# A Networked Haptic Embedded Controller

Carlo Alberto Avizzano, Emanuele Ruffaldi,
Daniele Leonardis and Massimo Bergamasco
PERCRO Scuola Superiore Sant'Anna, Pisa, Italy
Email: c.avizzano@sssup.it

***Abstract*—Recent innovation in embedded computing systems has allowed a new generation of smart devices and home appliances, such as tablets, smartphones and smartTVs, with embedded complete computing and networking capabilities for a more intuitive and functional operation. In this paper we present a novel approach for the control of haptic devices, where control, interfacing and networking capabilities are fully embedded into device electronics. The design and development is based on a dual core processor, separating the execution of low-level controls and high-level application layers, and a network controller supporting basic services and an embedded light web server interface. The approach allows us to fully operate the haptic device through network and the Web without the need of any additional hardware or software. In what follows we show the design guidelines, the physical implementation and the achieved results of the proposed device. The improved usability and flexibility of the system are presented through sample haptic rendering demonstrations.**

## I. Introduction

The idea of networking robots and haptic interfaces is not new at all. Internet based teleoperation[1], [2], has been a hot-topic for about twenty years. In these systems, however, the use of the network was intended roughly as one possible choice among the different available transport layers. Still these system required pre-shared knowledge among master-slave nodes, and specific driver setup at each side to make the teleoperation system operating correctly [3].

As an extension of the pure teleoperation approach, the use of Internet communication was applied to the sharing of haptic information. In 1998, Basdogan [4] and Wilson [5] experimented different types of collaborative haptic environments which used the network to share additional information about the physical behaviour of the haptic interaction.

In 2004, Ishibashi[6] proposed the simultaneous manipulation of digital objects through a set of cooperating devices (Sensable PHANToMs) that interact through separate workstations which were coordinated by a common server (named MAESTRO). Using four nodes connected in a local network this system achieves a typical communication delay close to 10-15 ms. Other approaches for integrating haptic interfaces on the Web where based on specialized plugins of the Web browser like in one of our previous works [7] and [8].

However, these systems get usually integrated through the support of one or more PC-based setups that provide the proper networking interface (e.g. [9]). According to this approach a computing server is dedicated to handshake the device haptic information through the Web.

All these approaches are based on the assumption that the haptic interface embeds a limited intelligence mostly to solve the local kinematic equations and to regulate the motor actions in response to the required force command sent from the application.

In this paper we describe how to support, within the low-level controller of an haptic interface, the required system integration for removing the need of an external workstation.

In our approach the embedded system integrates different service levels including: the low-level control of motor drivers; the high-level control of the whole robotic structure (such as kinematics, dynamics, calibration and weight compensation); the communication profiles to serve or command remote network clients/hosts; a Web server to facilitate the access, discovery, setup and control of the device; a set of JavaScript-based Web pages to control advanced behaviors and embedded on the remote device all the software and control loop required for controlling the haptic response. In particular the controller provides three types of haptic interation: motion control (position-velocity and position-force), haptic rendering of basic shapes and tele-operation.

The proposed controller can be operated through an autonomous Web interface, and it is smart enough to self-configure in a local network when multiple devices are present. For instance, as soon as two devices are available they may inter-connected each other and self configure themselves to operate as a master-slave tele-operation system.

The protocol adopted improves over existing peer-to-peer compliant system, like [10], [11], because it overcomes the need for a central processing unit and also provides a higher flexibility in the architecture, which in turn allows an unbounded number of devices to be connected together.

The paper is structured as follows: first the hardware setup is presented covering the haptic device and the computing system. Then the control architecture is covered. The fourth section discusses the software architecture. Finally the paper is closed by a demonstration setup and conclusions.

## II. Hardware setup

The controller has been tested on a pair of existing high-performance haptic devices: the GRAB interface [12]. Each device, shown in figure 1, is a 3 DOF (RRP) with spherical kinematic, designed with the specific aim of improving the transparency of the final haptic interface in terms of low perceived inertia, friction, backlash, and accurate force feedback [13].

