Hindawi Publishing Corporation
Advances in Human-Computer Interaction
Volume 2012, Article ID 162868, 7 pages
doi:10.1155/2012/162868

Research Article

Improved Haptic Linear Lines for Better Movement Accuracy in Upper Limb Rehabilitation

Joan De Boeck,[1,2] Lode Vanacken,[1] Sofie Notelaers,[1] and Karin Coninx[1]

[1] *Expertise Centre for Digital Media, transnational University Limburg, Hasselt University, Wetenschapspark 2, 3590 Diepenbeek, Belgium*
[2] *ICT and Inclusion, K-Point, Katholieke Hogeschool Kempen, Kleinhoefstraat 4, 2440 Geel, Belgium*

Correspondence should be addressed to Joan De Boeck, joan.deboeck@gmail.com

Received 15 August 2011; Accepted 5 January 2012

Academic Editor: Kiyoshi Kiyokawa

Force feedback has proven to be beneficial in the domain of robot-assisted rehabilitation. According to the patients' personal needs, the generated forces may either be used to assist, support, or oppose their movements. In our current research project, we focus onto the upper limb training for MS (multiple sclerosis) and CVA (cerebrovascular accident) patients, in which a basic building block to implement many rehabilitation exercises was found. This building block is a haptic linear path: a second-order continuous path, defined by a list of points in space. Earlier, different attempts have been investigated to realize haptic linear paths. In order to have a good training quality, it is important that the haptic simulation is continuous up to the second derivative while the patient is enforced to follow the path tightly, even when low or no guiding forces are provided. In this paper, we describe our best solution to these haptic linear paths, discuss the weaknesses found in practice, and propose and validate an improvement.

1. Introduction

Force feedback applications show their benefits when they are applied in a robot-assisted rehabilitation program [1, 2]. In such a setup, the training can be more finely tailored to the abilities and needs of the patient. At the same time, less assistance of the therapist may be required, allowing at the longer term to abolish the "one therapist for one patient" requirement. The ultimate goal should be to allow patients to perform their training independently without active assistance of the therapist, or even to use the force feedback enabled setup at home while being remotely monitored [3, 4]. It may sound evident that this opens possibilities for cost reduction or an increased training intensity, where the latter at its turn provides better training results [5]. Additionally, several studies suggest that using games in a therapy may improve the patient's motivation [6]. Not necessarily using force feedback, but surely by bringing the computer to the revalidation setup and exploiting the "fun"-factor, this improvement in motivation may be achieved.

Our research lab participated in a pilot study focussing on the training of the upper limbs in Multiple Sclerosis (MS) (MS is an autoimmune disease of the central nervous system, resulting in an increasing loss of force and coordination) patients [7]. As the results were promising, the project was prolonged and the focus was broadened to both MS and CVA (CVA: cerebrovascular accident or stroke is the loss of brain function(s) due to disturbance in the blood supply to the brain, caused by a blockage or a leakage of blood) patients. More information regarding our haptic-assisted rehabilitation approach can be found in [8].

A Phantom haptic device, a HapticMaster and/or a Falcon, were chosen to be used to control several game-like training exercises. The generated forces can be applied to either assist, support, or oppose the patient, according to their individual needs [9].

Assisting Forces. They "assist" the patient in performing a (new) movement. The patient can remain passive and can feel how the movement has to be made.

Supporting Forces. They help the patients, but patients have to perform the movement by themselves. The forces avoid making too large deviations from the intended motion path.

Opposing Forces. They oppose the patient's movement by generating forces opposite to the intended motion (friction, viscosity, etc.), requiring the patient to develop larger force amplitudes.

In the following sections we first introduce our previous work on this topic and illustrate the weaknesses found during further usage. Afterwards we discuss an improvement on our work as well as experimental data to validate and motivate it.

