Smooth Haptic Interaction from Discontinuous Simulation Data

Jesper MOSEGAARD[a, b, 1], Bo Søndergaard CARSTENSEN[c],
Allan RASMUSSON[c], Thomas Sangild SØRENSEN[c]
[a]*Institute of Information and Media Studies, University of Aarhus, Denmark*
[b]*Department of Computer Science, University of Aarhus, Denmark*
[c]*Centre for Advanced Visualisation and Interaction, University of Aarhus, Denmark*

Abstract. When a physical simulation that relies on haptic interaction is temporarily paused due to e.g. visualization, noticeable discontinuities are introduced in the interaction as well as the haptic-feedback. The source of this problem is a discrepancy between the notion of time in the simulation and in the real world. In this paper we analyze the general problem of executing simulation steps that each represent a constant amount of simulation time but are distributed non-uniformly in real world time. We have devised a solution that realigns the two notions of time, hereby insuring smooth interaction data and haptic feedback.

Keywords. Haptic rendering, noise-filtering, discontinuous simulation, GPU acceleration, surgical simulation

Introduction

In this paper we deal with a specific problem within the field of haptic-rendering, namely to achieve smooth haptic interaction from simulation-data that is sampled non-uniformly over time. The problem occurs specifically in GPU (graphics processing unit) based simulations where execution of several simulation steps for each visualization step is common [1,2]. The problem arises since the simulation is briefly suspended during each visualization step. The user continues to move the interaction device however, and when the simulation resumes, the position of the virtual instrument, as seen from the simulation timeline, will appear to have moved extraordinarily (or discontinuously) since the previous simulation step. This introduces noise and "jumps" in the interaction with the simulated tissue as well as in the haptic feedback. In Figure 1, graphs based on measurements demonstrate this undesired effect. The problem is clearly exhibited in the relatively large perturbations (solid line) in interaction *a)* and haptic-feedback *b)*. Notice how the errors from Figure 1 *a)* are more distinct in Figure 1 *b)* since the physical simulation collects perturbations as additional energy whereby oscillations occur. As humans are very sensitive to even the smallest discontinuities in the haptic rendering, this is an important problem to solve.

A naive "solution" would be to try and smooth the haptic forces through a filter. This is not a viable solution unfortunately, since a substantial amount of smoothing

[1] Corresponding Author: Jesper Mosegaard, Institute of Information and Media Studies, Helsingforsgade 14, 8200 Aarhus N, Denmark; E-mail mosegard@daimi.au.dk

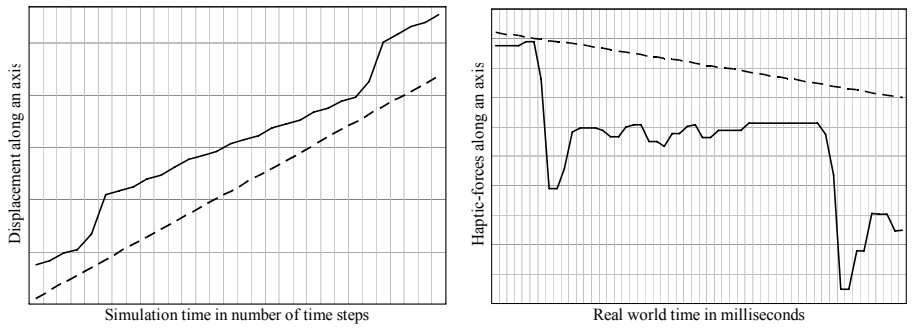

Figure 1. An interaction-device is moved with constant velocity (programmed) along an axis in the GPU-based simulator. The solid graphs are based on measurements of *a)* motion of interaction in simulation time and *b)* haptic-feedback in real world time. Delays in the execution of simulation-steps are clearly seen as noise and "jumps" in the two graphs. The dotted graphs are based on measurements of the simulator using the proposed mapping (Figure 3) to correct *a)* the motion of interaction in simulation time and the proposed inverse-mapping (Figure 4) to correct *b)* the haptic-feedback in real-world time. Notice how the dotted graphs are very close to the theoretically optimal linear graph of constant velocity.

would be required, which again seriously dampens the desired haptic changes that should be felt when manipulating the simulated tissue. In this paper we analyze the problem and present a method to solve the actual problem rather than repairing its symptoms.

