2013 IEEE RO-MAN: The 22nd IEEE International Symposium on Robot and Human Interactive Communication
Gyeongju, Korea, August 26-29, 2013

A flexible framework for mobile based haptic rendering

Emanuele Ruffaldi, Massimo Satler, Gastone Pietro Rosati Papini and Carlo Alberto Avizzano[1]

Abstract— **The present paper discusses a framework for creating flexible rehabilitation exercises on a mobile haptic interface. The approach is based on a combination of high level interactive scripting with a flexible real-time haptic control algorithm. The control problem has been decomposed into primitives such that the real-time issues are managed at low level on the mobile system, while a high level control can interact with the user through powerful and flexible scripting. The control architecture is flexible enough to operate haptic rendering on wireless mobile devices. Overall the system allows to create a large variety of rehabilitation exercises that can all share the same methodology for their parametrization and assessment. The rationale of the design and the implementation of the different control level will be provided. Design experiments, based on a rehabilitation game scenario will also be discussed.**

I. INTRODUCTION

There is a large interest in designing new haptically enabled applications in particular for the domains of rehabilitation robotics and education. These applications allow to support the cognitive and sensorimotor rehabilitation of subjects. Unfortunately the cost of the devices and the complexity of the haptic rendering aspects have so far posed limitations in the features of the applications. Due to their therapeutic nature these applications need also to be integrated with performance measures that can be used along with the training protocols, or on-line for adapting the complexity of the task.

In the rehabilitation task, as in many general haptic applications, there are two main approaches for providing feedback to the user while performing a task: virtual fixtures and shared controls [13]. Virtual fixtures [4] are haptic rendering features that allow to specify geometrical constraints for subject motions. An exercise based on virtual fixtures allows to impose specific paths of motion that are considered beneficial for the rehabilitation task.

Shared control is another complementary feature that allows to manage the amount of help provided by the robotic system to the subject while executing a task [7]. The role of shared control for rehabilitation is generally investigated by Carmichael et al. [2] modeling requirements and associated assistance during the execution of a task.

While these concepts are well known to experts in robotics and haptics, there is a need for mapping them to the rehabilitation protocols, and in general to the design of a high level application for exercising a subject. This paper introduces a haptic rendering approach and a representation of geometrical features for designing a variety of haptic exercises. The paper addresses the problem of designing this kind of rehabilitation exercises by introducing a framework for designing, testing and executing these applications. The framework is described from the low level control up to the high level scenario editor analyzing the primitive features that provide the trade-off between behavior flexibility and the easiness of design for non-experts.

[1]Perceptual Robotics Laboratory, TeCIP Institute, Scuola Superiore Sant'Anna, Pisa, ITALY

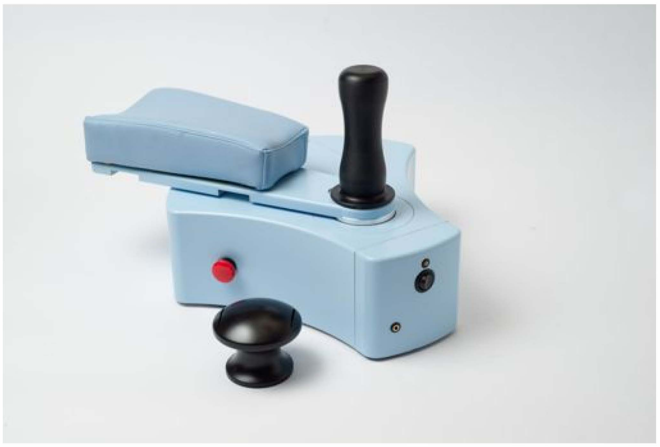


Fig. 1: MOTORE: MObile roboT for upper limb neurOrtho REhabilitation. The picture shows the rehabilitation tool with the two handles designed for different kind of patients.

