The Forcegrid: A Buffer Structure for Haptic Interaction with Virtual Elastic Objects

Frédéric Mazzella[1] Kevin Montgomery[1] Jean-Claude Latombe[2]

(1) Stanford-NASA National Biocomputation Center, Stanford, CA, USA
(2) Computer Science Department, Stanford University, Stanford, CA, USA
Emails: {fred|kevin}@biocomp.stanford.edu and latombe@cs.stanford.edu

Abstract Real-time surgical simulation often requires computing the deformations of visco-elastic human-body tissue and generating both graphic and haptic renderings. As human users are more sensitive to small haptic discrepancies than visual ones, good force rendering requires a refresh rate of 500Hz or more, which is far beyond the capabilities of current simulators. Moreover, if the simulator runs on a remote server, limited bandwidth, network latency and variance further compromise the quality of haptic rendering. To address these problems, a new interpolation-extrapolation data structure – called the *forcegrid* – is proposed, which makes it possible to approximate forces at rates greater than 500Hz, regardless of the complexity of the simulated object. The forcegrid is designed for haptic interaction with continuous contact. It is a regular grid defined over the virtual workspace, in which every vertex contains a force. An extrapolation algorithm updates this force at every simulation cycle (~30Hz), while an interpolation algorithm computes the force to be rendered on the haptic device at the desired rate (~500Hz). The forcegrid has been successfully tested on various virtual deformable objects.

Keywords: elastic objects, surgical simulation, haptic interaction, virtual environments.

1. Introduction

Over the past few years there has been considerable research in modeling deformable objects. See [12, 13] for reviews, respectively on human tissue modeling and on cloth simulation. An important domain of application is surgical simulation, which can be used for training, planning operations, and predicting surgical outcomes. In many cases, especially training, it is highly desirable to perform the computer simulation in real-time.

Most human-body soft tissues are visco-elastic. Simulating their deformations consists of computing their successive shapes over time. The key feedback to the user is subjective realism. Often, graphic visualization alone is not enough; it is critical to also provide force feedback on a haptic device (a mechanical linkage with actuated joints). This device is used to position a model of the surgical instrument in a virtual environment and to render the reaction forces resulting from contacts between this instrument and objects (e.g., deformable tissue) in this environment. Combining graphic and haptic feedbacks provides users with a more realistic experience than visual feedback alone [1, 16, 19].

Real-time graphic animation requires the shape of the objects to be updated at 30 Hz or more. This is already a challenging problem. A number of modeling and computational techniques – e.g., mass-spring [6, 15, 21] and finite-element [3, 5] methods – have been proposed that provide reasonably satisfactory solutions for visco-elastic objects. A typical technique computes internal forces throughout the model of an object E and derives the deformation of E from these forces. A by-product of this computation is the reaction force exerted by E on the surgical instrument. However, as humans are more sensitive to small haptic discrepancies than to visual ones, good force rendering requires a refresh rate of 500Hz, or more, which is far beyond the capabilities of current simulators. In the near future, the increase in computational power is more likely to serve more accurate and visually pleasing graphic animations than to feed haptic devices at greater rates. Moreover, a fair prospect in many surgical applications is that the simulator will run on a remote computer, connected to the controller of the haptic device by a LAN or the Internet. Then, even small network latency, delays, or inconsistencies can compromise satisfactory haptic interaction.

To address these difficulties, we propose a new data structure – called the *forcegrid* – which makes it possible to approximate the reaction force to be rendered on the haptic device at rates greater than 500Hz, regardless of the complexity of the model of the simulated object. The forcegrid is a structure in which we cache previous results generated by the simulator and that the haptic device controller exploits to render forces at the appropriate rate.

It is a regular grid placed over the virtual workspace, in which every vertex contains a force. The two key algorithms are the force-filling algorithm, which extrapolates the results returned by the simulator to update the vertex forces, and the interpolation algorithm, which computes the force to be rendered. The forcegrid is independent of both the modeling method used by the simulator and the simulation rate.

Section 2 reviews prior related work. Section 3 introduces the concept of a forcegrid. Section 4 describes the interpolation function. Section 5 presents two force-filling algorithms – the *silent* and the *request* methods. Section 6 describes our implementation and our experiments with it. Finally, Section 7 discusses limitations of the current techniques and proposes remedies to eliminate them.

2. Previous Work

A number of different techniques have been proposed to improve the quality of haptic feedback during the interaction with an elastic object.

The technique in [4] computes a reaction force that is proportional to the penetration depth of the surgical instrument (relative to the surface of the deformable object at rest). The rendered force is thus only loosely related to the physical model of the object.

In [8] the haptic simulator achieves greater speed by using a much smaller mesh than the one used by the graphic simulator. One difficult issue is to make sure that the two different meshes provide consistent results.

In [9] fast online computations are achieved by pre-computing forces for a number of deformations. This is an appealing approach, but it suffers from two drawbacks. First, the contact point of the instrument with the object is often not known in advance. Second, forces are computed as linear combinations of relatively sparse pre-computed forces, while human-body tissues are often non-linear.