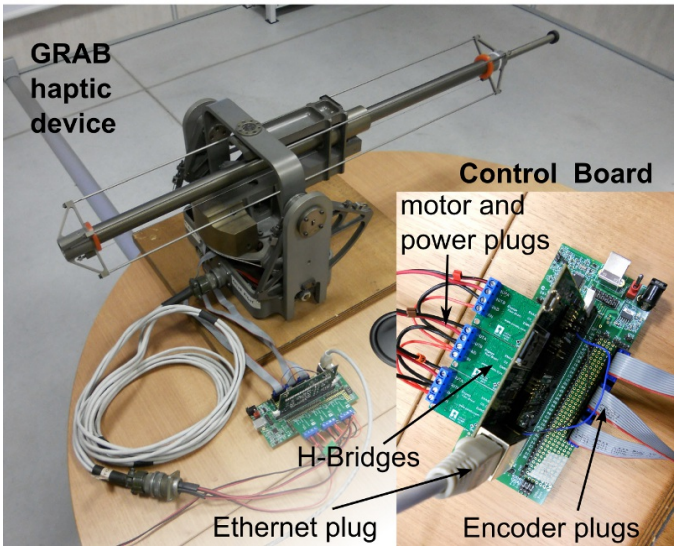



Fig. 1: Overview of the GRAB [13] haptic interface integrated with the proposed electronic controller.

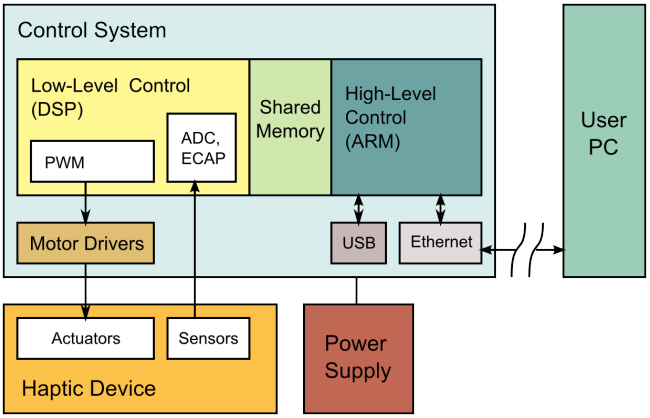



Fig. 2: Overview of the system modules and connections. Relevant control parameters are shared between the DSP and the ARM-CPU cores controller using a shared memory mechanism. The ARM controller implements a web server and provides to generate a local haptic response loop which reacts to networked commands.

The computing system is designed to separate the management of the low-level real-time haptic control from the networking and user interface. It is based on a recent Texas Instruments (TI) dual-core *heterogeneous* processor which allows us to decouple control from the high level logic: one core is a 150MHz 32-bit C28x DSP processor, specialized for control,(a 75MHz 32-bit Cortex-M3 ARM processor) provides to manage higher level services. The 32-bit ARM core provides a variety of communication interfaces including: Ethernet, USB full speed, SD memory card access and high speed serial communication. The 32-bit C28x core floating point DSP provides a single precision FPU together with motor control features, such as PWM signal generation, Quadrature Encoder interfaces, digital I/O, 12bit ADCs, and enhanced capture modules (eCAP). In the following we will refer to the two cores as the DSP and ARM-CPU.

The device is actuated by three Brushed PM-DC motors, which are directly controlled by the proposed controller through three separate H-Bridges ICs (ST VNH5019). The motor driving is achieved with a 20KHz PWM signal generated directly from the DSP pwm modules. The position sensing is performed through high resolution quadrature encoders (HEDL 5540) mounted directly on the motor shafts

Figure 2 shows an overview of the system modules and connections. In a general application the small-sized electronics can be wired close or embedded into the haptic device, resulting in a final system needing just a power plug and an Ethernet or USB cable for working.

## III. Control Architecture

In order to optimize the computing resources with respect to the required low-level real-time control tasks and high-level networking and interfacing tasks, we assigned to the DSP core the role of taking care of the sensor readings, actuators management and the low level control. The ARM-CPU core was instead delegated to high-level control and communication protocols.