Several types of interfaces are employed in rehabilitation, from desktop systems to exoskeleton [8], but there is also a new class of systems based on mobile platforms with haptic feedback generated through the wheels as proposed by the authors [1], [12]. MOTORE, the reference device of this paper, depicted in Figure 1. It is worth mentioning that some of the design element proposed by this work can be applied to passive mobile platforms [6], [11], [10].

The remaining of the paper is organized as follow. The following section discusses the high-level overview of the framework in Section III. Section IV introduces the low-level control scheme. Then Section V introduces the representation of trajectories. Examples of scenarios are presented in Section VI ad finally Section VII provides a brief evaluation of the force rendering.

II. HIGH LEVEL DESIGN

The framework proposed by the paper takes a global approach for dealing with exercises in training protocols, as depicted in Fig. 2. The system manages a set of exercises called scenarios, that are parametric with respect to execution condition, e.g. subject handiness, and to complexity and duration. Therapists manage the activation of these scenarios

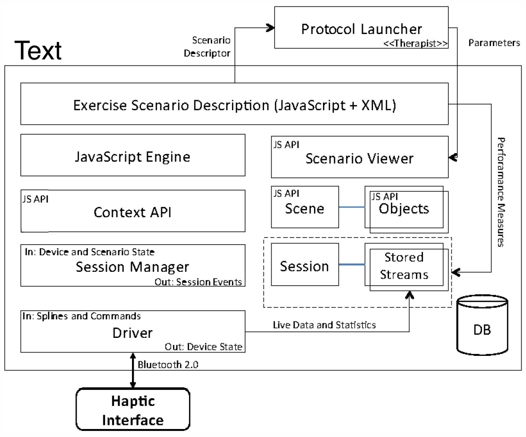


Fig. 2: High Level System presenting the functional blocks of the system.

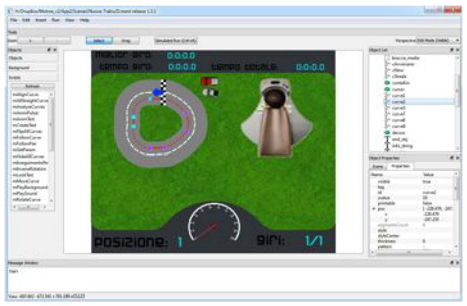


Fig. 3: Editor Screenshot

based on the training protocol and direct assessment of the subjects' performance.

Scenarios are composed by three elements: visual and haptic entities, behavior of the scenario, and performance parameters. The entities are expressed by a XML representation and manipulated using a visual Editor depicted in Figure 3. Behavior is represented in JavaScript allowing easy editing and flexibility. Finally the performance indicators are a means used for sharing the assessment of exercise performance across different executions: they comprise the amount of energy exchanged by the device and the user, the statistics about the forces and errors in trajectory, the amount of help received by the device, in addition to other measures that are specific of the given scenario.

The key contribution of the paper is the design of the entities employed in the haptic rendering and they are discussed in a bottom up approach in the following sections.

III. CONTROL AND RENDERING ALGORITHMS

The device control structure has been already accurately discussed in our previous works, [12], [1], here, after a short description of the basic control loops, we will focus on the haptic rendering mechanism and on the control modalities provided by the device.

A. Motion Control

The platform movement is achieved by mean of three identical DC-brushed motors which allow the rotation and the movement in any direction thanks to their configuration on the device chassis. Each motor rotation is measured by an optical encoders (2000 counts per revolution) that, given the robot wheel radius ensure an overall position sensitivity less than $1 \mu m$ per count. The absolute position of the device (w.r.t a fixed reference frame) is achieved by fusing the odometry information with the position measurement coming from an optical pen embedded inside the robot itself.

The close-loop control system is composed by three nested loops. At the inner level the current control loop manages the PWM duty cycle of the drivers to control the currents flowing within the motors. The current regulation is closed on the velocity controller that enables the platform to follow the commanded desired velocities. This loop has a base rate of 1kHz. Finally, the slowest loop (50 Hz refresh rate) provides position estimation correction to the odometry estimation by means of an ad hoc Extended Kalman Filter.