This specific problem has not previously been described in the field of haptic-rendering, although other related subjects of interpolation and extrapolation, to achieve a sufficient update rate of more than 500hz, has been dealt with in e.g. [3,4].

1. The GPU-based surgical simulator

We observed the behavior described in the introduction in GPU-based simulators, which effectively utilize the large amount of computational power in consumer-level programmable graphics processing units for the computation of tissue deformation. On the GPU, several simulation-steps can be performed for each visualization-step due to much faster computation of each simulation step compared to a CPU implementation [1,2]. The large number of simulation-steps per second ensures a fast, responsive, and stabile simulation. To obtain the highest quality of haptic feedback, forces are collected after each simulation step. In [2] a Geforce 7800 GTX is reported to sustain a simulation rate (and consequently a haptic feedback rate) of 450 Hz for a system of 20.270 particles (of 18 springs each) with two haptic-devices. This system was used to simulate surgical procedures on congenital heart defects. A visualization of 137.490 faces is executed at 30 Hz. This corresponds to 15 simulations-steps for each visualization step. The non-uniform distribution of simulation-steps not only arises from these pauses during visualization, but also from internal resource priorities on the GPU. That is, even a pure sequence of simulation steps may not arrive at regular intervals.

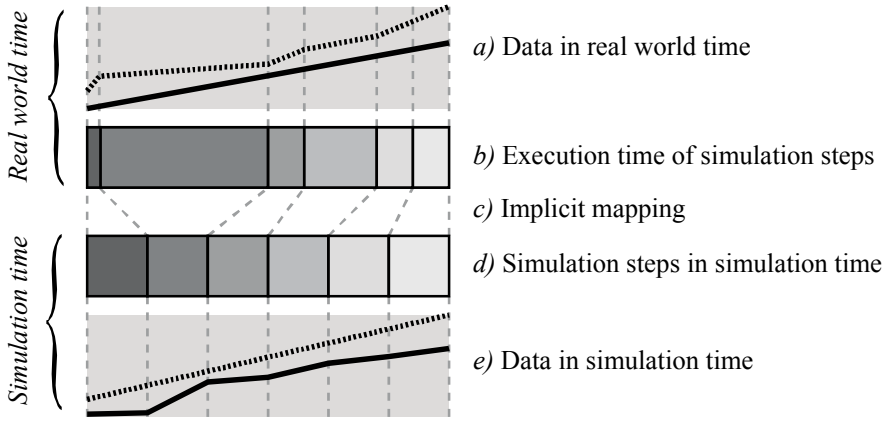

Figure 2. This figure illustrates the errors introduced when directly applying data from real world time in simulation time (and visa versa). The dotted graph is linear in simulation time, while the solid graph is linear in real world time. The mapping between the two notions of time *c)* is applied through the relationship between constant sized simulation-steps as seen from the simulation *d)* and the non-regular execution times as seen from real world time *b)*.

2. Analysis

In a computer simulation we have two notions of time; real world time and simulation time. The real world time is the time as measured by the internal clock of a computer and experienced by the user. The simulation time is the time "measured" by the numerical simulation of some phenomenon. The surgical simulation we consider, uses a discretized notion of simulation-time where each simulation-step increments the simulation-time by a *constant* factor. We analyze the problems arising when the simulation-steps are non-uniformly distributed in real world time but uniformly distributed along the simulation time-line. These issues become clear when specific actions in the real world manipulate the simulated object (and visa versa), specifically during interaction with simulated tissue and the corresponding haptic feedback.

To illustrate, consider directly using interaction data sampled at the beginning of each simulation step (in real world time) as input to the simulation. The interaction data would not be sampled at constant intervals due to run-time variation as mentioned previously. However, in the simulation the sampled data is assumed to represent interaction sampled in regular intervals. This transformation on interaction data is illustrated in Figure 2. The reader should follow the transformation of the fat line from real world time (Figure 2 *a)*) to simulation time (Figure 2 *e)*), where the fat line has become severely distorted. The same problem affects information calculated by the simulation but applied in the real world, such as haptic forces. This is illustrated by the transformation of the dotted fat line in simulation time (Figure 2 *e)*) to real world time (Figure 2 *a)*). Our solution to both problems is to introduce a mapping from both real world time to simulation time, as well as simulation time to real world time.