This choice allowed us to ensure by task separation the safety and robustness required by the control loop, and the flexibility and computing resources required by the network interface. Tasks related to the control loop implemented on the DSP core, could be allocated and executed in a predictable way. Conversely, we delegated all unstructured operation, such as those deriving from the network servicing to the ARM-CPU core, whose response time is not affecting the hard real-time control of the DSP operation.

The DSP core was programmed to operate in different three switchable *low-level modalities*: position, velocity and force, modified with the selection of two reference systems: end-effector ($EE$) and joint ($q$). The conversion between EE and joint references was internally achieved through the computation of the transposed Jacobian for the EE reference[14]. The velocity and position control were implemented with two nested proportional-integral (P-PI) control loops.

The resulting control algorithm is shown in Figure 3 where $K_r$ represents the motor-to-join matrix that maps joint torques to motor ones. Safety algorithms were also implemented onto the DSP side: the control algorithm monitors that the absolute position/force/velocity is kept below a given threshold. A finite-state-machine structure allows to safely recover from protection faults and to manage specific startup procedures such as automatic position calibration and device reset.

The connection between the two cores has been obtained through a synchronized double-buffer shared-memory technique. High-level control from ARM-CPU to DSP cores is obtained by transferring the chosen modality and the related reference variables. The ARM core writes in the synchronized

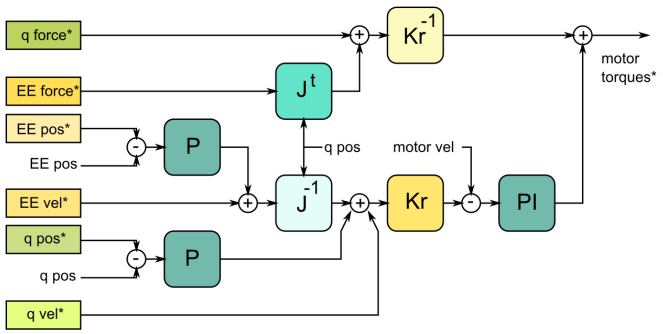



Fig. 3: Low-level control architecture

shared memory the requested modality, the parameters and the reference signals. This information is examinated by the finite-state machine to ensure coherence among different commands, then forwarded to the respective control loops. Once the information is gathered from the sensor accessible to the DSP core, the respective joint (end-effector) position, velocities are copied back in a different shared-memory area.

Three *high-level modalities* are detailed as follows describing their functionality, related low-level modality and communication:

1) *Velocity* control is used during early calibration phases in which the robot is controlled to move at constant speed toward some reference mechanical stops that help to uniquely identify the device posture
2) *Haptic Rendering*. The system renders a force at the EE which is computed on the basis of the actual EE position. In this case the low-level EE force control modality is enabled on the DSP core. A local server accepts EE force references in two ways: using Ajax with persistent connection [15] or through UDP datagrams. Using a local network an averaged closed-loop rate of 100Hz has been achieved in the first case , while 2 kHz has been achieved with direct UDP communication.
3) *Teleoperation* mode is used when two haptic devices are interconnected. Both devices share their position/velocity/force information to implement a wave variable based control as described also in previous work [3]. In absence of delays (less than 1 ms), the whole system behave ideally as if a virtual spring is connected between the EE of two devices, with a stiffness that can be set through position controller parameters.

## IV. Software Architecture

The architecture of the system is characterized by the distribution of the different services between the two heterogeneous cores of the control board and external computing nodes. Given the distributed computing architecture of the proposed system, the haptic rendering part can be computed on several locations (CPU) depending on the computing requirements, the network limitations and programming complexity.

Figure 4 shows three different approaches. In the first approach the rendering is performed inside the DSP, ensuring the shortest delay and most predictable performance, but with inherent limitations in the flexibility. This solution requires in fact that the force (position or velocity) profiles should be encoded within a rigid (and predictable) coding algorithm. In the second approach the force rendering is being computed on the ARM-CPU, and is provided as an additional Web server feature that closes the loop locally. In this approach the force delay is very short and the latency only depends on the shared memory synchronization.

