Haptic Linear Paths for Arm Rehabilitation in MS Patients

Peer-reviewed author version

DE BOECK, Joan; NOTELAERS, Sofie; RAYMAEKERS, Chris & CONINX, Karin
(2009) Haptic Linear Paths for Arm Rehabilitation in MS Patients. In: Proceedings of IEEE International Workshop on Haptic Audio visual Environments and Games, 2009. HAVE 2009. p. 42-47..

Handle: http://hdl.handle.net/1942/10326

Haptic Linear Paths for Arm Rehabilitation in MS Patients

Joan De Boeck Sofie Notelaers Chris Raymaekers Karin Coninx
Hasselt University - tUL - IBBT
Expertise Centre for Digital Media
Wetenschapspark 2
B-3590 Diepenbeek (Belgium)
Email: {joan.deboeck, sofie.notelaers, chris.raymaekers, karin.coninx}@uhasselt.be

***Abstract*—Force feedback in the context of a rehabilitation program may have its benefits. The generated forces can be used to assist, support or oppose the patients according to their personal needs and abilities. Using the Phantom haptic device, we conducted a pilot study focussing on the rehabilitation of the upper limbs in MS patients. Apart from the promising clinical results, we found that only a few haptic effects are commonly necessary. Among them is a 'linear path', a haptic effect that generates the necessary forces to follow a path defined by two or more points in space. In this paper we motivate why the current implementations in existing haptic APIs (such as H3D) are not completely suitable. We propose two possible alternative implementations based on our requirements: one using rounded polygons, another using cardinal splines. It will turn out that both solutions are equivalent, but depending on the application one of both will be more suitable.**

I. Introduction and related work

Haptic interfaces have been gaining importance over the last decades in a variety of domains [1]. One of the promising domains undoubtedly is rehabilitation, where force feedback devices are applied in the training sessions of locomotory impaired people [2][3]. The major advantages are that the training can be more finely tailored to the abilities and needs of the patient, while less assistance of the therapist is required. The ultimate goal is to allow a patient to perform the force feedback enabled setup at home while being remotely monitored by the therapist [4][5]. This opens possibilities for an increased training intensity, which at its turn provides better results [6]. Additionally, a computer driven therapy also has the ability to integrate gaming concepts within the training program, which may improve the patient's motivation [7].

Recently, our research lab participated in a pilot study focussing on the training of the upper limbs in Multiple Sclerosis (MS) patients [8]. MS is a autoimmune disease of the central nervous system, resulting in an increasing loss of force and coordination. A Phantom haptic device was used to control some game-like training exercises, while the generated force could be applied to assist, support or oppose the patient. Based on the promising results of this pilot study, a research project in a multidisciplinary consortium is currently being executed.

In this pilot study, and during the preparation of additional exercises for the continuation of this project, 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.

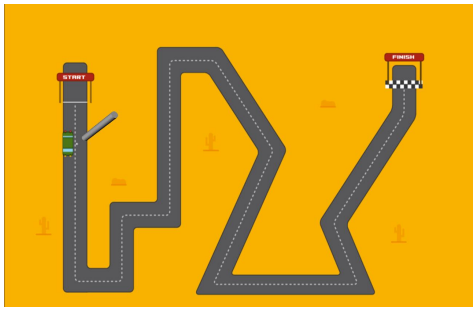

Fig. 1. Exercise where the patient has to steer a car from the start to the finish.

These and other force effects, applied in a rehabilitation context can be classified into three groups [9]:

- **Assisting forces:** These forces '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:** With these forces, patients have to perform the movement by themselves, but the force avoids making too large deviations from the intended motion path.
- **Opposing forces:** These forces oppose the patient by generating forces opposite to the intended motion. As a result, larger force amplitudes are necessary in order to complete the movement.