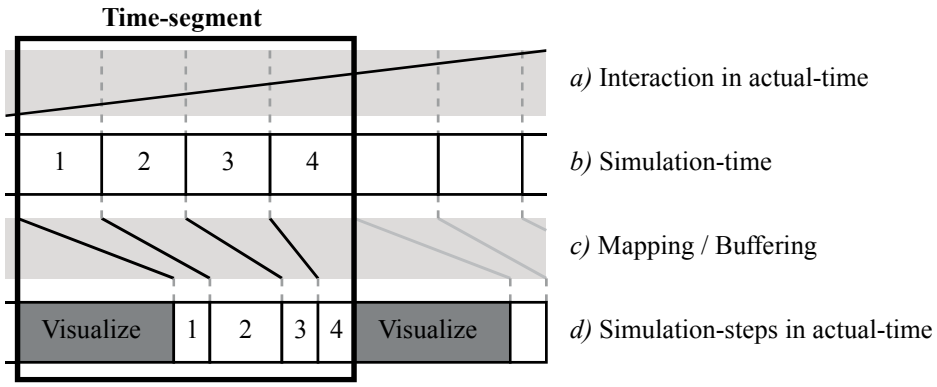

Figure 3. Consider a time-segment of one visualization-step and four simulation-steps. *a)* shows the interaction-data in real world time. *b)* shows the division of the time-segment into four equal parts, each corresponding to one of the four simulation steps. *c)* shows the mapping from execution time of simulations steps in d) and the corresponding division of the time-segment in *b)*. For each simulation step 1-4 in *d)* the mapping in *c)* provides the actual time *a)* at which to retrieve interaction-data from.

We can now see why the above identified issues are rarely problematic in conventional CPU-based simulators (See [5] for a good introduction). In many such simulators a simulation-step is always followed directly by a visualization step, thereby introducing the same delay of visualization for each simulation-step. Interaction can consequently be delivered directly to the simulation and haptic-forces can be retrieved directly.

3. Method

Without loss of generality we now further explore the case of the GPU-based simulator, in which the visualization step was the largest source of delay. A visualization-step and a set of simulation steps (up until the next visualization step) will be named a *time-segment*, see Figure 3. We assume that the execution time of a time-segment is constant – or slowly changing. The calculation of the execution time is based on a weighted average of several successive time-segments. We propose to construct a mapping (Figure 3 *c)*) from real world time to simulation time as follows. First, let us consider how the execution of simulation steps should have been distributed, in real world time, across a time-segment to reflect the constant sized time-step in the simulation. Since each time-step in the simulation is constant, the execution of simulation steps should also be uniformly distributed across a time-segment. By splitting a time-segment equally into a number of time slots, one for each simulation-step, we consequently know the point of time, in real world time, that the simulation-step corresponds to. Figure 3 *b)* can be said to show both this division into time slots, as well as the time-steps as seen in simulation-time. The actual execution time of a simulation step (Figure 3 *d)*) cannot be changed though. Instead we retrieve the correct interaction prior to the execution of a simulation step. We have chosen a time-segment such that a large delay (the visualization) is in the beginning of the time-segment (Figure 3 *d)*). This means that interaction-data is available before it is required in the simulation. We consequently buffer that incoming interaction-data until the corresponding simulation-step is ready to be executed. When the simulation-step is

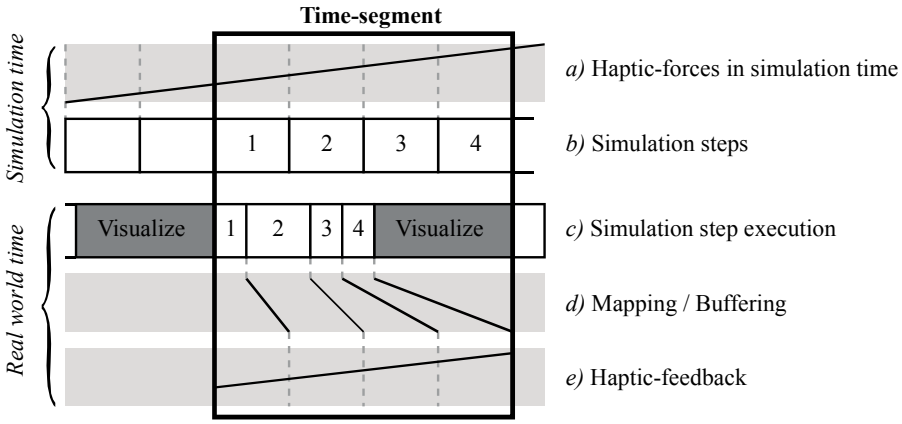

Figure 4. This figure represent the inverse mapping of Figure 3, now going from simulation time to real world time. The haptic-force in *a)*, calculated in simulation time *b)*, are mapped *d)* to real world time through a uniform distribution the simulation-steps *c)* in a time-segment. The result is that haptic-forces calculated in simulation-time *a)* are mapped correctly to real world time *e)*.

about to be executed, it retrieves the buffered interaction data according to its assigned time slot in the time-segment, see Figure 3 *c)*. The buffer is constructed of sampled interaction values and supports linear interpolation of these values when data is retrieved from the buffer.