The last approach is the typical configuration in which the force rendering is performed on a remote node external to the micro-controller. In such a case we may add the benefit of control versatility, but this come with networking bandwidth and latency limitation. Among the possible benefits of this approach we mention the possibility to interface the device control directly with Web browser application and without any plugin or driver.

Below we describe the services that are made externally available through the Ethernet interface and managed by the ARM-CPU core. Then we present the performance results on the UDP communication and finally the more sophisticated structure for the HTTP communication.

Two communication channels have been provided over the networked interface: HTTP over TCP and a simple protocol over UDP. The former provides the access to resources and to services embedded in the haptic interface, for configuration and for high-level control, while the latter provides support for low-latency inter-device communication. Figure 5 presents the overall organization of the communication, comprising the possibility of interconnecting several devices together.

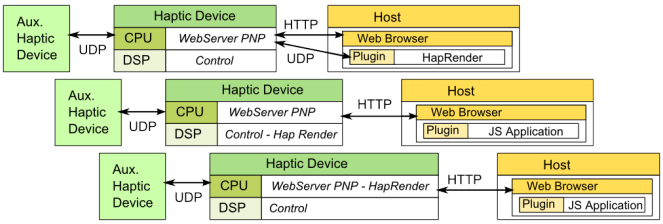



Fig. 4: Different possibilities for the haptic rendering configuration with the proposed architecture. Remote web-client rendering (top); DSP low level rendering (middle); and ARM-CPU Rendering (bottom).

### A. Services

The ARM-CPU masters all the services and manages the slave DSP core. We integrate on this core a set of task to control several activities such as: the handling of a complete file system, stored in an SD card, which contains the Web server pages; a basic USB flash-disk communication to manage, upload and update the Web server pages as if they were in a disk local to a remote PC; Ethernet low level handling; Web server with file serving from the SD card and

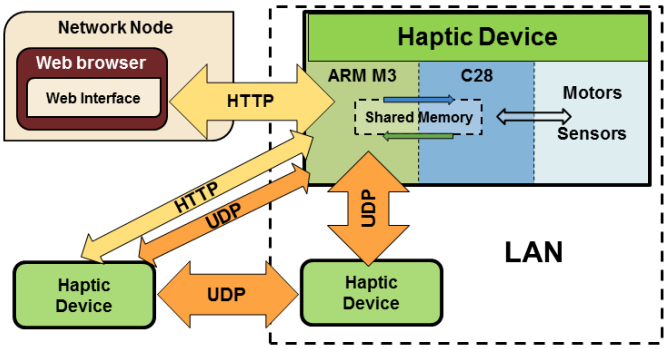



Fig. 5: Overall system and communication architecture. The haptic interface is described in terms of its connection with other haptic devices or with external computing nodes. The main communication channel with other devices is UDP, supported by HTTP for handshaking and communication.

AJAX interface; UDP server for handling low-latency haptic requests and information exchange with the DSP core. All of these services have been implemented on the ARM-CPU by using open source libraries, without any operating system layer. In particular lwIP [16] has been chosen for TCP and UDP communication over Ethernet and FatFS for SD card file system handling. A HTTP 1.1 Web server has been specifically developed over lwIP for this project and it will be contributed to the community being almost independent of the specificity of the controller interface.

## V. UDP Communication

When configured in tele-operation mode, the two GRAB devices communicate with each other using the UDP protocol thanks to the augmented performances: UDP packets (with error control over integrity) provide a fast and flexible way to implement multi-point communication as needed when more than two devices are working together. In our tests, using a 2GHz dual core PC running a Python server over Windows 7 32-bit OS, we managed to exchange a 50 bytes-packet payload at a roundtrip frequency of 2 kHz. All packets waited an acknowledgment answer before re-transmission, thus ensuring a minimum effective a bandwidth of about 100 kBs, with an averaged latency lesser than 500uS. These performances were achieved both when the board was connected peer-to-peer to the PC and when the connection was done through the lab Gigabit switched LAN. The protocol implemented over UDP, being the focus of direct communication between haptic interfaces is quite simple, and characterized by get/set pairs for controlling different variable.