The choice of using an internal velocity controller instead of a force controller does allow us to implement safety issues directly at very low-level since the wheels velocity signals are available, accurate and stable in all the control phases. Conversely, position information (required by an impedance controller) is not so accurate due to wheel slipping in combination with the latency of the fusing algorithm.

B. Force Rendering

Traditional haptic rendering algorithms rely on the impedance control. Impedance controllers require an internal force rendering subsystem. The distance between the device position ($\vec{D}$) and a desired estimated position ($\vec{P_e}$) provides the input to generate a proportional force to be rendered ($\vec{F_r} = -Z(\vec{D} - \vec{P_e})$), where Z is the control impedance. Conversely, admittance controllers use the force read by a force sensor ($\vec{F_S}$) to determine the device linear velocity ($\vec{V_r}$) through integration ($\vec{V_r} = \vec{F_S}/(Ms + b)$)[1] using an apparent mass (M) and a nominal viscosity (b).

Even being most suited for an admittance controller, we rearranged the force rendering algorithm to implement impedance and admittance controllers that may operate simultaneously on different and orthogonal axes:

$$\begin{cases} V_{\vec{d},i} = (k_1(\vec{P_e} - \vec{D}) + k_2\vec{F_S}) \cdot \vec{d} \\ V_{\vec{d},a} = (\vec{F_S} \cdot \vec{d})/(Ms + b) \\ V_{\vec{d}} = \gamma V_{\vec{d},i} + (1-\gamma) V_{\vec{d},a} \end{cases} \quad (1)$$

where $\vec{d}$ represents any control direction, $\gamma \in [0, 1]$ is the control selector that decides if the direction $\vec{d}$ is subject to impedance or admittance control type and k_1, k_2 are control gains to modulate the filter input signals. If $\gamma = 0$ the control over direction $\vec{d}$, reduces to an admittance controller, whereas if $\gamma = 1$, the control over the same direction resembles an impedance controller. It is easy to verify, that steady state equilibrium ($V_{\vec{d},i} = 0$) along direction $\vec{d}$, is achieved when $\vec{F_S} \cdot \vec{d} = -k_1/k_2(\vec{P_e} - \vec{D})$. In this case we have an equivalent impedance of $Z = k_1/k_2$.

[1]Here "s" denotes the Laplace variable.

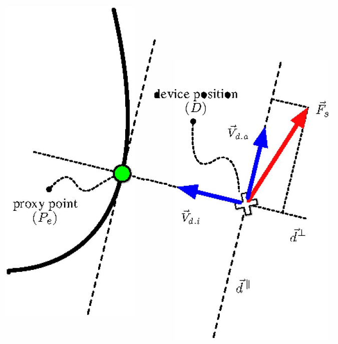


Fig. 4: Control law representation. The control algorithm generates a reference velocity along the orthogonal direction using an impedance like control law and a reference velocity along the parallel direction using an admittance control law.

C. Haptic Rendering

The haptic rendering has been organized to support different control modalities, such as:

1) free move: the device appears as a viscous virtual mass
2) point attraction/potential fields: the device is attracted to a point in space
3) trajectory constraint: the device is constrained to move over a given path

The first two behaviors may be implemented by setting the selectors γ equal to 0 and 1, respectively. The last behavior, i.e. the trajectory constraint, is achieved selecting two orthogonal directions $(\vec{d}^{\parallel}, \vec{d}^{\perp})$ and mixing the control modes $(\gamma \in (0,1))$ in order to obtain an admittance profile along the trajectory and an impedance one in the orthogonal direction, see Figure 4.

To implement such a profile and to decouple between low-level (embedded) control and high-level (application driven) control, we decided that each motion stroke is associated to a trajectory whose geometry is determined through a spline, and the associated admittance and impedance profile is passed as a parameter.