We also found in our pilot study that a 'linear path' between two or more points in space is a frequently used force effect. A practical example of this 'linear path' in our study is shown in figure 1. Here the patient has to 'steer' the car on the road from start to finish, while being 'assisted' by several forces classified as mentioned above: an adjustable spring force *supports* the patient by attracting the car to the center of the road. Additional forces, such as an *assisting* forward force, or an *opposing* friction or viscosity may be useful as well.

It is a key requirement in our project that for this 'linear

path', the entire scope of forces (assisting, supporting and opposing) must be covered. However, as will be motivated in the next section, it appears that the 'linear paths' built in into existing APIs do not fully cover all of our requirements. Therefore, we report on two possible alternative implementations that cover the full range of necessary forces. The first solution uses *rounded polygons*, the other uses *cardinal splines*. Similar mathematical approaches and haptic issues may be found in Sjostrom et al. [10] where mathematical functions are made 'touchable' in order to teach blind students. Furthermore, Raymaekers et al. [11] implemented a haptic paradigm for curve drawing using Bezier splines. Here, assisting forces are generated to facilitate the modification of an existing curve.

In the next section, we first enumerate the design requirements, and when appropriate motivate why the current implementations are not fully suitable. Thereafter in section III, we briefly describe both proposed solutions. Next, in section IV, we discuss the practical integration within H3D. Next, in section V we describe some considerations with both implementations and discuss their strengths and weaknesses. We end this paper by proposing which solution is the better in what situation.

II. Design Requirements

As mentioned in the introduction it may have become clear that the definition of a haptic linear path between a set of predefined points in space is one of the common building blocks in our rehabilitation project. However, the current existing implementations, are not completely suitable. In what follows, we first enumerate the different design requirements, and when appropriate argue why the standard API implementations are not suitable. We consider OpenHaptics [12], CHAI3D [13] and H3D [14]. However, as for our rehabilitation project, we opted for H3D, in the remainder of this paper we mainly focus on the H3D implementation.

1) **Continuous Path**: It is required that the interpolated path is continuous in the first derivative in each point. This is necessary for a smooth haptic rendering with no bumps or oscillations. Furthermore, the normal and tangential lines (first derivative) must be continuous as well when superimposing additional forces (see item 5). If this requirement is not met, patients tend to get stuck in sharp corners. The CHAI3D implementation only supports individual line segments, and hence is not continuous in the first derivative.
2) **Easy to design**: For the developer of a scene, the interface must be as easy as possible, preferably by just enumerating the subsequent points in space. This is how H3D's magnetic lines are implemented. Taking the continuity requirement into account, splines appear to be the obvious solution. However, depending on the chosen spline, coping with tangent lines and control points may not always be intuitive for novice designers.
3) **Approximate the given path**: The interpolated continuous path must approximate the original linear path *as close as possible*. Dependent on the applied interpolation formula the result may strongly differ from the original path, as can be seen from figure 2.

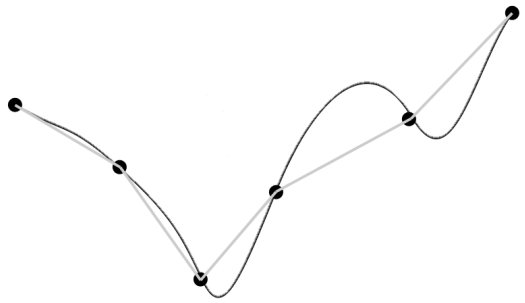

Fig. 2. Cubic spline curve through given set of points strongly differs from the original 'linear path' (gray line).

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 devices. CHAI3D currently does not support the HapticMaster and although H3D supports all of our devices, the magnetic lines implementation only works on the Phantom device as it relies on the openHaptics API. Obviously, the OpenHaptics solution at its own only supports the Phantom device, as well.
5) **Apply additional forces**: As will be elaborated upon in more detail in section IV and V-B, 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 the current H3D implementation other force fields can be combined with the 'magnetic lines', but very often this results in a jerky or instable rendering.