2. Previous Work

During the development of rehabilitation exercises, we noticed that only a few basic haptic effects were necessary. Among them, we identify the ones available in most haptic APIs such as feedback when touching objects, spring forces, or magnetic force fields. However, one important building block is what we call a "linear path" between two or more points in space. One of the important features of such a haptic linear path is that an adjustable spring force *supports* the patient by attracting the cursor to the center of the line. Supplementary, additional forces such as an *assisting* forward force, or an *opposing* friction or viscosity may be useful as well. The current existing implementations, in the most common haptic APIs. (We consider OpenHaptics [10], CHAI3D [11] and H3D [12]), unfortunately are not fully suitable. In what follows, we first shortly indicate the design requirements.

(1) *Continuous Path*. It is required that the haptic simulation is continuous in the first and the second derivative in each point. This is necessary for a smooth haptic rendering with no bumps or oscillations, furthermore allowing the smooth superimposing of additional forces (see item 5).

(2) *Easy to Design*. For the designer of a scene, the interface must be as easy as possible, avoiding complex shapes such as splines, to be entered manually. A simple enumeration of a set of points in space may be suitable, although a naive implementation would conflict with our first requirement.

(3) *Approximate the Given Path*. The interpolated continuous path must approximate the original linear path *as close as possible*.

(4) *Support for Several Devices*. In our rehabilitation project we consider three different haptic devices: the HapticMaster, the Phantom, and the Falcon. Obviously, the same software should apply to all three devices.

(5) *Apply Additional Forces*. For our rehabilitation program it is necessary not only to generate a spring force to the center of the curve. Additional forces, either supporting or opposing (such as a forward force, friction, or viscosity), have to be superimposed, as well.

In [13] we proposed two approaches that meet all of our requirements: one using rounded corners, another using cardinal splines, as is illustrated in Figure 1. Analysis showed that both types had their own benefits and disadvantages. Using both implementations in practice, we found that the cardinal spline solution provided the best results. Therefore, in the scope of this paper, we will limit our discussion to the cardinal spline implementation.

In the next section, we first explain the math behind the original implementation and explain the problem with this implementation. Thereafter, in Section 4, we propose our additional "pull-back algorithm" as an improvement. Finally, in Section 5, we show the results of our solution in a short benchmark.

3. Haptic Lines Implementation

As shown in Listing 1, the designer of a new scene or rehabilitation exercise defines a new path by defining a sequence of points in (3D) space. Using Cardinal splines interpolation [10], the haptic rendering is calculated. Cardinal splines are a subset of Hermite Splines where the tangent control points are defined as a function of the other control points (providing an easier interface to the "designer").

The Cardinal Splines solution guarantees that the continuous curve runs through the control points. Depending on the "tension factor," however, the actual curve can slightly deviate from the line between two successive points (see Figure 1(b)).

Given the points s_1 and s_2 and the tangential lines T_1 and T_2, the tangential line T_n is given by $\alpha \cdot (s_{n-1} - s_{n+1})$. This means that the tangential is parallel with the line between the previous and the next control point. Alpha (α) is the tension factor, typically between 0 and 1, defining how "tight" the curve is in the control points.

The spline curve is then defined by four parametric functions:

$$\begin{aligned} h_1(t) &= 2t^3 - 3t^2 + 1, \\ h_2(t) &= -2t^3 + 3t^2, \\ h_3(t) &= t^3 - 2t^2 + t, \\ h_4(t) &= t^3 - t^2. \end{aligned} \tag{1}$$

An arbitrary point P on the curve is calculated by

$$\vec{P}(x, y, z) = h_1(t) \cdot s1 + h_2(t) \cdot s2 + h_3(t) \cdot s1 + h_4(t) \cdot s2. \tag{2}$$

For a smooth haptic rendering, we need to find the perpendicular projection h' from the position of the haptic device h. The calculation of the minimal distance of a point to the spline can be achieved by solving the equation $\partial P(t)/\partial t = 0$, but this is not trivial [14]. We decided to take a more pragmatic approach and exploit the strong coherence between successive haptic rendering steps by searching for a new local minimum in the neighborhood of the previous projection point.

