Dynamic Local Models for Stable Multi-Contact Haptic Interaction with Deformable Objects

Federico Barbagli, Kenneth Salisbury
Stanford University
Robotics Lab
Stanford, CA, U.S.A.
[barbagli,jks]@robotics.stanford.edu

Domenico Prattichizzo
DII
Università di Siena
Siena, Italy
prattichizzo@ing.unisi.it

Abstract

This paper describes a new technique for allowing multiple users to haptically interact with a set of deformable slowly-simulated objects in a stable manner. Stability has been approached in the past by various researchers using passivity theory in order to avoid having to model the human operator closing the haptic loop. None of these solutions however can work well without the use of high update rates and thus break down in the case of haptic interaction with slowly simulated virtual environments such as the ones featuring highly precise deformable objects. This is particularly true for the case of surgical simulation with force feedback, where precision is a key issue and where complexity can reach high levels. The techniques presented in this paper are based on the concepts of local model for haptic interaction adapted to deformable objects. Such approach allows multiple users to stably interact with a same object while feeling the influence of other users on the same object. Experimental results employing a PHANTOM haptic interface are proposed for a simple example.

1 Introduction

Simulation of deformable objects has been widely studied in the past decades. Haptic interaction with deformable objects is however a more recent field of research, the first results dating back to the mid 90's. As of today no solution has been accepted as a standard and many open problems still exist.

Finding the right trade-off between the level of precision allowed by a specific simulation technique and its speed has been the main challenge in creating realistic haptic interaction with deformable objects. High precision techniques, such as those based on the FEM, tend to be very slow, strongly limiting the haptic interaction servo-rates and compromising the device's stability. Simpler techniques, such as spring and masses, are typically not feasible for applications where a high level of realism is necessary, such as surgical training.

Stable haptic interaction has been widely studied in the last decade. The first efforts in this direction have dealt with simple rigid objects, such as the *virtual wall* [15, 6, 10]. While these results are not directly applicable to complex virtual environments, such as the ones considered hereafter, they have given great insight on how haptic devices work. In [6] Colgate has shown the existence of a link between servo-rates used to control a haptic device and the range of impedance that the device can display in a stable manner ($Z-$width of the device). Higher servo rates are needed in the case of stiffer objects while lower servo-rates can be used in the case of more complaint objects. As a consequence of this it is common practice to drive haptic devices at rates of at least 1KHz.

Various solutions have been proposed in order to obtain stable haptic interaction with slowly simulated deformable objects. All of such solutions are based on finding ways to simplify existing high precision techniques to a level that allows for haptic rates ($\leq$ 1KHz) while not strongly limiting their level of precision.

Burdea et al. [9] as well as Hayward et al. [2] and Zhuang et al. [21] all propose simulating deformable objects using different meshes updated at different frequencies. This allows for local speed and precision.

Pre-computation has also been employed by various authors. James et al. [11] and Cotin et al. [19] propose various ways to compute interaction forces using offline precomputed functions.

The drawback of all the solutions above is that haptic rendering algorithms and simulation techniques become inter-dependant. In general it is not possible to use the same haptic rendering algorithms with different simulation techniques.

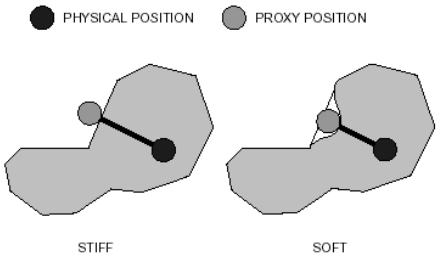

Figure 1. Using a proxy with a non-rigid object: the proxy is still computed using the static image of the object (this picture is taken from [18])

Solutions that are independent from the particular simulation technique used have also been proposed in the past.

Bosdogan et al. [5] as well as Ruspini et al. [17] compute forces proportional to the penetration depth of the virtual probe controlled by the user with respect to the surface of the deformable object at rest. As a consequence the haptic loop is performed on a rigid representation of the deformable object and therefore can run at high servo rates.

Mazzella et al. [8] propose a data structure, called the Forcegrid, which is updated every 30Hz using an extrapolation algorithm that keeps track of all past interaction forces with the deformable object. In order to drive a haptic device at high servo-rates an algorithm performing interpolation of past forces is used.