III. Proposed solutions

Taking the above considerations into account, we selected two possible solutions:

1) Rounded polygons, where the corners of a polygon are rounded using arc segments [15].
2) Cardinal splines [16], a subset of Hermite Splines where the tangent control points are defined as a function of the other control points.

In what follows, we briefly describe the mathematical principles of both solution. Within the scope of this paper, we will particularly focus on the issues of a smooth haptic rendering.

A. Rounded Polygons

The first selected solution adopts the principles of rounded polygons [15]. This solution guarantees maximal coincidence of the interpolated curve with the linear connection between the control points, but as a result, the curve will never actually go through these control points (as the corners are rounded).

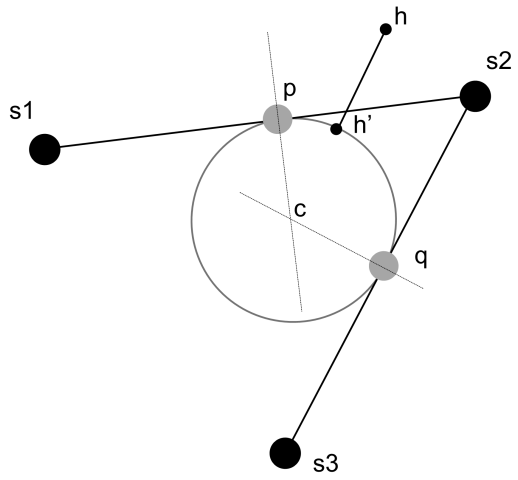

Fig. 3. Rounded corners between two line segments.

Figure 3 shows how the arcs are defined. Given two subsequent segments of a 'linear path', defined by s_1,s_2 and s_3, at a user-defined distance from s_2, two points p and q are chosen. The intersection between the two perpendicular lines through p and q define the center c of the arc. Now, the arc $\overset{\frown}{pcq}$ defines a new segment that is placed in between segments $\overrightarrow{s_1p}$ and $\overrightarrow{qs_3}$.

For a smooth haptic rendering the perpendicular projection h' of the position of the haptic pointer h is calculated in each haptic loop iteration. This is mostly a trivial operation for both the line and the arc segments, although some issues with the arc segment exist, as will be explained in section V. For the calculation of the required forces (spring, friction, viscosity, ...) we need to calculate the tangent and the perpendicular line in h'. From figure 3 it may be clear that at the segment transitions points (p and q) the first derivative is continuous, which guarantees a smooth haptic rendering for all additional forces.

B. Cardinal Splines

Our second solution guarantees that the continuous curve runs through the control points. As a result, however, the path in between may slightly deviate, depending on the 'tension factor'. Cardinal splines are a subset of Hermite Splines, where the tangent control points are calculated depending on the other control points.

Given the points s_1 and s_2 and the tangential lines T_1 and T_2 (see figure 4), 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 defined by four parametric functions:

$$h_1(t) = 2t^3 - 3t^2 + 1$$
$$h_2(t) = -2t^3 + 3t^2$$

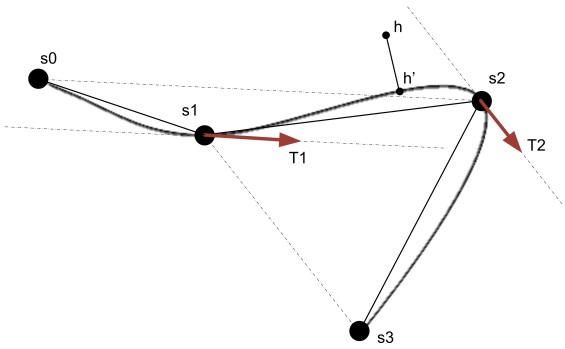

Fig. 4. Cardinal Spline interpolation.

$$h_3(t) = t^3 - 2t^2 + t$$
$$h_4(t) = t^3 - t^2$$

An arbitrary point P on the curve is calculated by:

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

For a smooth haptic rendering, again 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 is not trivial and can be achieved by solving the equation $\partial P(t)/\partial t = 0$ [17].

