coupling between a commercial COTS haptic feedback device and a general purpose modelling and simulation environment. The aim is to give developers the possibility of adding haptic capabilities to different systems with various requirements. Therefore, the choice of a multi-purpose haptic device as well as the selection of a general modelling and simulation environment are crucial. In this section, we first describe the adopted haptic device and modelling environment providing the motivations for these choices and focusing on the possibilities of integration. Subsequently, the proposed coupling library is described in detail.

A. Force Dimension omega.7

When choosing the haptic device to be adopted for the proposed coupling, several aspects were considered:

- High dexterity. To provide haptic capabilities to various systems with different specifications, a high dexterity is required for the adopted haptic device;
- High performance. To provide the user with a realistic experience, a highly accurate haptic rendering is required.

These guidelines were provided by the *Force Dimension omega.7* [5], which is probably the world's most advanced desktop 7-DOFs haptic device. It features high precision active grasping capabilities with orientation sensing. The force feedback gripper offers extraordinary haptic capabilities, enabling instinctive interaction with complex haptic applications. Based on a unique parallel kinematics structure, the *omega.7* is designed for performance featuring superior mechanical stiffness compared to any other competing devices on the market. The *omega.7* controller enables the rendering of high contact forces at a rate attaining 4 KHz. To provide the highest degree of haptic transparency, accurate gravity compensation is maintained in translation and orientation space by perfectly coupling passive and actuated components together. The unique kinematics design perfectly decouples translations and rotations, enabling the *omega.7* base to accommodate various interchangeable end-effectors to meet different application requirements. The *omega.7* is widely used for both research and education applications [12].

The *omega.7* can be controlled in two different operational scenarios: haptics and robotics mode. These two operational modes can be also combined. As already mentioned in Section II, *Force Dimension* provides a convenient SDK, which is split in two sub-components, the *Haptics* SDK and the *Robotics* SDK, to offer a highly accurate control over the *omega.7*, in both the possible operational modes. The *Force Dimension* SDK provides support for multi-threaded, multi-device programming and is available for a wide range of operating systems. The combination of both haptics and robotics capabilities into a single unified framework allows developers to create powerful collaborative interfaces between people and machines.

Even though the *Force Dimension* SDK offers great advantages by partially hiding the complexity of haptic device programming, the integration with external modelling, simulation and analysis environments still requires advanced programming skills to implement custom-built solutions and can be a time-consuming process for developers.

B. 20-sim modelling and simulation environment

When considering the simulation environment to be adopted for the proposed coupling, different aspects were taken into account:

- Modelling of multi-domain and complex systems. The idea is to develop a common haptic control system that can be used for controlling different models. Therefore, a flexible modelling and simulation environment that includes system dynamics and makes it possible to design complex mathematical models is needed. Such models can be complicated given that they may represent multi-domain systems where a large number of DOFs and a large number of rigidly connected parts are involved;

- Modular approach. Since the aim of this work is to control different system by using the same haptic device, the adopted modelling and simulation environment needs to be flexible enough to make it possible to easily modify the kinematics and dynamics of the controlled systems. Therefore, support for a modular design approach is needed;
- Physical interaction. The possibility of developing controllers for embedded software and the creation of virtual prototypes for use in hardware-in-the-loop (HIL) simulations are crucial requirements for choosing the proper modelling and simulation environment.

For these reasons, the *20-sim* modelling and simulation environment [6] was adopted to be coupled with the selected haptic device. *20-sim* is a modelling and simulation program for mechatronic systems. *20-sim* allows for designing different models in a graphic and intuitive manner, similar to drawing an engineering scheme. Various models can be created using equations, block diagrams, physical components and graphs. With these models the behaviour of multi-domain dynamic systems can be transparently simulated, analysed and controlled. Different virtual objects can be created using a *3D Animation Editor*. Animations are composed of predefined objects like cubes, spheres, lines, squares, cameras and lights. Complex objects can be imported from Computer Aided Design (CAD) packages using common exchange formats. It is also possible to simulate plasticity for the virtual objects. Besides, it is possible to generate control algorithms and run them on hardware for rapid prototyping and HIL-simulation. Besides, *20-sim* also offers the possibility to users to write their own source code using an external program compiler. A compiled dynamic link library (DLL) can be loaded into *20-sim* so that functions that describe input-output relations can be used with the desired model.