D. Haptic Protocol

The decoupling between the low-level control, and high-level control is achieved at spline description level. The low-level control operates at 1kHz frequency, while the high-level control may operate at any frequency (even asynchronously).

When the high-level control wishes to program a certain profile, it sends to the low-level control a tentative packet containing the following information:

Spline geometry	2 parameters
Admittance/Impedance properties	4 parameters
Operating modes	3 parameters
Guards	4 parameters

Spline geometry: whenever each new candidate spline is proposed to the low-level control, the device checks the spline and re-adapts it in order to match the spline boundary conditions with the actual platform's position. If required this operation may also imply a re-sizing of the spline geometry. Eventually, in the case of failure on the spline curvature, the low-level controller can refuse to accept the spline.

Admittance/Impedance properties: these parameters control the mass, the viscosity and the stiffness values as defined in (1). To avoid real-time exceptions, only a finite number of values are available.

Operating and modes: these parameters defines how the device behaves at the beginning, during the execution and at the end of the rehabilitation exercise. The switch between these phases is guided through a set of time/length guards described below. In combination with these guards, the low-level controller decides which feedback policy to use to help the user following the given trajectory.

Guards: four different types of guards have been implemented in the low-level controller. These guards monitors the time spent by the user in each phase of the spline tracking, as well as the overall length of the performed trajectory with respect to the length of the reference. In particular, start, length and running guards switch the device from one of the active driving modes $(\gamma \neq 0)$ to the passive mode $(\gamma = 0)$, where the proxy point is moved at constant speed (see sec. IV-C for details).

To complete the protocol, the low-level controller regularly informs the high-level control about relevant performance data including: percentage of spline completion, elapsed spline time, statistical data on force, motion and work performed by both the user and the machine.

Using such information the high-level controller may decide when to download a new candidate spline to the low-level controller.

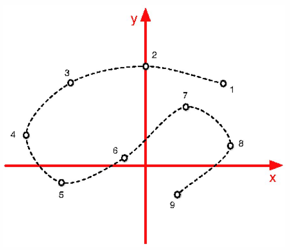
This protocol ensures that in each condition the low-level controller is intrinsically stable with clearly defined instructions of what to perform even in the absence of surveillance from the high-level control. Missing or delayed information coming from the high-level control will result in decreasing the quality of servicing, but never falls in unstable conditions.

IV. Trajectory Constraint

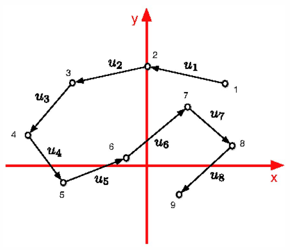
In order to implement the control strategy previously described, some trajectories have been defined. Further, in order to assure the system portability as well as its simple use, these trajectories have been coded into the device itself, i.e. in the DSP flash memory. This section explains how the base elements have been designed and how a generic path can be obtained using such base elements.

A. Spline Definition

The device embeds 20 base elements which are combined together to set up the desired geometry which will be used to constrain the user movement according to the control law given in (1). Each base element represents a planar curve in the bi-dimensional xy-plane and it is defined by a sequence of eight control points (x_i, y_i). Such points are interpolated by a cubic spline with given end conditions. In particular, we set a starting and an ending direction as end conditions for the interpolating process. In this way we can assure



(a)



(b)

Fig. 5: Bi-dimensional spline example. (a) points definition, (b) curve parameter definition.

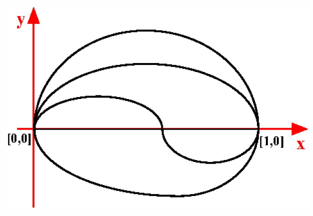


Fig. 6: Base elements definition examples. The figure shows four kinds of base elements: a straight line, three arches and a "sine" like element. Each element has been designed in the $[0, 1]$ range along the x-axis.

the starting and the ending tangent curve which allows to concatenate the base elements without shape discontinuities. The interpolation process employs cubic splines of class C^3 [3] which assures the continuity of the trajectory as well as the continuity of its first and second derivative.