In our implementation 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. Obviously for the very first rendering step (at startup), the entire closest line segment has to be sought for a minimum distance. For performance optimization, the spline curve is precalculated in a spline table.

Another implementation consideration is the resolution of the spline interpolation. 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. With a fixed step-size, longer segments hence have a coarser resolution, while small segments may be over-sampled. Therefore, the number of steps is proportional to the euclidian distance between the control points ($n_{steps} = k \cdot |p_n - p_{n-1}|$, where k is determined at 7000 based upon the Phantom's resolution of 0.03mm). For the other devices, this value appears to produce good results, as well.

IV. Practical Implementation in H3D

The above described principles have been implemented in an H3D node that can be used within an H3D (X3D) scene file. Listing 1 shows an example of the syntax.

The implementation requires that the developer first defines the control points of the desired path using the 'LinearPath' tag. Next, in line 21, the haptic representation is defined using the 'LinearPathForces' tag. This tag includes the control points in line 27. The applied interpolation algorithm (polygons or splines) is chosen by the '*interpolation*' attribute, while

the force parameters are defined each by their appropriate attribute(s) (as shown in line 22 to 26)

In the enumeration below we discuss the possible parameters and, when appropriate, refer to the example. The mathematical background for the generated forces is discussed later in section V-B.

- **Assisting forces:** Here we define a *constant forward* force with a given magnitude, tangential to the interpolated curve. Similarly a force can be defined that strives for constant (forward) velocity. Line 23 in listing 1 shows the definition of a constant forward force using the attribute *aidConstant*. It's also possible to provide negative force values; obviously the force will then become opposing.
- **Supporting forces:** This is the main spring-force, perpendicular to the interpolated curve. The developer can adjust the stiffness by changing the attribute *springConstant* (line 20).
- **Opposing forces:** These forces include static and dynamic friction, as well as viscosity (both shown in listing 1). Friction is defined by 'staticFriction' and 'dynamicFriction' (line 25 and 26). Viscosity can be added using the 'viscosityConstant' attribute (line 24).

```
1  <ToggleGroup DEF="LinearPad" hapticsOn="false" graphicsOn="false">
2      <Shape>
3      <Appearance>
4          <Material emissiveColor="0 0 0" />
5          <LineProperties linewidthScaleFactor="5" />
6      </Appearance>
7      <LinearPath vertexCount="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     </LinearPath>
20     </Shape>
21     <LinearPathForces DEF="myForces" interpolation="splines"
22       springConstant="40"
23       aidConstant="1.3"
24       viscosityConstant="5"
25       staticFriction="0.8"
26       dynamicFriction="0.5">
27       <LinearPath USE="myPath" />
28   </LinearPathForces>
29 </ToggleGroup>
```

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

V. Considerations

In this section we report on some considerations that rose when implementing and testing the aforementioned solutions.

A. Artifacts when approaching a corner from the inside bend

When approaching a corner from the inside bend as shown in figure 5 (dashed line), both the rounded polygon as the spline solution suffer from some artifacts.

With the rounded polygons implementation a problem arises when a path $H[..., h_1, h_2, h_3, h_4, ...]$ is followed (figure 5). As soon as h traverses the line cp, a transition to the next segment has to be made. In our case the arc $\overbrace{pcq}$ should be expected as the successor. However, when in h_3, the closest point on the curve already lies on the next segment $[qs_3]$. It may be clear that a jump from h'_2 (lying on segment $[s_1p]$) to a point on segment $[qs_3]$ in two successive iteration would violate the continuity requirement. Therefore, an extra condition has been added to solve this problem: upon traversing the line cp, when the haptic pointer is 'below' the center of the circle (e.g. point h_3), it is first projected onto the arc $\overbrace{\mathbf{r}cq}$ (h'_3). Next the arc $\overbrace{rcq}$ is linearly mapped onto the arc $\overbrace{pcq}$, bringing h'_3 to h''_3. This way, the continuity requirement is maintained. As soon as $h''_n = h'_n$, which is only true if $h''_n = h'_n = q$, this special condition is left, and the path can continue in a continuous way with segment $[qs_3]$.