The pragmatic approach can be applied for all rendering steps, except for the very first one (at startup). In that situation, the entire closest line segment has to be sought for a

```
(1) <Toggle Group DEF="Linear Pad" haptics On="false" graphics On="false">
(2)   <Shape>
(3)   <Appearance>
(4)     <Material emissive Color="0 0 0"/>
(5)     <Line Properties linewidth Scale Factor="5"/>
(6)   </Appearance>
(7)   <Linear Path vertex Count="11" DEF="myPath">
(8)     <Coordinate point="−0.1466667 0.02666667 0,
(9)        −0.1053333 0.078 0,
(10)      −0.005333333 0.07933334 0.0,
(11)       0.018 0.04666667 0.0,
(12)      −0.01066667 0.01866667 0,
(13)      −0.06933333 0.04333333 0,
(14)      −0.07333333 0.1213333 0,
(15)       0.05466667 0.1266667 0,
(16)       0.1166667 0.05333333 0,
(17)       0.1146667 −0.012 0,
(18)       0.072 −0.036 0"/>
(19)  </LineraPath>
(20)  </Shape>
(21)  <Linear Path Forces DEF="myForces" interpolation="splines"
(22)      spring Constant="40"
(23)      aid Constant="1.3"
(24)      viscosity Constant="5"
(25)      static Friction="0.8"
(26)      dynamic Friction="0.5">
(27)     <Linear Path USE="myPath"/>
(28)  </Linear Path Forces>
(29) </Toggle Group>
```

Listing 1: Listing example of a linear path with some haptic effects.

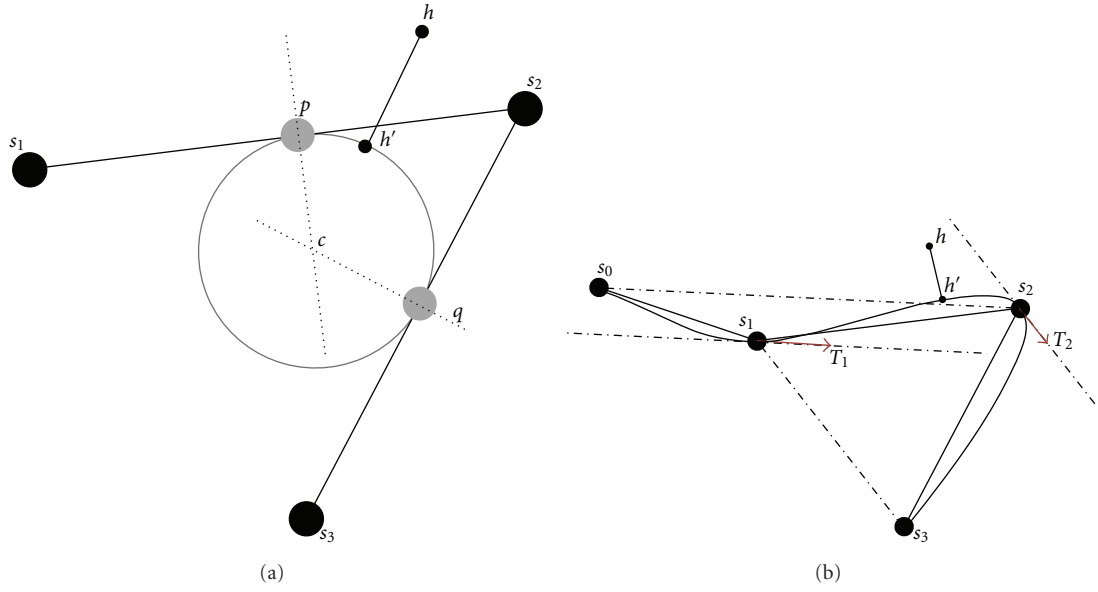

Figure 1: (a) First solution using rounded polygons. (b) Second alternative using cardinal splines.