C. Bidirectional coupling library

The proposed bidirectional coupling architecture is shown in Fig. 2. The haptic device is represented as a two-port interface which characterises the exchange of energy between the human operator and the virtual environment [13]. This representation captures the relationship between efforts (forces) and flows (velocities). In particular, forces, F_h, are applied to the human operator in response to applied velocities, v_h. The corresponding forces and velocities in the virtual environment are indicated as F_e^* and v_e^*, respectively. The star superscript indicates that a variable is discrete; it is otherwise assumed to be continuous. The negative sign on the velocity v_e^* is necessary to maintain consistency. In the following, the key elements of the proposed bidirectional coupling are presented.

As shown in Fig. 2, the proposed coupling library is built on top of the *Force Dimension* SDK stack layers. This allows for a transparent low-level access to the adopted haptic

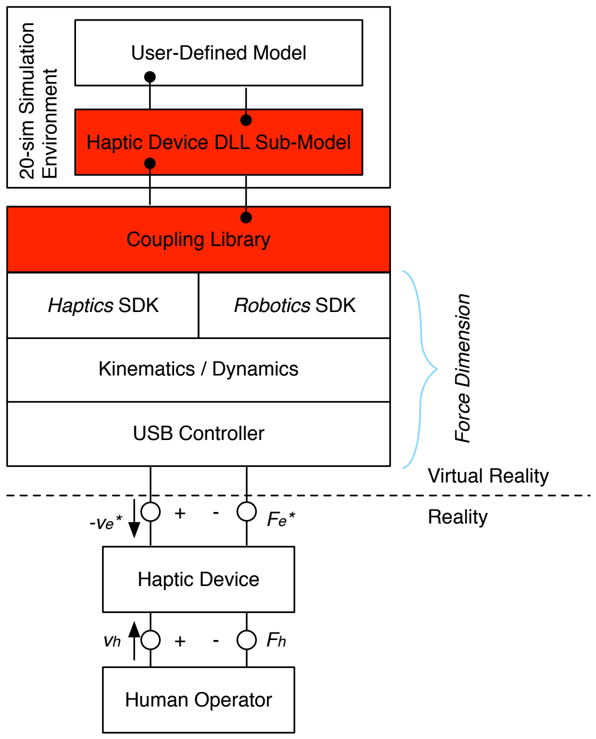


Fig. 2: The proposed bidirectional coupling architecture.

device. It should be noted that this approach allows for supporting all the different haptic devices produced by *Force Dimension*. Starting from the lowest logic level, the native SDK provides the following layers:

- a real-time Universal Serial Bus (USB) controller, which is responsible for the communication to the physical device through the serial channel;
- the kinematic and dynamic model of the adopted haptic device so the latter can be controlled in both haptics and robotics mode;
- the *Haptics* SDK and the *Robotics* SDK. The former offers all the basic functions to read positions and to program desired forces in Cartesian space. Expanding these fundamental capabilities, the latter leverages the *Haptics* SDK by introducing an advanced set of real-time routines to precisely control the position of the device.