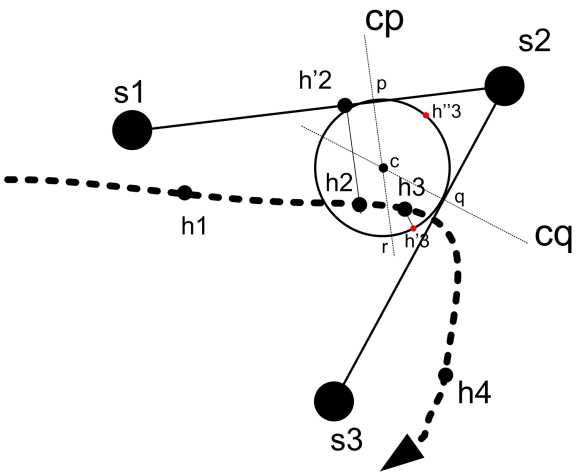

Fig. 5. 'Getting Stuck' when approaching the control point s_2 from the inside bend (polygon solution).

The result of this solution for the user is that apparently the cursor 'gets stuck' when the inside bend is taken too sharp. This blockage is only released when the cursor is brought *close enough* to the control point (s_2), in particular when traversing the line cq. This solution requires the user to take the curve close enough to the control point. In practice this is not a problem when the spring-force is stiff enough (as it attracts the pointer close enough to the curve). In a very loose situation however, the described artefact may be confusing for the user.

A similar (but inverse) artefact shows up with the spline implementation. From figure 6, it can be easily understood that for the depicted path $H[..., h_1, h_2, h_3, h_4, ...]$ the projections of $[h_2, h_3]$, covers nearly the entire curve $[h'_2, h'_3]$. This may give the feeling of slipping off the edge of an object, fast-forwarding the curve at that point. Here again, a stiffer attraction to the curve's center reduces this effect. Although the haptic rendering remains smooth and stable, the artefact may be undesired in some situations, as well.

Whether or not it is too easy (spline) or difficult (rounded polygon) to take the inside bend, depending on the application both artifacts may be desired or undesired, and may be key arguments to choose for one or the other implementation.

B. Additional Forces

As already stated in section IV, additional forces can be easily added from within the H3D-file. In what follows, we shortly describe how the respective forces are designed.

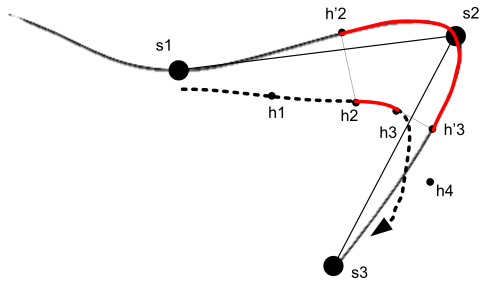

Fig. 6. 'FastForward' when approaching the control point s_2 from the inside bend (spline solution).

For the **assisting forces**, we defined a *constant forward* force, with a given magnitude, tangential to the interpolated curve. Similarly we have added an assisting force that tries to keep a constant (forward) velocity: when the velocity is below a given value, the forward force is slowly increased in a continuous way. Alternatively when the given velocity is exceeded, a negative force will slow down the movement.

The **supporting force** is a spring-force, attracting the pointer perpendicular to the center of the interpolated curve, with a magnitude proportional to the distance from the center. The force is given by $F_s = k.s$. With k the spring stiffness and s the perpendicular distance from the curve.