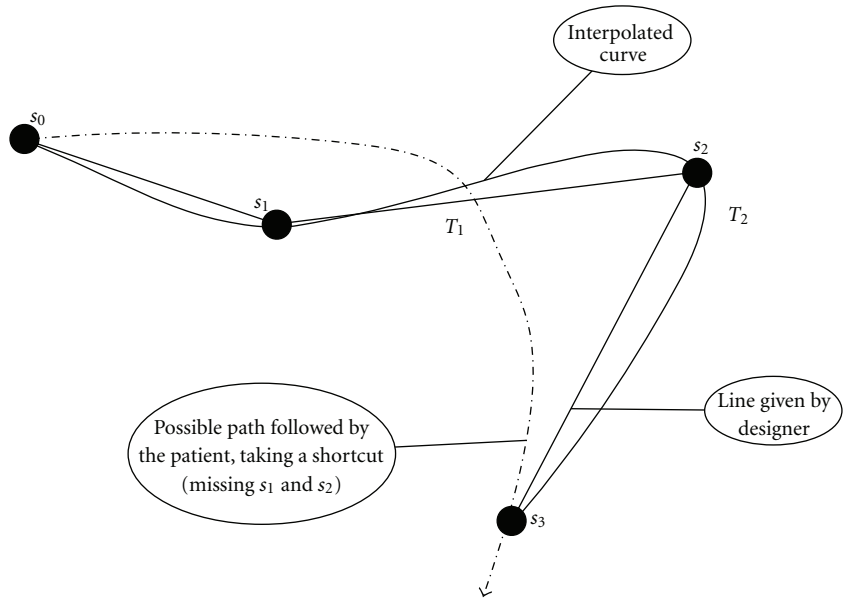

Figure 2: Illustration of a possible "shortcut a user can make."

minimum distance. In our implementation, for performance optimization, the spline curve is precalculated in a spline table.

The spline equation is a parametric equation (in t), with t varying between 0 and 1, respectively, corresponding with the start and the end point of a segment. It may be clear that the step size of the samples of t, must be linked to the actual length of a segment; otherwise small segments would be oversampled while longer segments would have a coarser resolution. A reasonable high resolution is important to ensure smooth haptics; otherwise the haptic simulation may suffer from stutter effects. In our implementation, the number of steps is proportional to the Euclidian distance between the control points, or is written as:

$$n_{\text{steps}} = k \cdot |\, p_n - p_{n-1} \,|. \tag{3}$$

For example, for the phantom k is determined to be 7000 based on the resolution of the phantom which is 0.03 mm. For the other devices k would be smaller or close to k; therefore we define k as a constant at 7000 for all our haptic devices.

4. Haptic Lines Problem and Improvement

The implementation described in the previous section has as a downside that while the spring forces are low, it may be easy to find "a shortcut" between the beginning and the end of a line. Figure 2 illustrates this shortcoming. It may be clear that in a rehabilitation context; this effect is unwanted, as we want the patient to accurately follow the predefined path. This is particularly true in low-force circumstances where it is (physically) easy to deviate from the path. In this section we will describe an improvement which forces the patient to better keep on the track.

4.1. Pullback Function. The central idea of our solution is to define a continuous function that virtually "pulls back" the perpendicular projection of the cursor on the spline curve based upon the distance of the actual cursor from the curve. The larger the distance is between the cursor and its projection, the more the projection point is pulled back. The new (pulled back) projection point is then used for calculating the feedback forces. Pulling back the projection point increases the attraction forces and introduces a slight resistance force to advance on the path, but even when low or no attraction forces are present, the pullback function will ultimately halt a wrong progression of the user in a continuous way.

4.2. Implementation. Although the general idea is pretty simple, the implementation is less evident. In this section, we will clarify how the function is implemented and how continuity is ensured.