The drawback of all of the solutions above is that they do not allow multiple users interacting with a same object to feel each other's influence. To better explain this point let us consider an example of two users touching a balloon filled with water in different points. Each user globally deforms the object. As a consequence of such global deformations each users is able to feel the other user's influence. Clearly this is not possible if interaction forces are pre-computed, based on past force values or based on a rigid shell of the object (see for instance Fig. 1 from [17]).

In this paper we propose a haptic rendering algorithm that is independent from the particular simulation technique used and allows a real multi-user interaction. The solution proposed is based on the concept of local model [1, 13, 3] adapted to the case of deformable objects. This paper extends the results presented in [7] to more complex local models.

2 Local models with deformable objects

One common practice that allows for high servo-rates while interacting with slowly simulated virtual environments is to decouple the haptic loop from the graphics and simulation loops. Various techniques have been proposed in the past in order to accomplish this [1, 13, 3]. The basic idea behind all of these solutions is to use a simple implicit function that approximates, to a good extent, a small part of the object being touched. More specifically such *intermediate representation*, or *local model*, represents the part of the object which is closest to the current position of the haptic device. Figure 2 gives an idea of this simple concept. Such

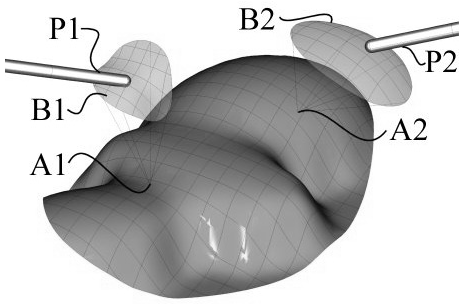

Figure 2. Local representation of object surface

model can be computed in the slow simulation loop without the user noticing discontinuities, since the frequency of the human hand movement is typically lower than the simulation frequency. Haptic rendering algorithms, such as the proxy or god-object [17, 22], can run at high rates thanks to the simplicity of the implicit surfaces involved.

Extending the local model technique to the case of deformable objects would have various advantages. It would allow us to decouple haptic loop from a slowly simulated deformable object simulation thus widening the $Z-$width of the device. Furthermore it would create a haptic rendering algorithm totally independent from the simulation technique used. Such an extension is however non-trivial in the case of a multi-user scenario.

In order for different users to feel each others' influence while touching a deformable object a local model computed using the current surface representation of the object should be used. By doing so, any global deformation on the object (due to any user) can be felt by all other users. This however complicates the overall stability of the system.

Computing a local model for rigid objects (or for "soft rigid objects") can be seen as an "open loop" problem. Given a new probe position inside the VE, a new local model can be computed solely based on geometric considerations. The same does not apply to deformable objects. The local model position depends on the state of the object's surface. This state, on the other hand, depends on the interaction force between user and virtual object, i.e. on the

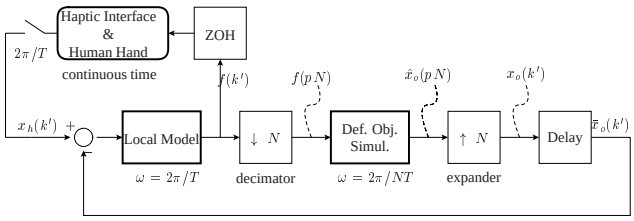

Figure 3. All the blocks that are typically present in a deformable object simulation with haptic feedback. Note that two separate closed loops exist.

local model position. Hence a closed loop is created. Such closed loop can become unstable, as discussed in the following, thus driving both the VE and the haptic interface in a vibrating state that completely destroys any sense of realism. In order to avoid these problems a "smarter" local model, one not solely based on geometrical considerations, must be defined.

2.1 Mathematical description of the problem

The closed loop mentioned above can be rigorously described. Consider the simple case of a one-dimensional deformable object, i.e. a spring, used in conjunction with different types of local models. We will analyze this in a multirate system framework.