The proposed coupling library works as an abstraction layer for all the methods provided by the *Force Dimension* SDK. In particular, the library is implemented as a DLL library, which can import the two main interfaces corresponding to the *Haptics* SDK and the *Robotics* SDK, respectively. In this way, all the native low-level methods provided by *Force Dimension* are accessible. For example, the pseudo code for the DLL main class that imports the *Haptics* SDK interface for the *Force Dimension omega.7* is shown in Algorithm 1. At run-time, the native *dhdSetForceAndTorque-*

```
//imports the Haptics SDK interface
#include "dhdc.h"
DLLEXPORT int omega7(double *inarr, int inputs,
    double *outarr, int outputs, int major) {
  if (major) {
        // set force and torque
        dhdSetForceAndTorqueAndGripperTorque(inarr
            [0], inarr[1], inarr[2], inarr[3],
            inarr[4], inarr[5], inarr[6]);
       // get position and orientation
       dhdGetPositionAndOrientationDeg(&outarr[0],
            &outarr[1], &outarr[2], &outarr[3], &
           outarr[4], &outarr[5]);
       // get gripper angle
       dhdGetGripperAngleDeg(&outarr[6]) < 0);
    }
  return FUNCTION_OK;}
```

Algorithm 1: The pseudo code of the DLL main class that imports the *Haptics* SDK interface.

```
parameters
  string dll_name = 'omega7.dll';
  string function_name = 'omega7';
variables
  real dll_input[7], dll_output[7];
code
  // prepare the DLL function inputs with desired
      forces and torques
  dll_input = [fx;fy;fz;ta;tb;tg;t];
  // invoke the DLL function
  dll_output = dll(dll_name, function_name,
      dll_input);
  // read the DLL function outputs containing
      position and orientation
  [px;py;pz;oa;ob;og;a] = dll_output;
```

Algorithm 2: The pseudo code for the *20-sim* static DLL sub-model.

AndGripperTorque is invoked to set forces and torques for the device, while the low-level methods *dhdGetPositionAndOrientationDeg* and *dhdGetGripperAngleDeg* are adopted to get position and orientation.

To achieve the desired coupling with the *20-sim* program, the *20-sim* DLL loading mechanism is used. To do so, a haptic device static DLL sub-model is implemented. This sub-model can easily be imported into the *20-sim* modelling and simulation environment, interconnected to the user-defined model and integrated with the block diagram of the system to be designed. At each simulation time-step the static DLL sub-model calls the DLL specific function to set the efforts that are used to give force feedback to the operator and to read the current position of the input device's end-effector. In particular, the pseudo code for the *20-sim* static DLL sub-model is shown in Algorithm 2.

As already mentioned, it should be noted that this approach allows for supporting all the different haptic devices produced by *Force Dimension*.

IV. Case Study and Simulations

To demonstrate the potential of the proposed coupling library, a simple case study is presented from a user point of view. To add haptic capabilities to the system to be designed, the user needs to simply follow the following steps:

- import and drag the haptic device static DLL sub-model from the *20-sim* library of block diagram elements to the *20-sim* editor;
- store the DLL library in the same *20-sim* project folder.

Once these steps are accomplished, the user can build any desired system by graphically representing the mathematical relationships between signals. In particular, the *20-sim* block diagrams are especially suited for modelling a large variety of different control systems.

The system block diagram considered as a case study is shown in Fig. 3. The idea is to show a simple haptic interaction between the operator and a virtual object with the intent of demonstrating the features of the proposed library. The simulated virtual object can be manipulated by the operator in the simulator. In this specific case, constant source blocks are used to generate all the input forces and torques for the haptic device static DLL sub-model, except for the force input along the x-axis. The latter is given by the output from a proportional-derivative (PD) controller in series form. The actual position and orientation, which is the output of the haptic device static DLL sub-model, flows first to a delay block with one sample interval delay. This block is a discrete-time operator. The input of the PD controller is the difference between the delayed output and the signal coming from a motion profile block. The selected profile type is a continuous pulse.

Related simulations are carried out. The experiment set-up is shown in Fig. 4-a. By using the *20-sim* 3D-animation Editor, a simple cube is created as a virtual probe, as shown in Fig. 4-b. This animation is fully linked to the presented system block diagram. In particular the cube position and orientation is given by the haptic device static DLL sub-model output. By using the *20-sim* plot tool, different time plots for the main system variables can easily be obtained as shown in Fig. 4-c. In this case, the DOFs of the haptic device are plotted including the wrist orientation angles, *oa*, *ob*, *og*, the end-effector position in Cartesian space, *px*, *py*, *pz*, and the gripper angle, *a*.