The **opposing forces** include friction and viscosity. Both static and dynamic friction are implemented using the 'classic' Coulomb friction model, as described in [18]. Static friction is implemented as a spring force, until the force exceeds the 'static friction force' given by $F_{c_s} = k_s.N$. Dynamic friction is calculated by the formula $F_{c_d} = k_d.N$. With $k_s > k_d$, respectively the static and dynamic friction coefficient. N is the normal force caused by the 'weight of the object'. In our implementation $N = F_s$, the supporting spring force, but alternatively, a constant N can be entered as well.

Viscosity is implemented as $F_v = -k_v.v^2$: the viscosity force is proportional to the square of the cursor's velocity, opposing the direction of the movement.

C. Compatibility with the required devices

As already stated in the requirements-section three devices are considered to be applied in the rehabilitation project: the HapticMaster as a high-end, high-force device; the Phantom, as a high-end device, requiring a finer coordination and the Falcon, as a low-cost solution. All three devices are required to smoothly render the designed curve, according to their characteristics.

Although the proposed solution is independent of the chosen renderer, We have chosen to verify our solution using H3D's Ruspini Renderer [19]. As this renderer is device independent, it may not surprise that the solution automatically runs on all compatible devices. An informal test on our three devices did not show any surprising results: the Phantom showed a very smooth and stable rendering of the interpolated curve. With the HapticMaster, we had a more 'bouncy' simulation, that could very easily started to oscillate (low frequency oscillations). This result, however, was expected as the HapticMaster is an admittance-controlled haptic device [20] where H3D 'simulates' a frictionless free mass. In particular the property of being frictionless easily leads to these low-frequency oscillations. Simply adding a viscosity-effect of $k_v = 15$ on our path, entirely solved the problem. The Falcon device, showed a nice rendering, but as could be expected the lower resolution and higher friction were noticeable.

As these findings are the result of the properties of the haptic devices, rather than from our solution, further details fall beyond the scope of this paper.

VI. Comparison

As both solutions meet our requirements, the question rises which of both proposed solution is the most preferable: both solutions provide a smooth and stable haptic rendering, are continuous in the first derivative, include the required additional forces, are compatible with all required devices, are easy to design, and approach the original linear path. The rounded polygons solution never actually 'includes' the control points in the curve, while the spline implementation slightly deviates from the original path. Both differences are small however, and do not argue for one or the other.

As described in section V-A, however, a substantial difference exists when approaching a control point from the inside bend. With the rounded polygons, the projection point is 'blocked' until the curve is taken decently. In contrast, the spline interpolation gives the feeling of slipping off the corner, making it easier to continue the curve.

For very small spring forces, where large deviations from the path are allowed, the 'blocking' behavior of the rounded polygons can be confusing as the users do not feel where they actually went wrong. With a moderately high spring constant however, this solution is ideal to enforce the user to follow the path as good as possible.

The spline interpolation, provides a very natural way to follow the curve, even when the actual cursor is (too) far from the curve. This may give a convenient feeling, but in a rehabilitation setup is sometimes undesired. On the other hand, with a moderately high spring force, the feeling of 'sliding off' the corner may feel 'uncomfortable'.

Choosing between both solutions is hence dependent on the application in particular, and what behavior is desired. It may be expected that practical tests with patients are necessary to determine the preference for one of both solutions. For this reason, we decided to provide both solution in our H3D implementation, making it easy for the developer to alter between both.

VII. Conclusion

From our pilot study on the rehabilitation of MS patients using force feedback supported training sessions, we learned that a haptic 'linear path' is one of the frequently necessary

effects. In this paper we argued why the existing implementations in current APIs do not completely satisfy our needs. We proposed two possible implementations, one using rounded polygons, another using cardinal splines. Both solutions have been implemented as a H3D node. They appear to be equivalent and also fully suit our initial requirements. However both solutions have a different behavior when approaching a control point from the inside bend. Which of both solutions is preferable depends on the particular application and the required behavior. In order to allow the developer to choose between both solutions, they can be easily changed within the H3D file. Finally, the implemented software can be found at http://research.edm.uhasselt.be/haptic-linear-paths.