Typically simulations comprising both deformable objects and haptic interfaces feature two separate loops. The haptic loop is a process that reads the new position of the haptic interface while it's being moved by the human operator, computes the new interaction force with the VE and writes such force to the haptic interface. Such process runs at high servo rates (e.g. 1KHz), as previously described, in order for the $Z-$width that device can display to be maximized. The simulation loop is a process that computes how the interaction force between haptic interface and VE influences the deformable object surface. Such process is usually slow due to the complexity of the simulated environment (e.g. 20Hz). For simplicity such loops are considered synchronized. The simulation engine receives data from the haptic interface every NT secs and responds to such data every NT secs while the haptic loop runs every T secs. The overall structure of such loops is depicted in Fig. 3 where the N–fold decimator $f(pN)$, being p an integer number, of the high rate signal $f(k')$ and the N–fold expander

$$x_o(k') = \begin{cases} \hat{x}_o(pN), & \text{if } k' = pN \\ 0, & \text{otherwise} \end{cases}$$

have been used to model the multirate system consisting of two synchronized dynamics, the VE simulation engine running at low rate $\frac{2\pi}{T}$ (20Hz) and the local model which is N times faster $\frac{2\pi}{NT}$ (1KHz).

Since we are considering a single dimensional case, the position of the local model, and of the proxy used on it, are coincident with the free end of the object (see Fig. 4), and are thus computed 20 times per second. As a first approximation let us suppose that the local model is based on a simple spring, i.e. there is always a purely elastic element between proxy and HI positions. Thus given a new HI position x_h every T secs, the corresponding interaction force to the user is given by $K_h(x_h - x_o)$, where K_h denotes the local model stiffness. Such force is then sampled, every NT secs, by the simulation block, which returns a new deformable object surface position x_o, and thus a new local model position, after NT secs. We assume the deformable

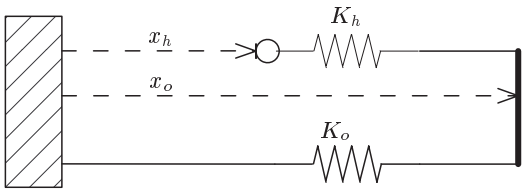

Figure 4. The mechanical model of the haptic interface interacting with a deformable object.

object to have local stiffness K_o. It is important to note that the deformable object surface position used to compute the interaction force that is fed to the HI is constant for periods of NT seconds, i.e. the local model position $\bar{x}_o$ at a given step k is given by

$$\begin{aligned} \bar{x}_o(k') &= x_o(k'T - T_d(k')) \\ T_d(k') &= (k' - \lfloor k'/N \rfloor N)T \end{aligned} \tag{1}$$

where $\lfloor a \rfloor$ denotes the integer part function. This type of behavior is often referred to as a *sawtooth* delay, and has been studied for instance for internet-based teleoperation [4, 16].

While the time domain formulation of the problem makes it easier to analyze the system, the study of the system stability is better approached by employing Z-transforms. Note that in the following we will consider the stability of the simulation and haptic loop as independent from each other. While this is not sufficient to ensure overall stability of the haptic interaction it is a necessary condition. Experimental data supports the results presented in the following. Work is currently being performed to analyze the overall system stability.

Let $L(z)$ be the discrete transfer function representing the Local Model force algorithm, and $D(z)$ the discrete

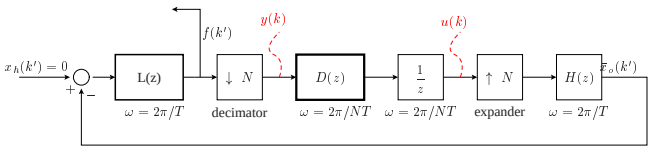

Figure 5. Simulation loop with generic discrete time transfer functions.

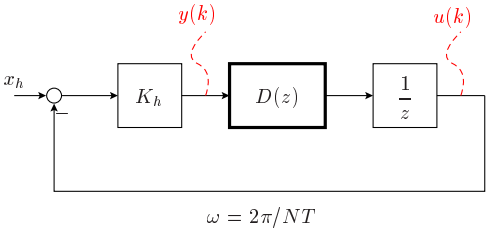

Figure 6. The virtual environment simulation loop reduces to a simple NT-sampled control loop.

transfer function representing the deformable object surface algorithm. The multiple rates in the system ($L(z)$ refers to a period of T seconds while $D(z)$ to a period of NT seconds) and the sawtooth delay make the stability analysis not obvious.

It can be shown that the sawtooth delay is equivalent to the combination of a pure NT delay before the expander in Fig. 3 and a discrete-time zero-order hold whose impulse response is

$$H(z) = 1 + z^{-1} + z^{-2} + \cdots + z^{-(N-1)} \qquad (2)$$

after the expander, i.e. with sampling period T as reported in Fig. 5.