The method in [14] uses pre-computed Green's functions and fast low-rank updates to achieve real-time interaction with linear elastostatic objects. Further optimizations are needed to simulate large-scale models.

The approach in [11] combines a large-displacement strain tensor formulation (Green) with adaptive (in time and space) techniques.

In [20] extrapolation techniques over time and position are used to estimate reaction forces between two time steps of the graphic simulation. The forcegrid also uses some form of extrapolation technique to fill the forcegrid, but it makes better use of the data flow produced by the simulator. By caching the results successively produced by the simulator, it takes advantage of several of them to compute a new reaction force.

In the "buffer model" of [2], the haptic device interacts with a local physical model of the deformable object. This model is a numerical function whose parameters are periodically adjusted to optimize the fit with the data produced by the simulator. Our forcegrid bears some similarities with this buffer model, but it is a global data-caching structure rather than a local model. While every parameter adjustment in [2] leads to forgetting past results, the forcegrid exploits all previous simulation results. We believe that the forcegrid is easier to use, as it does not require selecting a numerical function adapted to the mechanical properties of the object. On the other hand, the buffer model of [2] is likely to give better results during a cutting operation or if the mechanical properties of the object have hysteresis (see Section 7).

3. The Concept of a Forcegrid

We consider a haptic device with three degrees of freedom, which produces a desired 3-D force at a point, called the *operational point*, selected in the last link of the device. We map the operational point to a point P in a virtual workspace W. By extension, we also call P the operational point. For simplification, but with no loss of generality, we see W as a rectangular parallelepiped (Figure 1). If this is not the case, we enclose the actual workspace by a parallelepiped W and some regions of W are simply not reachable by P. The axes of the workspace coordinate system are aligned with the edges of W.

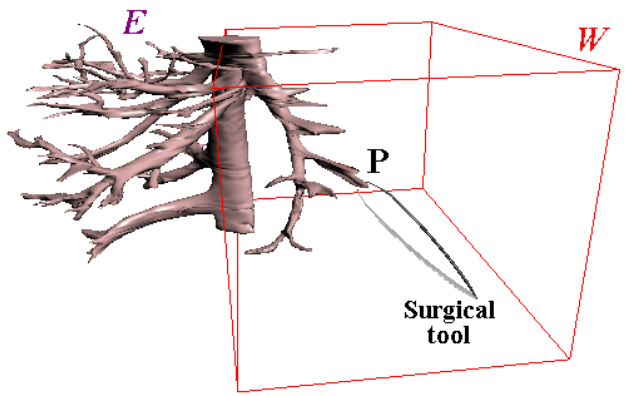

Figure 1: Virtual deformable vena cava in the workspace W of the operational point P

W contains a virtual elastic object E. A mechanical model of E is given to a simulator that computes the object's deformations when external forces are applied to points in its surface. The simulator tends to bring the object to equilibrium at each computation step, according to the deformations that the object undergoes. Consider the situation where P is in contact with the surface of E.

We assume this contact to be sticky, so that any further motion of P yields a deformation of E. The simulator computes this deformation and renders it on a graphic display in real-time. Our goal is to also render the reaction force exerted by E on P on the haptic device.

In practice, we attach a real surgical instrument to the last link of the haptic device. The user moves this instrument, which causes a model I of the instrument to move accordingly in the virtual workspace. An efficient collision-detection technique (see Section 6) detects the occurrence of a contact between I and E. The point of contact on I is then chosen to be the operational point P.

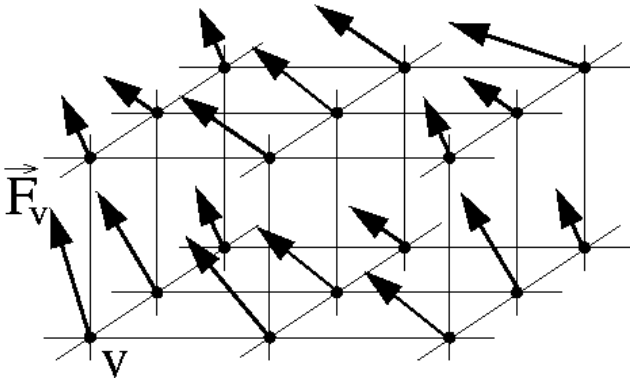

Figure 2: A portion of a forcegrid

We assume that the simulator computes the deformation of E at roughly 30Hz, and provides the value of the reaction force $\boldsymbol{F}$ exerted by E at P. The forcegrid allows an approximation of the reaction force to be rendered on the operational point at rates on the order of 500Hz, or greater, independent of the complexity of the model of E. It is essentially a data-caching structure which combines previous force computation results. The haptic device controller exploits this data structure concurrently to render forces at the desired rate. More precisely, the forcegrid is defined as a regular grid G of vertices placed over W, in which every vertex v contains a force $\boldsymbol{F}_v$. See Figure 2. In the ideal case, $\boldsymbol{F}_v$ is the reaction force that would be exerted by E on P if P was located at v. But it is not practical to fill G with forces prior to interacting with E. Indeed, the contact point is not known in advance, and filling G when the contact is detected would cause a prohibitive delay. Therefore, we propose in Section 5 a method that computes and updates approximations of the ideal forces while the haptic device is interacting with E. Whenever the controller of the haptic device needs to render the reaction force at P (~500Hz), it localizes P in the grid and interpolates among the forces labeling neighboring vertices. We now describe the interpolation function.