The interpolation process can be formalized of follow: given n control points (x_i, y_i) and the curve parameter u, we are looking for the cubic spline which interpolates the given control points according with the starting and ending slope:

$$p(u) = (p_x(u), p_y(u)) \tag{2}$$

where $p_x(u)$ and $p_y(u)$ are one dimensional spline which interpolates (u_i, x_i) and (u_i, y_i), respectively and $i = 1, 2...n$.

Whichever bi-dimensional shape can be defined considering the sequence of control points, and defining between each pair of points two cubic curves being function of u that varies from 0 to 1 between the control points, see Figure 5(b):

$$\begin{cases} x = a_{x_i} + b_{x_i} u_i + c_{x_i} u_i^2 + d_{x_i} u_i^3 \\ y = a_{y_i} + b_{y_i} u_i + c_{y_i} u_i^2 + d_{y_i} u_i^3 \end{cases} \tag{3}$$

Repeating the approach for each pair of control points, leads to the desired cubic spline, see Figure 5(a).

In MOTORE, each base element has been defined spanning eight control points in the plane constrained by the $[0, 1]$ range along the x-axis and it consists in simple shape which has been oriented along the x-axis, see Figure 6. Defining the appropriate functions $x = f(u)$ and $y = g(u)$, we obtained the eight control points and hence the polynomial coefficients, obtained according with (2), has been stored in the DSP (64 floats for each base element). During the design phase, the use of cuspid or high-curvature strokes has been avoided in order to provide good references to MOTORE, i.e. smooth curve without discontinuities in the velocity (first derivative) and acceleration profile (second derivative).

B. Trajectory definition

Point-based trajectories allow to scale, translate and rotate the stored geometries with low computational cost, and hence it is possible to fit with the task requirements adjusting in real-time the base elements previously defined. In particular, based on the actual device position, the desired final position, and the chosen base element, the low-level control calculates in real-time the new geometry using the original spline coefficients, a roto-translation transformation matrix and a scaling factor. Moreover an additional safety check has to be performed, while manipulating the base elements. Scaling the base elements requires that the minimum curvature of the trajectory is not going below a given threshold that has been carefully defined by experimental trials with the device. Hence, the low-level control checks in real-time if the desired destination is feasible based on the actual device position and the chosen base element.

The base elements can also be combined each other in order to obtain arbitrary and more complex trajectories (Figure 8). In MOTORE, an internal buffer stores the next trajectory to be accomplished as soon as the device reaches the end of the current trajectory. In this way the device assures the continuity of the exercise even in presence of communication isues with the hosting PC. While connecting the base elements, the continuity of the trajectory in the contact points between two adjacent geometries has to be explicitly assured. This check is performed before sending the request for new trajectory to the device and it is done by the trajectory editor witch runs on the host computer.

C. Proxy Algorithm

In order to implement the control law defined in (1) and set-up the rehabilitation protocol, it is required to know which is the trajectory point closest to the device (the proxy point). It is also required to track this point on the trajectory itself while the device moves according to the pre-programmed movement law or due to the patient driving forces. To track the proxy point a longitudinal curve coordinate ($l \in [0, 8]$) is defined. The integer part of l identifies the spline stroke to be consider (the sub-set of spline to be selected), whereas the decimal part of l identify the u value which is used according with (3) to obtain the proxy point.