As a consequence of this and of the assumption that the two loops are synchronized the stability analysis for the simulation loop becomes simpler. In the case of $L(z) = K_h$ (as depicted in Fig. 4), i.e. when the haptic interaction with local model is modeled as a simple stiffness K_h, it is an easy matter to verify that the input-output relationship between the slow-rate signals $u(k)$ and $y(k)$ in Fig. 5 is given by $y(k) = -K_h\delta(k)$. In fact when $u(k) = \delta(k)$, $\bar{x}_o$ is a sequence of N δ's,

$$\bar{x}_o(k) = \sum_{i=0}^{N-1} \delta(k' - i) \qquad (3)$$

which is then scaled by $-K_h$ and finally decimated as

$$y(k) = -K_h\delta(k).$$

This yields to substitute in Fig. 5, the block sequence from $u(k)$ to $y(k)$ with a simple gain $-K_h$ as shown in Fig. 6.

Therefore, it has been shown that for a linear elastic model of the deformable object, the virtual environment simulation loop analysis reduces to study the simple control loop at the lower rate $\omega = \frac{2\pi}{NT}$

$$G(z) = \frac{1}{1 + \frac{K_o}{K_h} z}$$

whose asymptotic stability is guaranteed if and only if

$$K_h < K_o \qquad (4)$$

3 Problems with this approach

Simply picking K_h and K_o to satisfy relationship 4 in order to obtain a stable behavior for the deformable object is not enough to ensure a realistic haptic experience. Stability is in fact only one aspect of a satisfactory system response. In the following we will closely examine both transient and steady state response for our system as well as the trade-offs between such behaviors.

3.1 Transient response

Factors characterizing the transient response of a dynamic system, such as settling time and overshoot, play a key role in the case of realistic haptic interaction. In reality in fact a purely elastic deformable object will assume a new surface configuration instantaneously and without vibrating. In our case, however, this might not always be true and certain sets of parameters might lead to noticeable oscillations (see Fig. 8). Considering the transfer function (2.1), it appears obvious that settling time (and overshoot) grows with $\frac{K_h}{K_o}$ and thus, to limit such effect, it should be $K_o \gg K_h$ whenever possible.

3.2 Steady state response

Factors characterizing the steady state response of our system are equally important in order to obtain an overall sense of realism. In general the stiffness perceived at steady state by the user, calculated as the steady state force divided by the deformable object surface position change, is always equal to K_o. However, due to the nature of impedance devices [20], a position error, defined as the distance between proxy and haptic device position, is always present. While such error may go unnoticed at times due to the limitations of the human position system, certain thresholds exists above which the sense of realism is lost.

Such error is equivalent, in our specific case, to the steady state tracking error for a step input applied to sys-

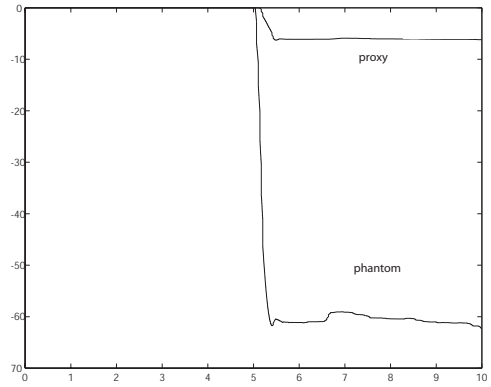

Figure 7. Linear spring response to a step input ($K_o = 0.9$ and $K_h = 0.1$)

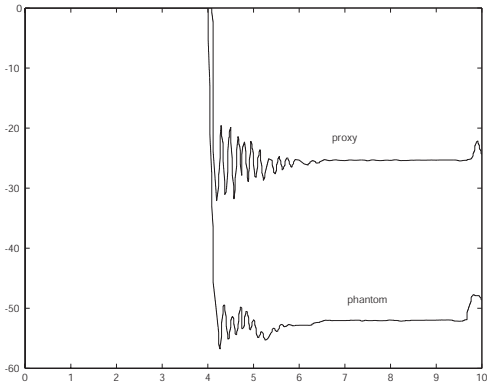

Figure 8. Linear spring response to a step input ($K_o = 0.2$ and $K_h = 0.19$)

tem (2.1), which is equal to

$$\bar{x}_o = \frac{K_h}{K_h + K_o} x_h \tag{5}$$

In order to limit the extent of such error to be under the human position system perception threshold, it should be $K_o \ll K_h$. This would however drive the system to instability and thus K_h should be chosen to be the closest possible to K_o.