### A. HTTP Communication

The lightweight Web server has two roles: static content serving and AJAX requests. The static content is served by accessing the files stored on the SD card of the system, thanks to the FAT file system handler. The AJAX service allows to read and write a large number of configuration parameters and real-time values like position of the end-effector, joints and forces. The AJAX service exposes as HTTP GET requests the possibility to read or write variables specified as query parameters. The write command allows not only to obtain confirmation of the writing operation but also to return the status of other variables. Each response is encoded using the JSON protocol (IETF RFC4627).

The implemented protocol allows any web client having scripting capabilities to interact with the controller. In such a way, the device itself provides to the client all the required software to run basic interaction programs.

A particular care has been given in the implementation of the protocols to minimize the computational cost and improve throughput. The CPU core has a very limited memory amount, hence we decided the server to implement the HTTP 1.1 protocol that allows multiple requests on the same TCP connection. As a result performances improve several times when delivering both static content and AJAX requests.

The controller reached 150Hz on the local network while serving AJAX requests with total packet request payload of about 200 bytes. If using the HTTP 1.0 connection the rate decreases 10-20Hz due to the TCP handshake and the overhead of the HTTP headers. In the design of the system we also evaluated other HTTP connection schemes like the recent WebSockets standard (IETF RFC6455), that, after an initial security handshake allow to exchange data in a bidirectional way by means of messages. The other option of HTTP 1.0 with MIME multi-part replace, as used by some M-JPEG servers, has not been tested.

JSON introduces an overhead of about 20 bytes on every request and response, but everything fits well in the MTU of the communication. JSON has been preferred over binary forms of JSON, or other protocols, to increase portability and human readability.

For assessing the quality of the communication when triggered from inside the browser we tested the execution of 1000 requests from the Google Chrome (release 32) browser scaling from one single connection to up to 6 concurrent connections. The results span from 58Hz in the best condition to 10 Hz in the worst case. This limitation is mostly due on how the browser manages its thread priority and request scheduling. So far, the data rate was not adequate to implement a complete force closed loop from inside the Web browser, and its use requires specific coupling strategies as described in the examples below.

### B. Integration within a client scripting

As said, using AJAX services the user may control the different operation modalities supported on the ARM-CPU core. At this stage, in order to cope with the reduced closed loop frequencies achievable within modern browser, the ARM-CPU core also implements basic haptic rendering of implicit entities: shapes render, damped springs and planes.

The use of these rendering functionalities allows a single AJAX request to exhibit complex (and robust) haptic rendering profiles which are internally at 1kHz between the two

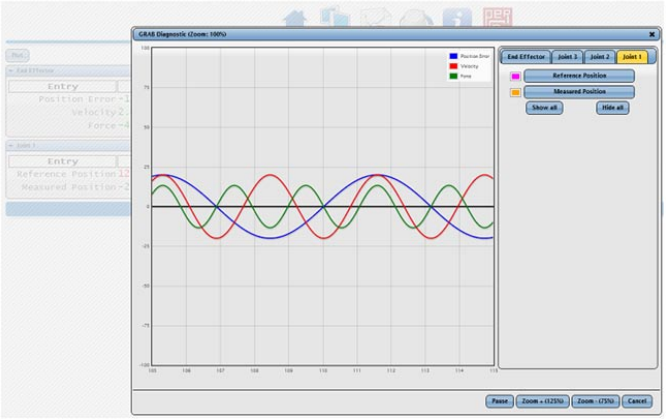


Fig. 6: Diagnostics Web interface showing example data sent from the device during a test

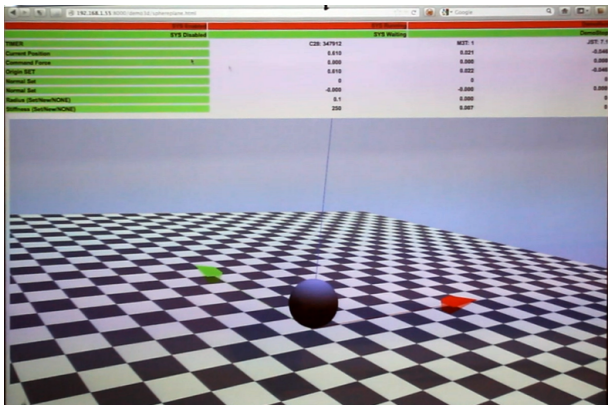
controller cores.