The proxy point is evaluated in real-time using a 2D Newthon-Raphson method on the spline curve. The result is used as attractive point in the impedance filter along the orthogonal direction. The updating law for the longitudinal coordinate l is defined by the following smoothed gradient algorithm:

$$l(k+1) = l(k) + \alpha \left(\left[\frac{\vec{d}(k)}{|\vec{d}(k)|} \right]^T \frac{\vec{\nabla}(k)}{|\vec{\nabla}(k)|} \right) \tag{4}$$

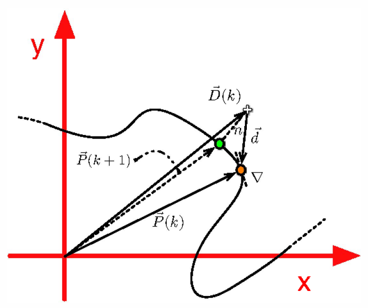


Fig. 7: Graphical representation of the proxy algorithm. $\vec{D}$ is the device position vector, $\vec{P}$ is the spline position vector, $\vec{d} = \vec{P} - \vec{D}$, $\vec{\nabla}$ is the local curve gradient whereas n is the minimal distant between the device position and the trajectory.

where, according with Figure 7, $\vec{d}$ has been defined as the distant vector between the device position and the proxy point, $\vec{\nabla}$ is the local curve gradient and $\alpha \in [0, 1]$ is the tracking speed gain. This latter gain (α) is chosen as a trade-off between performance and possible stability issues that could be associated to curvature changes. The local curve gradient $\vec{\nabla}$ is symbolically computed differentiating (3) as follow:

$$\vec{\nabla} = \begin{bmatrix} b_{x_i} + 2c_{x_i}u_i + 3d_{x_i}u_i^2 \\ b_{y_i} + 2c_{y_i}u_i + 3d_{y_i}u_i^2 \end{bmatrix} \tag{5}$$

Figure 7 shows on a generic trajectory the curve coordinate evolution over the time. The orange point is the curve coordinate at (k)-instant and the green point represents the expected curve coordinate at $(k + 1)$-instant. $l(k)$ comes from the previous proxy algorithm step, whereas $l(k + 1)$ is obtained projecting on the curve the device position along the minimal distant direction.

In some specific guidance conditions (see "Operating and guidance mode" in section III-D), the curve coordinate l may also be assigned with a prefixed update law which drives the device to follow the spline at constant speed (v):

$$l(k+1) = l(k) + \frac{v}{|\vec{\nabla}(k)|} \tag{6}$$

This proximity algorithm is affected by cases in which the closest point to MOTORE is not unique, as when the device is in the center of a circular trajectory. The possible instability caused by multiple solution can be solved by carefully designing the trajectory and by limiting the distance to the reference trajectory.

V. SCENARIOS

The framework adopts a combination of declarative and behavioral structures for supporting a large set of scenarios, from basic repetitions of trajectories to complex games involving haptic tasks. In particular, in this paper we highlight two typical task in rehabilitation settings: trajectory following and reaching. The trajectory following scenario is actually a family of scenarios in which different trajectories

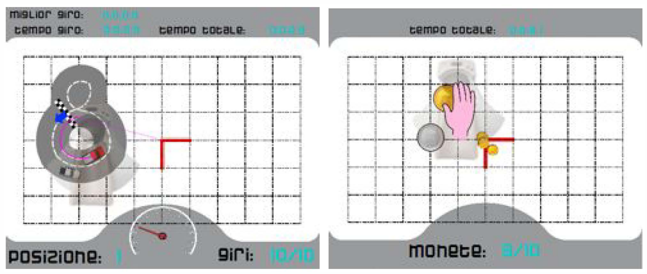


Fig. 8: Left: trajectory following task in the pursuit configuration, where the subject drives the red car and the opponent has the blue one. Right: reaching scenario in which the subject virtual hand has to collect coins that are placed in a radial way.

can be experienced, with similar shapes and different sizes and placements in the workspace. A variant of this scenario has been designed to make the task more engaging and for this reason the subject has the duty of trying to reach an opponent car as shown in Figure 8 (left). The opponent car behavior is designed to limit the maximum speed of the subject and give some chances to the subject.