3.3 Trade-off between steady state and transient response

A tradeoff between transient and steady state response exists, as mentioned above and has been tested experimentally. In the case of $K_o \gg K_h$, depicted in Fig. (7) where $K_o = 9K_h$, the system response has virtually null settling

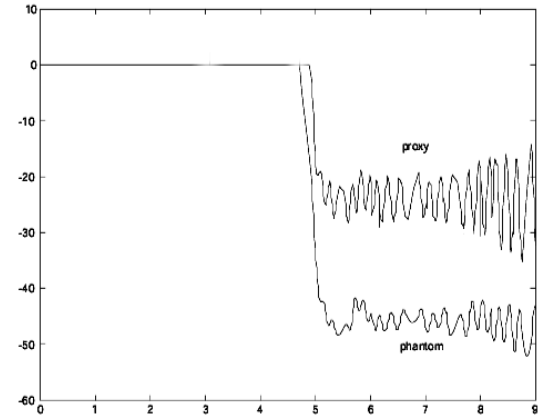

Figure 9. Linear spring response to a step input ($K_o = 0.2$ and $K_h = 0.22$)

time but the steady state error is very large since the surface moves only 1cm for a physical movement of the haptic device equal to 10 centimeters.

In the case of $K_o \simeq K_h$, depicted in Fig. (8) where $K_h = 0.19$ and $K_o = 0.2$, the system response has a settling time of about 2 seconds and it is thus clearly perceived as an oscillatory effect. The steady state distance between proxy and phantom position is however better, being equal to 25mm for a surface deformation of 12mm.

3.4 A solution for steady state impedance: using an Integral term

The main problem with the a purely elastic local model is that the extent the haptic device must penetrate inside the object, in order to obtain a non-oscillatory force feedback, cannot be set. In order to solve this problem a new local model, enhanced with an integral term in parallel to the proportional term used before, has been implemented. A similar idea has been proposed and implemented by T. Massie in [14] for the case of rigid virtual walls.

Considering Fig. 5, let us consider the $Z-$transform of the local model to be

$$L(z) = K_h + \frac{TI}{z-1} = K_h + \frac{TIz^{-1}}{1-z^{-1}} \tag{6}$$

where coefficient I weighs the integral action of the local model. Recall that both the proportional term K_h and the discrete integrator term $\frac{IT}{z-1}$, run at a high servo rate ($\frac{2\pi}{T}$) while the deformable object simulation runs slowly ($\frac{2\pi}{NT}$). The basic idea of such scheme is that the computed force that is fed to the user and to the deformable object simulation is still dependant on the haptic device penetration with

respect to the local model. However the integral term integrates such error through time eliminating it, i.e. reducing the distance between proxy and haptic device position. This is accomplished through a dual effect: on one side the force fed back to the user by the haptic device grows, pushing the user's hand towards the proxy position; on the other side a larger force is fed to the deformable object algorithm and thus tends to further indent the simulated surface, i.e. move proxy and hand position closer to each other.

Clearly the stability condition (4) will not hold true any longer since the discrete transfer function describing the local model is no longer a simple proportional term. As for the simple proportional local model, it is possible to obtain a closed loop transfer function representing the simulation loop by computing the impulse response of the block sequence from $u(k)$ to $y(k)$ (see Fig. 6). In this case we obtain open and closed loop transfer function of the NT-sampled system as

$$G(z) = \left(\frac{NTI}{z-1} + K_h\right) \frac{1}{zK_o} \tag{7}$$

and

$$F(z) = \frac{G(z)}{1+G(z)} \tag{8}$$

In order to study how the stability of the simulation loop depends on the parameters K_o, K_h and I Jury's method has been applied [12]. In order for the simulation loop to be stable the following conditions must hold

$$\begin{cases} |INT - K_h| < K_o \\ \frac{INT-K_h}{K_o} > \frac{K_o-K_h}{K_o} - 1 \\ \frac{INT-K_h}{K_o} > \frac{K_h-K_o}{K_o} - 1 \end{cases} \tag{9}$$

keeping in mind that $K_o > 0$, $K_h > 0$ and $I > 0$. These relationships lead to a stability area for parameters K_h and INT for a given K_o, as the one depicted in Fig. 10.