Using such functionalities we implemented several teaching demos on the embedded website. These demos take advantage of modern standards for fast graphics like HTML5 and WebGL, and the power of recent JavaScript libraries, like jQuery [17] , that allows rapid design development. In the next section we will show in practical examples how these technologies can be combined to achieve effective and interoperable haptic demonstration. The web server also hosts all the required web pages to monitor any device functionality or trigger specific interaction modality (such as the embedded teleoperation) as show in Figure 6.

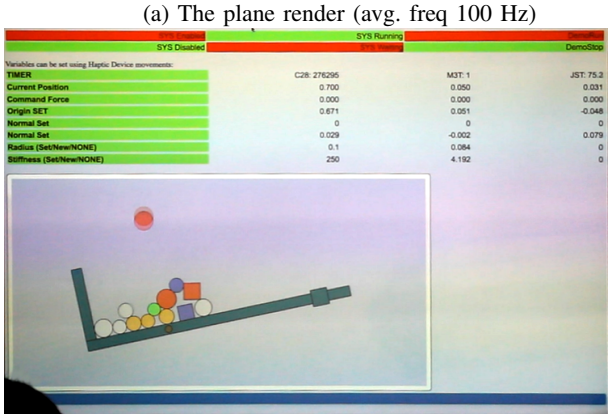
## VI. Demonstration

To assess the controller capabilities we developed two different demonstrations, both deployed on the web as HTML5 pages without plugins. First a haptic plane rendering where the graphical part is implemented by means of the HTML5 3D capabilities provided by the WebGL API, that is a JavaScript interface to the OpenGL ES 2.0 standard. Higher level functionalities have been obtained over WebGL by means of the Three.js library. In this demonstration the Web browser receives the haptic position and the forces generated by the implicit plane renderer to display updated information within the 3D canvas. Figure 7a shows a snapshot from the running system where the haptic position is represented by the sphere.

The second test application, shown in Figure 7b, is an adaptation of a physics demo based on the JavaScript port of the Box2D physics library. We adapted an existing non-haptic enabled application to identify the challenges of haptically extending it within our framework. The original demo was enhanced adding a coupling connection with the GRAB interface, handled through the implicit spring model. The demo presents a L-shaped object fixed at a joint over which several balls and boxes are falling. The haptic coupling to the EE was attached to right side of the shape (square marker) and used to add balance force within the physics. In this way the user was able to control the lever. By using a provided wrapper



(a) The plane render (avg. freq 100 Hz)



(b) the coupling contact (avg. freq 66Hz)

Fig. 7: Demonstration scenarios. In both cases the objects in the canvas move accordingly to user interaction. For debugging purposes haptic statistics from the device are shown on the top.

included at the beginning of the Web page, the changes of the original application were in the order of a dozen of lines of codes.

## VII. Conclusions and Future Work

We presented a novel type of embedded haptic controller that allows the integration in a networked environment. The approach allows to fully operate the haptic device through network and the Web without the need of any additional hardware or software. The controller embeds two different servers (UDP and HTTP), that directly exploit all the capabilities of the device without requiring any installation procedure, moreover, the overall system can be used from any operating system. The combined use of the Web interface with scripting technologies facilitates the development of teaching applications, and the use from any remote client such as cell-phones or tablets.

The controller presented here offers several opportunities of investigation in the way haptic interfaces can be programmed and interfaced with other devices and applications. First ,in the direction of mobility, by allowing to integrate wireless communication and haptic controller. This allows the creation and exploration of new haptic applications combining mobile technologies and grounded haptic devices, like, for example,

augmented haptics. The JSON approach could allow the system to interoperate with more complex robotic setups as provided by the Robotics Operating System (ROS) through the rosbridge [18], that maps ROS messages to JSON. Second, in the direction of interoperability, allowing the creation of cooperative devices that increase discoverability through the adoption of standard protocols such as Universal Plug and Plan (UPnP) or multicast DNS (mDNS). Third, the aspect of security, to prevent or exploit the activation of the application in relation to specific access algorithms. Finally, in the domain of haptic rendering to support more sophisticated implicit rendering algorithms that overcome the present performance limitation in current browser technologies.