ACKNOWLEDGMENT

Part of the research at EDM is funded by the ERDF (European Regional Development Fund) and the Flemish government. This research was funded by the INTERREG-IV programme (project Nr. 4-BMG-II-1-84, Euregio Benelux). The authors also want to thank Lode Vanacken for his appreciated assistance during the implementation phase of this work.

REFERENCES

[1] J. K. Salisbury, "Making graphics physically tangible," *Communications of the ACM*, vol. 42, no. 8, pp. 74–81, August 1999.

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

[3] M. Holden, "Virtual environments for motor rehabilitation," in *CyberPsychology and Behaviour*, June 2005, pp. 212–219.

[4] M. Rosen, "Telerehabilitation," in *NeuroRehabilitation, special topic issue on Technology in Neurorehabilitation*, vol. 12, 1999, pp. 11–26.

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

[6] G. Kwakkel, R. van Peppen, R. Wagenaar, S. Dauphinee, C. Richards, A. Ashburn, K. Miller, N. Lincoln, C. Partridge, I. Wellwood, and M. P. Langhorne, "Effects of augmented exercise therapy time after stroke: a meta-analysis," in *Stroke 35*, 2004, pp. 2529–2539.

[7] S. J. Housman, T. Rahman, V. Le, R. J. Sanchez, and D. J. Reinkensmeyer, "Arm-training with T-WREX after chronic stroke: Preliminary results of a randomized controlled trial," in *2007 IEEE International Conference on Rehabilitation Robotics*, 2007.

[8] J. De Boeck, G. Alders, D. Gijbels, T. De Weyer, C. Raymaekers, K. Coninx, and P. Feys, "The learning effect of force feedback enabled robotic rehabilitation of the upper limbs in persons with MS - a pilot study," in *Proc. of ENACTIVE08*, November 2008, pp. 117 – 122.

[9] V. Popescu, G. Burdea, M. Bouzit, M. Girone, and V. Hentz, "Pc-based telerehabilitation system with force feedback," in *IEEE Trans Inf Technol Biomed*, vol. 4(1), 2000, pp. 45–51.

[10] C. Sjostrom, H. Danielsson, C. Magnusson, and K. Rassmus-Grohn, "Phantom-based haptic line graphics for blind persons," *Visual Impairment Research: The official publication of the International Society for Low-vision Research and Rehabilitation ISL*, vol. 5, 2003.

[11] C. Raymaekers, G. Vansichem, and F. Van Reeth, "Improving sketching by utilizing haptic feedback," in *Proceedings of AAAI Spring Symposium on Sketch Understanding*, march 25-27 2002.

[12] Sensable Technologies, "OpenHaptics Toolkit," Available at http://www.sensable.com/, 2008.

[13] "Chai 3d api," Available at http://www.chai3d.org/, June 2009.

[14] SenseGraphics, "H3D API," Available at http://www.h3dapi.org/, June 2009.

[15] "Rounded polygons," Available at http://www2.tcl.tk/15313, June 2009.

[16] "Hermite splines," Available at http://www.cubic.org/docs/hermite.htm, June 2009.

[17] "Distance to a bezier curve," Available at http://www.tinaja.com/glib/bezdist.pdf, June 2009.

[18] C. Richard, "On the identification and haptic display of friction," Ph.D. dissertation, Stanford University, Stanford, 2000.

[19] D. Ruspini, *Haptics: From Basic Principles to Advanced Applications*, ser. Course Notes for SIGGRAPH '99. ACM, August 8–13 1999, no. 38, ch. Haptic Rendering.

[20] K. Wen, D. Necsulescu, and J. Sasiadek, "Haptic force control based on impedance/admittance control aided by visual feedback," *Multimedia Tools and Applications*, vol. 37, no. 1, pp. 39–52, 2008.