Picking K_h and I inside such area will lead to an asymptotically stable behavior for the simulation loop, unstable otherwise. It is important to note that such area has the same shape for different values of K_o. Larger values of K_o imply areas scaled upward. A stability volume as the one depicted in Fig. 11 can be defined[1]. Interestingly the simulation loop tends to have larger slices for higher values of K_o, i.e. it is easier to stabilize it when dealing with stiffer objects. This is an unexpected result in a way since typically HIs are unstable when simulating very stiff objects. The scalability of such areas is a consequence of the fact that in (7), $G(z)$ can be expressed with K_h and I normalized with respect to K_o. This is a very important note because it shows that picking

[1]Note that only some slices of the solid are shown to better give the idea of the volume

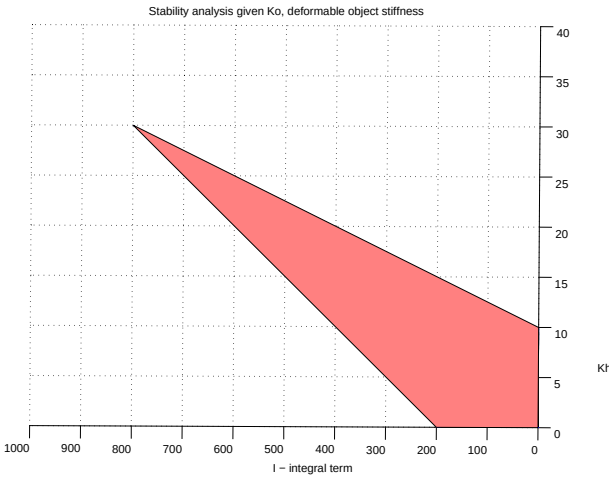

Figure 10. The stability area for parameters K_h and INT given K_o of the object being touched

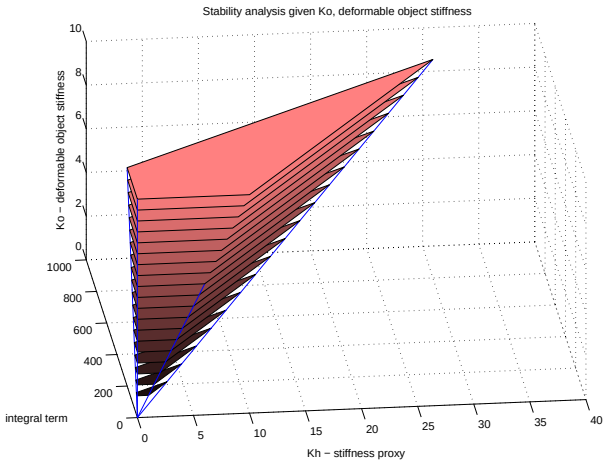

Figure 11. The stability area for parameters K_h and INT while K_o varies

points that have the same relative positions on any volume slice, i.e. the points of intersection between a line contained inside the volume and ending in the reference system origin, leads to a same transient response type for any value of K_o. Hence the transient response can be studied for one single slice and then applied to all slices obtaining the same results.

It is also important to note that while for the stability analysis we must consider the integral term to be $\frac{NTI}{z-1}$ in the real algorithm we will use $\frac{TI}{z-1}$ since the local model is computed at a 1KHz rate.

Similar results can be obtained when using a continuous order hold instead of a zero order hold. In such case the surface of the object does not jump to its new position every NT seconds but linearly moves between successive positions. However an extra delay is introduced. While this does not compromise the simulation loop stability it may be noticeable in the case of large values of NT. Such analysis is not reported due to limited space.

3.5 Some notes on PI local models

While in theory the scheme proposed above perfectly accomplishes the proposed goals, in practice some changes must be made in order to adapt it to the real case of haptic interaction.

In order to prevent the surface of the deformable object to feel unrealistically active the feedback force to the user should be null when the user is not in contact with the object, i.e. when the proxy and haptic device position coincide. As Thomas Massie pointed out in his paper, however, this should not happen instantaneously when pulling out of the object but should be a somewhat linear process. One possible approach that has been successfully tested is based on driving the integral term to zero linearly over a a-priori fixed time interval. If such interval is picked to be comparable to NT the visual delay introduced by this solution will not be noticeable. Moreover the object surface will not pop up to its original configuration but will do so in a more gradual way. Typical time intervals can be small multiples of NT. Longer periods introduce higher delays on the surface movement but lead to a nicer more gradual force feedback effect.