Although the interaction is now correctly aligned, the haptic-forces calculated after each simulation step, should not be directly applied to the haptic hardware. As discussed in the previous section, we must apply the inverse mapping to the calculated forces to transfer data from simulation-time to real world time. Refer again to Figure 2, and follow the fat dotted line (which could represent haptic-forces) in simulation time to real world time, and notice the severe perturbations if the inverse mapping is not applied. We consequently buffer the resulting haptic-forces, distribute the forces equally across a time-segment, and send them on to the haptic-hardware at regular intervals, see Figure 4.

Although the presented solution was specific to a scenario with one relatively long visualization step and a series of non-regular simulation steps, the method is generally applicable. The only demand is that a cycle of repeating execution times can be found. The buffering of data from either interaction or haptics must be large enough for the access of data in the buffer never to catch up with the buffered content.

4. Results and Discussion

The result of applying the proposed method is that interaction data from the haptic-device arrives at the simulation-step at correct simulation-time, delayed maximally with one visualization-step. Haptic-forces are recorded after each step of the simulation, and are realigned with real world time, delayed maximally one further visualization step. Since the interaction was already delayed one visualization step, the haptic-feedback is delayed two visualization steps compared to the interaction that caused it. In the example from [2] a visualization-step is executed in 11.1 ms. Hence, this delay in interaction is barely noticeable. The results of applying the proposed

scheme can be inspected in Figure 1 *a)* (the dotted line). The results on the quality of haptic-feedback can be inspected in Figure 1 *b)* (the dotted line). The results in both cases are actual measured values from the GPU-based surgical simulator [1], and are very close to the theoretical optimal linear line.

In our solution we have assumed that the execution time of each time-segment is constant or slowly changing since this is used to define the "length of time" to divide by the number of simulation steps. We have also assumed that a cycle of repeating execution times is present. In the case of the GPU accelerated simulator these assumption hold.

Practically all simulators are based on differential equations, and consequently solved by numerical integrators. The issues identified in this paper originate from the fact that the time-step was constant. There exist numerical integrators that support varying time-step lengths from one simulation step to the next. Those are most often used to actively guarantee a given precision of approximation though, and not as a *reaction* to expected run durations. The problem with this solution is that the time-step length directly influences the stability of the numerical integration. This is a critical drawback in a real-time simulation and hence not a viable solution.

The proposed scheme solves the problems of constant length time-steps, hereby maintaining the stability of the simulation.

Acknowledgments

We kindly acknowledge the funding we received from the Danish Research Council's Program Committee on IT-Research (grant #2059-03-0004) and The Danish Heart Foundation (grant #05-10-B359-A657-22265).

References

[1] J. Mosegaard, T.S. Sørensen. GPU accelerated surgical simulators for Complex Morphology. In proceedings: IEEE Virtual Reality (IEEE VR), Bonn, Germany, 2005; 147-53.

[2] T.S. Sørensen, J. Mosegaard. Haptic Feedback for the GPU-based Surgical Simulator. 14th Medicine Meets Virtual Reality 2006. Stud Health Technol Inform; 119:523-8.

[3] F. Mazzella, K. Montgomery, J. C. Latombe. The Forcegrid: A Buffer Structure for Haptic Interaction with Virtual Elastic Objects. ICRA 2002:939-46.

[4] G. Picinbono, J-C. Lombardo, H. Delingette, N. Ayache. Improving realism of a surgery simulator: linear anisotropic elasticity, complex interactions and force extrapolation. Journal of Visualisation and Computer Animation, 13(3):147-67.

[5] A. Liu, F. Tendick, K. Cleary and C. R. Kaufmann. A survey of surgical simulation: applications, technology, and education. Presence: Teleoperators and Virtual Environments. 2003;12(6):599–614.