The pullback function is defined as follows (see also Figure 3):

$$G - \left(\left(1 - \cos\left(\frac{\text{dist}}{d} \times \pi\right)\right) \times \frac{m}{2.0}\right), \tag{4}$$

where G is the global spline parameter (see further) on the spline curve (taking all segments into account) and dist is the perpendicular distance of the actual cursor to the curve. d and m are two constants that define the behaviour of the pullback function as will be explained later in this section.

We work with the global distance parameter G, rather than the parametric spline values per segment, as the the pullback function must behave consistently independent of the length of a particular segment. In order to calculate G, in a first step, the current spline parameter (in the current segment) must be converted into a global distance value taking all the past segment lengths into account. The total distance

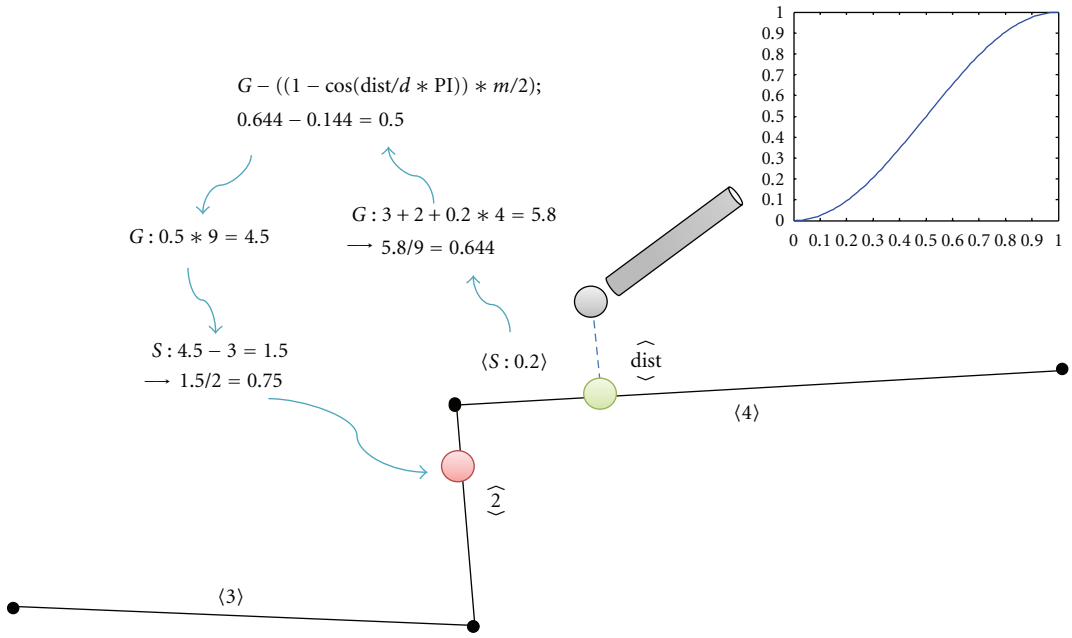

Figure 3: The pullback algorithm.

from the beginning of the curve, divided by the total spline length gives us the global distance parameter (G).

The second term in the formula is a sinusoidal function in dist that behaves as shown in the upper right corner in Figure 3. d is a constant that defines the distance at which the pullback function is maximal. When $d >$ dist, the pullback function is kept at its maximum. Hence, when dist approaches from zero to d, the cosine function ensures a smooth and continuous increase.

The parameter m defines the maximum value the projection point is "pulled back". When dist approaches to d, the cosine becomes 0 and the maximum pull back value (divided by 2.0) is applied. Hence, our global distance parameter is lowered by the pullback value.

After applying the equation we reconvert G back into a parametric spline value (taking the lengths of the different segments into account) that gives us the new position onto the haptic line: the *pullback point*.

The explanation above describes the basic algorithm. However, a couple of smaller refinements were necessary in order to make the algorithm work in practice and insure continuity in all cases. Amongst them, some logic takes into account that it is possible for a user to make a shortcut that goes beyond the border of one segment, or even to skip one or more segments.