As a consequence of this the proxy position should never reach the current haptic device position, i.e. the integral term should not completely erase the distance between such points. In such case, in fact, the force would drop down to zero and the object would move back to its original configuration. In order to avoid this, the error fed to the integral block should not be $e = x_{proxy} - x_{haptic}$ but a smaller quantity. Feeding the I term with $e - \theta$, where θ is given by

$$\theta = \begin{cases} 0 & \text{if } e < 0, \text{ i.e. no contact} \\ \bar{\theta} & \text{if } e > 0, \text{ i.e. when there is contact} \end{cases} \quad (10)$$

and $\bar{\theta}$ is the maximum error between proxy and device position that we decide to tolerate at steady-state, accomplishes such effect.

The side effect of using a threshold is that a steady state error will always be present. However the extent of such error is always controllable and can be made very small. In the end every force will be affected, at steady state, by a small offset error, which is an acceptable compromise on the overall performances of the system.

In practice a variable threshold θ has been used. More specifically θ can be made dependant on the HI device penetration speed, i.e. $\dot{e} = \dot{x_o} - \dot{x_h}$, or/and on the current total value of the integral term. In the current implementation θ is given by

$$\theta = \begin{cases} 0 & \text{if } e < 0, \text{ i.e. no contact} \\ \bar{\theta}_1 & \text{if } e > 0 \text{ and } (\dot{x_o} - \dot{x_h}) < 0 \\ \frac{\bar{\theta}_2}{I_{tot}} & \text{if } e > 0 \text{ and } (\dot{x_o} - \dot{x_h}) > 0 \text{ and } I_{tot} < \bar{I}_{tot} \\ \bar{\theta}_2 & \text{if } e > 0 \text{ and } (\dot{x_o} - \dot{x_h}) > 0 \text{ and } I_{tot} > \bar{I}_{tot} \end{cases} \quad (11)$$

hence θ is zero when there is no contact with the object; it grows linearly when penetrating the object up to a certain limit and then is fixed to such limit; it is set to different values when entering an object and when exiting an object. The value of θ should be linearly growing to its set value $\bar{\theta}_2$ while penetrating the object because otherwise the integral term would not be active for penetrations smaller than $\bar{\theta}_2$. Moreover it is very useful to have different values of θ depending on the penetration speed and sign. In fact it is usually desirable to have small thresholds while penetrating an object, in order to limit the steady state force offset, and large thresholds while exiting the object, in order to limit the cases when the penetration error drops to zero and the surface is brought back to its original configuration.

4 Local model algorithm

The algorithm we have used in conjunction with slowly simulated deformable objects can be roughly described as in the following:

Algorithm:

Step 1 the local model is calculated inside the slow simulation loop using the current surface configuration of the deformable object, as if it was static, as well as the current haptic interface position; the algorithm proposed in [3] is currently being employed.

Step 2 K_h and I are chosen inside the volume described in Fig. 11. Note that such parameters can always be

chosen in a robust way so that stability is not lost for errors on K_o.

Step 3 the force that the local model returns to deformable object simulation, for it to compute its next configuration, is evaluated as an weighted average function of the N forces returned by the local model to the haptic device inside an NT period.

5 Conclusions

This paper describes new techniques allowing users to haptically interact with a set of deformable slowly-simulated objects. In particular the paper analyzes possible causes of instability that are specific to the case of deformable objects. The techniques proposed in this paper are based on the concepts of local model for haptic interaction previously introduced by various researchers, which are adapted for the particular case of deformable objects. Experimental results featuring a simple one-dimensional scenario are reported.

Future work will focus on obtaining more general stability conditions for the overall system, i.e. considering both haptic and simulation loop together. Moreover the proposed stability conditions will b extended to the case of more complex VE.

References

[1] Y. Adachi, T. Kumano, and K. Ogino. Intermediate representation for stiff virtual objects. In *IEEE Virtual Reality Annual Intl. Symposium*, pages 203–210, Research Triangle Park, N. Carolina, March 1995.

[2] O. Astley and V. Hayward. Multirate haptic simulation achieved by coupling finite element meshes through norton equivalents. In *Proceedings of IEEE Conference on Robotics and Automation (ICRA98)*, pages 989–994, Leuven, Belgium, 1998.