A typical reaching task is managed by the scenario in Figure 8 (right) in which the user cursor is represented by a hand that has to collect coins. The coins are placed in a radial form and the motion from the center to each coin is constrained by a straight path.

These scenarios are characterized by the combination of purely visual features, namely sprites, and visual-haptic entities that correspond to exercise trajectories. All these entities can be manipulated in the visual editor that supports the direct testing and debugging of the scenario with the haptic interface. The trajectories in particular can be manipulated by modifying the control points, by changing the basic spline, and the editor supports the enforcement of specific design requirements like maximum forces, curvature, or adjacency between segments. The editor is also characterized by a representation of the device workspace allowing to verify in advance mobility problems.

In terms of implementation, scenarios run inside a custom C++ based Qt (Digia) application extended with JavaScript for the behavior of the scenarios. This choice has been done for keeping under control the update rate and the performance of the connection with the device. This system could be enhanced by moving the user interface and application logic to HTML5 provided that performance criteria are met.

VI. EVALUATION

The real-time proxy point estimation and the driving capability of the device have been tested on several experimental tests with patients. In the following two results concerning a trajectory following exercise are presented.

During the exercise it has been asked to the patient to follow the given reference trajectory in counter clock direction, see Figure 9. The reference trajectory is represented in solid red line, with red star as the connection points between the elementary splines. The green circles represent device

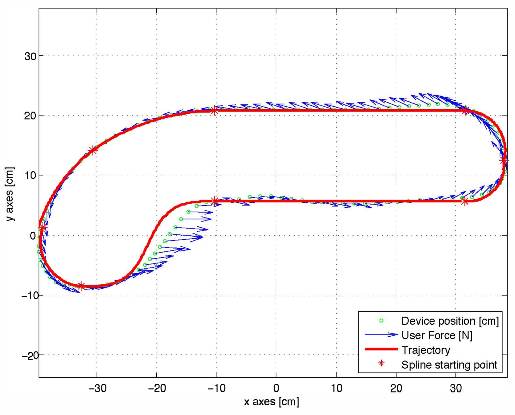


Fig. 9: Trajectory following example. The figure shows the reference trajectory, the actual device position and the force exerted by the patient.

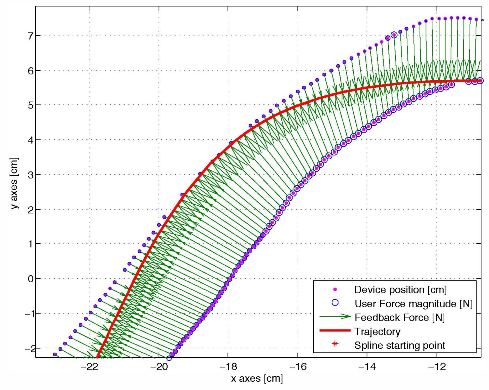


Fig. 10: Force feedback provided during the trajectory following exercise. The reference trajectory is in red, device forces are green arrows, the path, during two passages, is in magenta, user force as blue circles.

positions while the blue arrows show the force applied by the user on the robot handle.

Figure 10 is a detailed view of the previous trajectory that shows the feedback provided to the user. The sequence of device positions are represented with magenta points, whereas the corresponding blue circles are the magnitude of the forces exerted by the user (equivalent to the arrow magnitude of Figure 9). The green arrows represent the force feedback provided to the user as driving aid by the impedance control which is strictly related with the proxy point estimation, as explained in sec. III-D. The result proves the effectiveness of the proxy algorithm that in real-time is able to follow the closest point to the device. This assure the right computation of the normal and parallel direction which are the requirements for the stability of the implemented control law.