V. Conclusions and Future Work

An open-source library that allows for establishing a bidirectional coupling between the *Force Dimension omega.7* haptic device and the *20-sim* modelling and simulation environment was presented in this work. It should be noted that the presented coupling interface is compatible with all the different haptic devices produced by *Force Dimension*. The library presented enables developers to easily add haptic capability to their systems. Furthermore, this coupling allows the developers to fully exploit the flexible and intuitive tools for fast-developing offered by *20-sim*. Different out-of-the-box functionalities are present, including tracking

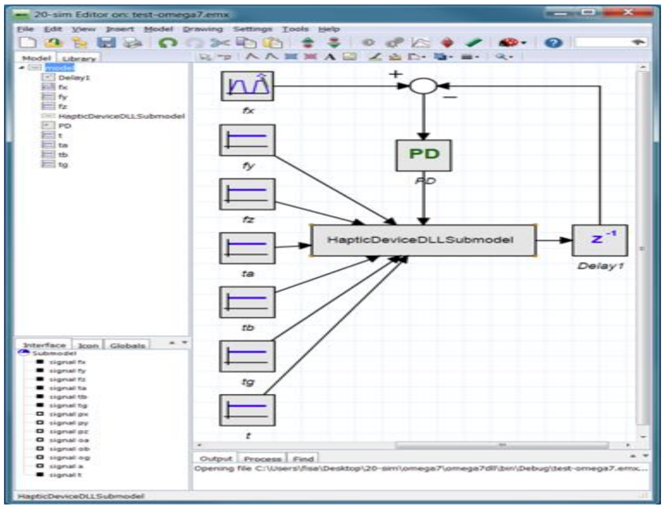

20-sim Editor on: test-omega7.emx
fx
fy
fz
ta
tb
tg
t
PD
HapticDeviceDLLSubmodel
z^{-1}
Delay1
Opening file C:\Users\fisa\Desktop\20-sim\omega7\omega7dll\bin\Debug\test-omega7.emx...

Fig. 3: The system block diagram considered as a case study.

the user's motion, detecting collisions between the user-controlled probe and the virtual objects, computing reaction forces in response to motion or contacts and exerting an intuitive force feedback on the user. This opens up to a variety of possible applications that are relevant for both research and application purposes.

In the future, the possibility of adding new functionalities to the proposed library may be considered. For instance, we may consider the possibility of exposing haptic devices physically connected to remote computers, to make them available through a computer network. All the new features can be logically layered without modifying the presented library.

References

[1] R. J. Adams and B. Hannaford, "Stable haptic interaction with virtual environments," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 3, pp. 465–474, 1999.

[2] Y. Yokokura and S. Katsura, "Adaptive motion-copying system based on real-world haptics," in *Proc. of the 36th Annual Conference on IEEE Industrial Electronics Society (IECON)*, 2010, pp. 1228–1233.

[3] S. D. Laycock and A. Day, "Recent developments and applications of haptic devices," in *Computer Graphics Forum*, vol. 22, no. 2. Wiley Online Library, 2003, pp. 117–132.

[4] M. De Pascale and D. Prattichizzo, "The haptik library," *IEEE Robotics & Automation Magazine*, vol. 14, no. 4, pp. 64–75, 2007.