5. Benchmarking

5.1. Experimental Design. In order to verify our improvement, an evaluation was necessary. However, conducting a standard user study would not provide us with satisfying results. This is because the enhancement described in this paper is to prevent patients to take a shortcut instead of following the defined path; this situation grows only after a patient is fully familiar with the application and has learned to exploit all back doors. Hence, evaluation using naive users would not give us any useful feedback on the quality of our solution.

Therefore, we opted to perform a benchmark test. In this benchmark we deliberately try to search for the shortest path between the start and the end of the curve, not necessarily following the path. This was done using different paths and different parameters, as defined:

(i) four different values for the attractive force to the center of the line (0 N, 50 N, 100 N, 150 N), (these are the values given to the API. The values were empirically chosen in order to have "no," "weak," "medium," and "strong" attraction forces),

(ii) four different pullback strengths, defined by the parameter m (0, 0.033, 0.066, 0.1) (these values are expressed as a percentage of the total curve length),

(iii) two different haptic lines: a highly curved curve and a less curved, as shown in Figure 4.

This results in 16 different benchmarking conditions for the two curves, or 32 conditions in total.

Two of the authors performed the benchmarking, as they are supposed to know the insights of the solution and are better able to find the shortest possible path. In order to ensure that the authors would really strive for the best minimal values, a little competition between the benchmarkers was set up, electing the winner as the one who found the shortest path in the majority of the conditions.

For each of the 32 conditions, each benchmarker was obliged to try for 10 times to find the shortest path, while the result (the ratio between the covered path and the actual length of the path) was shown after each trial. The latter was also logged to file for later analysis.

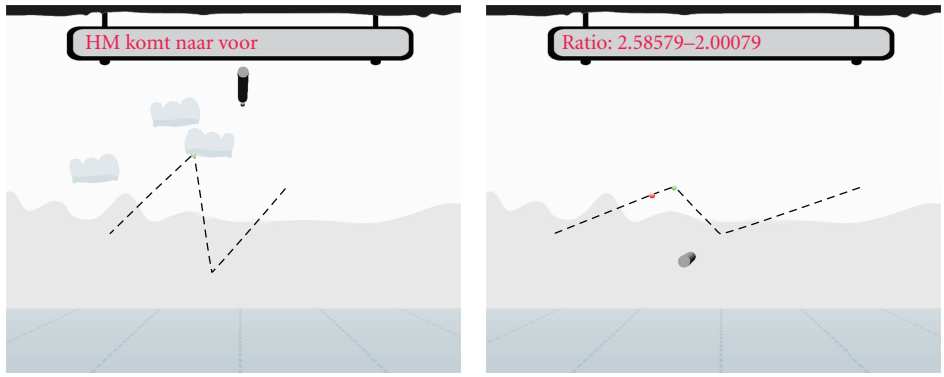

Figure 4: The two curves used in the benchmarking test. Curve 1 (a) with a high curvature, curve 2 (b) with much less curvature. Note that the visual representation only shows the straight lines, while the haptic simulation uses the spline interpolation.