[3] R. Balaniuk. Using fast local modeling to buffer haptic data. In *Proceedings of Fourth PHANTOM Users Group Workshop-PUG99*, 1999.

[4] K. Brady and T.-J. Tarn. Internet-based teleoperation. In *Proc. of the 2001 IEEE Int. Conf. Robotics and Automation*, Seul, Korea, May 2001.

[5] M. S. S. S. C. Bosdogan, C. Ho and S. Dawson. Force interaction in laparoscopic simulation: Haptics rendering of soft tissues. In *Proceedings of Medicine Meets Virtual Reality (MMVR '98)*, pages 28–31, January 1998.

[6] E. Colgate, P. Grafing, M. Stanley, and G. Schenkel. Implementation of stiff virtual walls in force-reflecting interfaces. In *Proceedings of VRAIS*, pages 202–208, Seattle, WA, September 1993.

[7] D. P. F. Barbagli and K. Salisbury. Multirate analysis of haptic interaction stability with deformable objects. In *Proceedings of the IEEE 2002 Conference on Decision and Control (CDC02)*, Las Vegas, NE, December 2002.

[8] K. M. F. Mazzella and J. Latombe. The forcegrid: A buffer structure for haptic interaction with virtual elastic objects. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA02)*, Seoul, Korea, may 2002.

[9] V. P. G. Burdea, G. Patounakis and R. Weiss. Virtual reality training for the diagnosis of prostate cancer. In *Proceerings of IEEE Symposium on Virtual Reality and Applications (VRAIS'98)*, pages 190–197, Atlanta, GA, March 1998.

[10] B. Gillespie, M. Cutkosky, and S. UserSpeci. Rendering of the virtual wall. In *Proceedings of the ASME International Mechanical Engineering Conference and Exposition*, volume 58, pages 397–406, Atlanta, GA, November 1996. ASME.

[11] D. James and D. Pai. A unified treatment of elastostatic contact simulation for real time haptics. *haptics-e, The Electronic Journal of Haptics Research*, 2(1), September 2001.

[12] E. I. Jury. *Theory and Applications of the z-Transform Method*. Wiley, New York, 1964.

[13] W. Mark, S. Randolph, M. Finch, J. V. Verth, and R. Taylor. Adding force feedback to graphics systems: Issues and solutions. In *Computer Graphics: Proceedings of SIGGRAPH'96*, pages 447–452, August 1996.

[14] T. Massie. Taking the mush out of haptics with infinitely stiff walls. In J. Salisbury and M.A.Srinivisan, editors, *Proceedings of the First PHANToM Users Group workshop*, Dedham, MA, September 1996.

[15] M. Minsky, M. Ouh-young, O. Steele, F. Brooks, and J. Behensky. Feeling and seeing: issues in force display. *Computer Graphics*, 24(2):235–243, 1990.

[16] G. Niemeyer and J. Slotine. Toward force-reflecting teleoperation over the internet. In *IEEE International Conference on Robotics and Automation (ICRA)*, Leuven, Belgium, 1998.

[17] D. C. Ruspini, K. Kolarov, and O. Khatib. The haptic display of complex graphical environments. *Computer Graphics*, 31(Annual Conference Series):345–352, 1997.

[18] D. C. Ruspini, K. Kolarov, and O. Khatib. The haptic display of complex graphical environments. In *Computer Graphics (SIGGRAPH 97 Conference Proceedings)*, pages 345–352. ACM SIGGRAPH, 1997.

[19] H. D. S. Cotin and N. Ayache. A hybrid elastic model allowing real-time cutting, deformation and force feedback for surgery training and simulation. *The Visual Computer*, 16(8):437–452, 2000.

[20] T. Yoshikawa, Y. Yokokohji, T. Matsumoto, and X. Zheng. Display of feel for the manipulation of dynamic virtual objects. *Trans. ASME, Journal of Dynamic Systems, Measurement, and Control*, 117(4):554–558, 1995.

[21] Y. Zhuang and J. Canny. Haptic interaction with global deformations. In *IEEE International Conference on Robotics and Automations (ICRA00)*, pages 24–28, San Francisco, CA, April 2000.

[22] C. Zilles and J. Salisbury. A constraintbased god-object method for haptic display. In *Proc. IEE/RSJ International Conference on Intelligent Robots and Systems, Human Robot Interaction, and Cooperative Robots*, volume 3, pages 146–151, 1995.