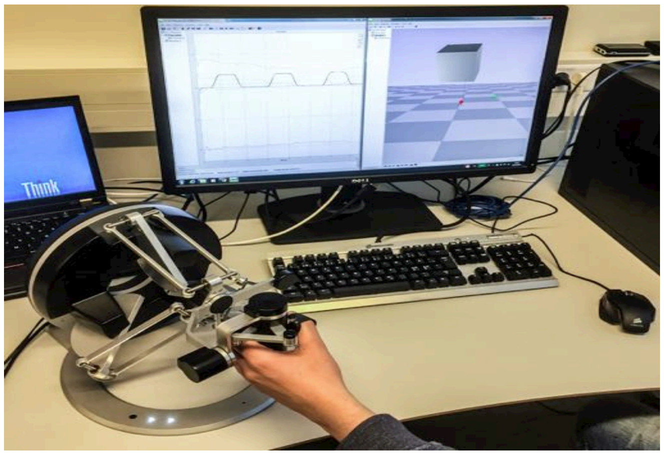
[5] Force dimension. (2015, February) "omega.7". [Online]. Available: http://www.forcedimension.com/products/omega-7/overview

[6] Controllab Products B.V. (2015, February) "20-sim". [Online]. Available: http://www.20sim.com/

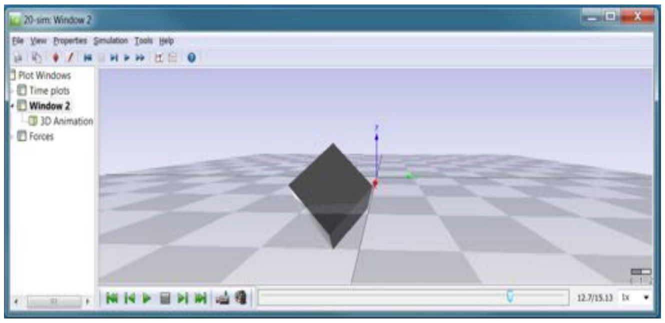
[7] A. J. Silva, O. A. D. Ramirez, V. P. Vega, and J. P. O. Oliver, "Phantom omni haptic device: kinematic and manipulability," in *Proc. of the IEEE Electronics, Robotics and Automotive Mechanics Conference (CERMA)*, 2009, pp. 193–198.

[8] B. Itkowitz, J. Handley, and W. Zhu, "The OpenHaptics toolkit: a library for adding 3D Touch navigation and haptics to graphics applications," in *Proc. of the 1st IEEE Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 2005, pp. 590–591.

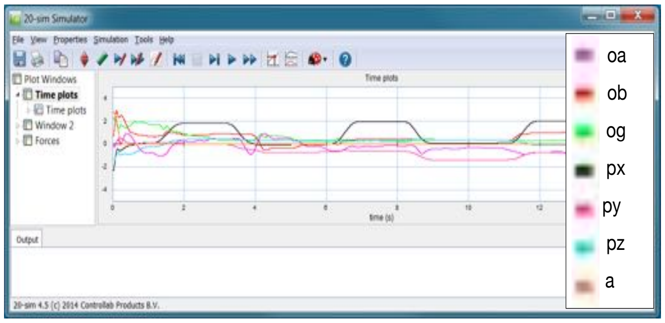
[9] D. Shreiner, M. Woo, J. Neider, T. Davis *et al.*, *OpenGL (R) programming guide: The official guide to learning OpenGL (R), version 2.1*. Addison-Wesley Professional, 2007.



(a)



(b)



(c)

Fig. 4: (a) The experiment setup, (b) the simple cube created as a virtual probe, (c) different time plots for the main system variables.

[10] Novint. (2015, February) "Falcon". [Online]. Available: http://www.novint.com/

[11] CHAI3D. (2015, February) "The Chai Library". [Online]. Available: http://www.chai3d.org/

[12] F. Sanfilippo, O. L. Osen, and S. Alaliyat, "Recycling a discarded robotic arm for automation engineering education," in *Proc. of the 28th European Conference on Modelling and Simulation (ECMS), Brescia, Italy*, 2014, pp. 81–86.

[13] K. S. Hale and K. M. Stanney, *Handbook of virtual environments: Design, implementation, and applications*. CRC Press, 2014.