VII. Conclusions

The approach proposed in this paper allows to design different types of exercise scenarios while taking into account the peculiarities of the haptic rendering required by a mobile wireless haptic interface. There are several directions of improvement as the integrated presentation of exercise performance and haptic oriented enhancements to the editing environment. The system is currently tested in real rehabilitation settings and the effectiveness of the design will be discussed in future work. The paper discussed only motor based scenarios but others involving both cognitive and motor tasks are being tested.

Acknowledgment

The present work has been developed in collaboration with Humanware s.r.l for the control design of a robotic-rehabilitation device.

References

[1] CA Avizzano, M. Satler, G. Cappiello, A. Scoglio, E. Ruffaldi, and M. Bergamasco. Motore: A mobile haptic interface for neuro-rehabilitation. In *RO-MAN, 2011 IEEE*, pages 383–388. IEEE, 2011.

[2] Marc G Carmichael and Dikai Liu. A task description model for robotic rehabilitation. In *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pages 3086–3089. IEEE, 2012.

[3] E. Catmull and R. Rom. A class of local interpolating splines. *Computer aided geometric design*, 74:317–326, 1974.

[4] David Feygin, Madeleine Keehner, and R Tendick. Haptic guidance: Experimental evaluation of a haptic training method for a perceptual motor skill. In *Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2002. HAPTICS 2002. Proceedings. 10th Symposium on*, pages 40–47. IEEE, 2002.

[5] A. Formaglio, D. Prattichizzo, F. Barbagli, and A. Giannitrapani. Dynamic performance of mobile haptic interfaces. *Robotics, IEEE Transactions on*, 24:559–575, 2008.

[6] Rajibul Huq, Elaine Lu, Rosalie Wang, and Alex Mihailidis. Development of a portable robot and graphical user interface for haptic rehabilitation exercise. In *Biomedical Robotics and Biomechatronics (BioRob), 2012 4th IEEE RAS & EMBS International Conference on*, pages 1451–1457. IEEE, 2012.

[7] Yanfang Li, Joel C Huegel, Volkan Patoglu, and Marcia K O'Malley. Progressive shared control for training in virtual environments. In *EuroHaptics conference, 2009 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. World Haptics 2009. Third Joint*, pages 332–337. IEEE, 2009.

[8] Luis I Lugo-Villeda, Antonio Frisoli, Edoardo Sotgiu, Giovanni Greco, and Massimo Bergamasco. Clinical vr applications with the light-exoskeleton for upper-part neurorehabilitation. In *RO-MAN, 2010 IEEE*, pages 1–6. IEEE, 2010.

[9] N. Nitzsche, U.D. Hanebeck, and G. Schmidt. Mobile haptic interaction with extended real or virtual environments. In *Robot and Human Interactive Communication, 2001. Proceedings. 10th IEEE International Workshop on*, pages 313–318. IEEE, 2001.

[10] A. Peattie, A. Korevaar, J. Wilson, B. Sandilands, X. Chen, and M. King. Automated variable resistance system for upper limb rehabilitation. In *Proceedings of Australasian Conference on Robotics and Automation 2009*, pages 2–4, 2009.

[11] J.C. Perry, H. Zabaleta, A. Belloso, C. Rodriguez-de Pablo, F.I. Cavallaro, and T. Keller. Armassist: Development of a functional prototype for at-home telerehabilitation of post-stroke arm impairment. In *Biomedical Robotics and Biomechatronics (BioRob), 2012 4th IEEE RAS & EMBS International Conference on*, pages 1561–1566. IEEE, 2012.

[12] M. Satler, C.A. Avizzano, and E. Ruffaldi. Control of a desktop mobile haptic interface. In *World Haptics Conference (WHC), 2011 IEEE*, pages 415–420. IEEE, 2011.

[13] Govindarajan Srimathveeravalli, Venkatraghavan Gourishankar, and Thenkurussi Kesavadas. Comparative study: Virtual fixtures and shared control for rehabilitation of fine motor skills. In *EuroHaptics Conference, 2007 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. World Haptics 2007. Second Joint*, pages 304–309. IEEE, 2007.