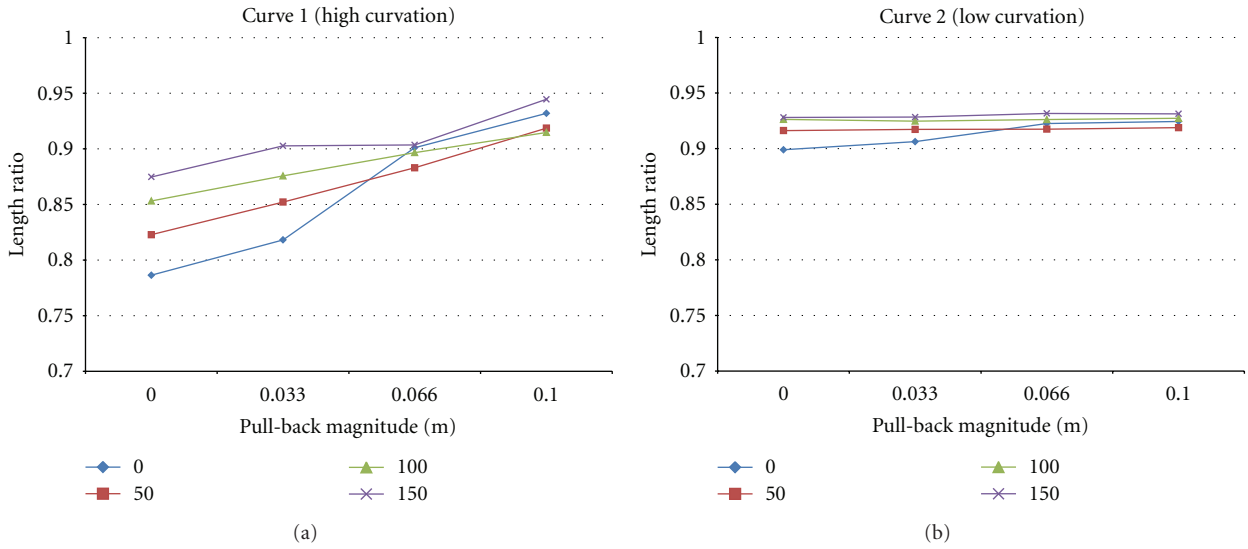

Figure 5: Ratio (shortest path/actual path length) for curve 1 (high curvation) and curve 2 (low curvation) for the different attraction forces (colours) and different pull-back values (x-axis).

5.2. Results. The results of the benchmark are given in Figure 5. The graph depicts the lowest ratio (measured distance versus actual distance), measured for a given condition, respectively, for curves 1 (a) and 2 (b). The x-axis contains the different m values for the pullback algorithm; the different graphs show the different attraction forces (no force, low, medium, and high force).

In what follows, we clarify how these graphs can be interpreted.

In Figure 5(a), given the $m = 0$ condition (first column), we get a ratio of 0,78 in the no-force condition (lower curvature). This means that one of the benchmarkers could complete the trial by making a movement that was 78% shorter than the actual intended path. In this condition there was no attraction force and no pull back, so we can assume that this is the shortest path that the spline interpolation allows. In Figure 5(b), we see for the same condition a ratio of 0,90, which can be explained by the fact that the intended path is less curved, and hence the possible deviation is smaller.

When looking at the different force conditions (still given $m = 0$), we observe that the more the attraction force increases, the more tightly the original curve must be followed. This may in no means be a surprise, as this is the actual goal of the haptic lines: supporting a patient in practising a given path or movement and keeping them tight to this path. It may be evident that in curve 2 the difference is less pronounced as the path is more straight.

Having a better understanding of how to interpret the graphs, let us now discuss the influence of the parameter m, indicating "how far" the projection on the curve is "pulled back" when the perpendicular distance to the curve is increasing. In Figure 5(a), we see that increasing m forces the user to better follow the original curve in all force conditions. In the situation where $m = 0.1$, the ratio is even almost

independent from the attraction force, with ratios between 0.91 and 0.94. For curve 2, the result is obviously less pronounced, but here we also notice that the "no force, no pull-back" condition allows to slightly "bypass" the intended curve, while adding our pull-back algorithm inhibits this behaviour, independent of the attraction forces.

This is exactly the initial intention of our pull-back mechanism: finding a smooth and continuous manner of forcing a user to follow a given path, when attraction forces are low. Moreover, we felt that our solution did not hinder the natural interaction with the system, which was the case in our former rounded corners solution described in [13]. The latter, however, remains to be confirmed by a practical usage in our project.