A video of the working system has been uploaded at this location: http://youtu.be/WULoIM54s9Y.

## REFERENCES


[1] K. Brady and T. Tarn, "Internet-based remote teleoperation," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 1. IEEE, 1998, pp. 65–70.

[2] K. Goldberg, M. Mascha, S. Gentner, N. Rothenberg, C. Sutter, and J. Wiegley, "Desktop teleoperation via the world wide web," in *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, vol. 1. IEEE, 1995, pp. 654–659.

[3] M. Satler, C. Avizzano, A. Frisoli, P. Tripicchio, and M. Bergamasco, "Bilateral teleoperation under time-varying delay using wave variables," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 4596–4602.

[4] C. Basdogan, C. Ho, M. Slater, and M. Srinivasan, "The role of haptic communication in shared virtual environments," 1998.

[5] J. Wilson, R. J. Kline-Schoder, M. A. Kenton, and N. Hogan, "Algorithms for network-based force feedback," 1999.

[6] Y. Ishibashi, T. Hasegawa, and S. Tasaka, "Group synchronization control for haptic media in networked virtual environments," in *Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2004. HAPTICS'04. Proceedings. 12th International Symposium on*. IEEE, 2004, pp. 106–113.

[7] E. Ruffaldi, A. Frisoli, M. Bergamasco, C. Gottlieb, and F. Tecchia, "A haptic toolkit for the development of immersive and web-enabled games," in *Proceedings of the ACM symposium on Virtual reality software and technology*. ACM, 2006, pp. 320–323.

[8] M. O'Malley and S. Hughes, "Simplified authoring of 3d haptic content for the world wide web," in *Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2003. HAPTICS 2003. Proceedings. 11th Symposium on*. IEEE, 2003, pp. 428–429.

[9] P. Tripicchio, E. Ruffaldi, C. Avizzano, and M. Bergamasco, "Virtual laboratory: a virtual distributed platform to share and perform experiments," in *Haptic interfaces for virtual environment and teleoperator systems, 2008. haptics 2008. symposium on*. IEEE, 2008, pp. 311–318.

[10] M. Panzirsch, J. Artigas, A. Tobergte, P. Kotyczka, C. Preusche, A. Albu-Schaeffer, and G. Hirzinger, "A peer-to-peer trilateral passivity control for delayed collaborative teleoperation," *Haptics: Perception, Devices, Mobility, and Communication*, pp. 395–406, 2012.

[11] P. Malysz and S. Sirouspour, "Trilateral teleoperation control of kinematically redundant robotic manipulators," *The International Journal of Robotics Research*, vol. 30, pp. 1643–1664, 2011.

[12] M. Bergamasco, C. Avizzano, A. Frisoli, E. Ruffaldi, and S. Marcheschi, "Design and validation of a complete haptic system for manipulative tasks," *Advanced Robotics*, vol. 20, no. 3, pp. 367–389, 2006.

[13] M. Bergamasco, F. Salsedo, M. Fontana, F. Tarri, C. Avizzano, A. Frisoli, E. Ruffaldi, and S. Marcheschi, "High performance haptic device for force rendering in textile exploration," *The Visual Computer*, vol. 23, no. 4, pp. 247–256, 2007.

[14] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2008.

[15] J. Garrett *et al.*, "Ajax: A new approach to web applications," 2005.

[16] A. Dunkels, "Design and implementation of the lwip tcp/ip stack," *Swedish Institute of Computer Science*, vol. 2, p. 77, 2001.

[17] A. Freeman, *Pro jQuery*. Apress, 2012.

[18] C. Crick, G. Jay, S. Osentoski, B. Pitzer, and O. C. Jenkins, "Rosbridge: Ros for non-ros users," in *Proceedings of the 15th International Symposium on Robotics Research*, 2011.