6. Conclusion

In our current research project, focussing on the upper-limb rehabilitation of MS and CVA patients using force feedback supported exercises, we found that several basic force feedback behaviors were necessary, among which a linear haptic path. Unfortunately current implementations in available APIs do not suite our particular need. In this paper, we described one of our best performing alternatives to implement these curves, based upon Cardinal Splines. In practice, however, we observed that patients getting experienced in the exercises, had the possibility to shorten their path as the attraction force was lowered. Therefore, we proposed a "pull back" algorithm, smoothly pulling back the projection on the curve based upon the distance of the cursor and the curve. In a benchmarking test, we could show that adding a higher pull-back factor makes the user to better follow the intended path, independent of the available attraction forces.

During informal tests, we found that the pull-back function felt natural and did not hinder the interaction. This, however, requires verification in a formal user study, by letting naive users or patients work with our solution during real rehabilitation exercises.

Acknowledgment

This research was partly funded through the INTERREG program (Project 4-BMG-II-1-84 and IVA-VLANED-1.14, Euregio Benelux).

References

[1] J. E. Deutsch, J. Latonio, G. C. Burdea, and R. Boian, "Post-stroke rehabilitation with the rutgers ankle system: a case study," *Presence*, vol. 10, no. 4, pp. 416–430, 2001.

[2] M. K. Holden, "Virtual environments for motor rehabilitation: review," *Cyberpsychology and Behavior*, vol. 8, no. 3, pp. 187–211, 2005.

[3] M. Rosen, "Introduction to special topic issue on technology in neurorehabilitation," *NeuroRehabilitation*, vol. 12, no. 1, pp. 1–2, 1999.

[4] G. Lathan, "Dimensions of diversity in design of telerehabilitation systems for universal usability," in *Proceedings of the Conference on Universal Usability (CUU '00)*, pp. 61–62, ACM, New York, NY, USA, November 2000.

[5] G. Kwakkel, R. van Peppen, R. Wagenaar et al., "Effects of augmented exercise therapy time after stroke: a meta-analysis," *Stroke*, vol. 35, no. 11, pp. 2529–2539, 2004.

[6] S. J. Housman, V. Le, T. Rahman, R. J. Sanchez, and D. J. Reinkensmeyer, "Arm-training with T-WREX after chronic stroke: preliminary results of a randomized controlled trial," in *Proceedings of the IEEE 10th International Conference on Rehabilitation Robotics (ICORR '07)*, pp. 262–268, Noordwijk, The Netherlands, June 2007.

[7] J. de Boeck, G. Alders, D. Gijbels et al., "The learning effect of force feedback enabled robotic rehabilitation of the upper limbs in persons with MS—a pilot study," in *Proceedings of the 5th Enactive International Conference (ENACTIVE '08)*, pp. 117–122, Pisa, Italy, November 2008.

[8] T. de Weyer, S. Notelaers, K. Coninx et al., "Watering the flowers: virtual haptic environments for training of forearm rotation in persons with central nervous deficits," in *Proceedings of the 4th International Conference on Pervasive Technologies Related to Assistive Environments (PETRA '11)*, ACM, Crete, Greece, May 2011.

[9] V. Popescu, G. Burdea, M. Bouzit, M. Girone, and V. Hentz, "Pc-based telerehabilitation system with force feedback," *IEEE Trans Inf Technol Biomed*, vol. 4, no. 1, pp. 45–51, 2000.

[10] Sensable Technologies, OpenHaptics Toolkit, 2011, http://www.sensable.com/.

[11] Chai 3d api, June 2011, http://www.chai3d.org/.

[12] SenseGraphics, H3D API, 2011, http://www.h3dapi.org/.

[13] J. de Boeck, S. Notelaers, C. Raymaekers, and K. Coninx, "Haptic linear paths for arm rehabilitation in MS patients," in *Proceedings of the IEEE International Workshop on Haptic Audio Visual Environments and Games (HAVE '09)*, pp. 42–47, Lecco, Italy, September 2009.

[14] Distance to a bezier curve, June 2011, http://www.tinaja.com/glib/bezdist.pdf.