4. Interpolation in a forcegrid

Let $W = [0,X]\times[0,Y]\times[0,Z]$. Let $(p_x+1)\times(p_y+1)\times(p_z+1)$ be the size (number of vertices) of G. We define:

$$\delta_x = X/p_x,\ \delta_y = Y/p_y, \text{ and } \delta_x = Z/p_z.$$

Each *cell* of G is a parallelepiped whose edges have lengths δ_x, δ_y, and δ_z.

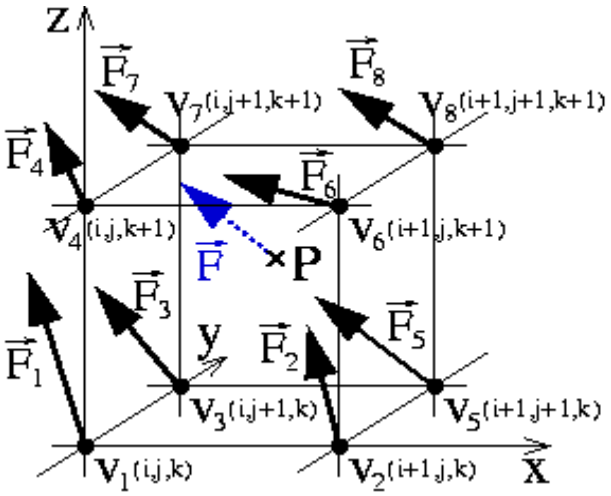

Figure 3: A cell of the forcegrid and its vertex forces

The real coordinates of a vertex v of G are of the form $(i\delta_x, j\delta_y, k\delta_z)$, with $(i,j,k) \in [0,p_x]\text{x}[0,p_y]\text{x}[0,p_z]$. So, we identify each vertex v by its integer coordinates (i,j,k), and each cell c of G by its vertex (i,j,k) with the smallest indices ($i < p_x$-1, $j < p_y$-1, and $k < p_z$-1). Hence, the eight vertices of c are obtained by forming all combinations of indices in $\{i,i+1\}$, $\{j,j+1\}$, and $\{k,k+1\}$ (Figure 3).

Given the position (x,y,z) of the operational point P in W, the cell c that contains P is obtained by computing:

$$i = \lfloor x/\delta_x \rfloor, \quad j = \lfloor y/\delta_y \rfloor, \quad \text{and} \quad k = \lfloor z/\delta_z \rfloor.$$

To simplify the calculations, we normalize c to be the parallelepiped $[0,1]\times[0,1]\times[0,1]$, and we redefine (x,y,z) to stand relative to a coordinate system whose origin is vertex (i,j,k). Hence, $(x,y,z) \in [0,1]\times[0,1]\times[0,1]$. Let $\boldsymbol{F}_1$, $\boldsymbol{F}_2$, …, $\boldsymbol{F}_8$ denote the forces in the eight vertices of c, as depicted in Figure 3. We express the reaction force $\boldsymbol{F}$ at P as the following combination:

$$\boldsymbol{F} = \boldsymbol{a} + x\boldsymbol{b} + y\boldsymbol{c} + z\boldsymbol{d} + xy\boldsymbol{e} + xz\boldsymbol{f} + yz\boldsymbol{g} + xyz\boldsymbol{h},$$

where $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, $\boldsymbol{d}$, $\boldsymbol{e}$, $\boldsymbol{f}$, $\boldsymbol{g}$, and $\boldsymbol{h}$ are forces computed from the forces at the eight vertices:

- At (0,0,0), $\boldsymbol{F} = \boldsymbol{F}_1 = \boldsymbol{a}$
- At (1,0,0), $\boldsymbol{F} = \boldsymbol{F}_2 = \boldsymbol{a} + \boldsymbol{b}$
- At (0,1,0), $\boldsymbol{F} = \boldsymbol{F}_3 = \boldsymbol{a} + \boldsymbol{c}$
- At (0,0,1), $\boldsymbol{F} = \boldsymbol{F}_4 = \boldsymbol{a} + \boldsymbol{d}$
- At (1,1,0), $\boldsymbol{F} = \boldsymbol{F}_5 = \boldsymbol{a} + \boldsymbol{b} + \boldsymbol{c} + \boldsymbol{e}$
- At (1,0,1), $\boldsymbol{F} = \boldsymbol{F}_6 = \boldsymbol{a} + \boldsymbol{b} + \boldsymbol{d} + \boldsymbol{f}$

- At (0,1,1), $\boldsymbol{F} = \boldsymbol{F}_7 = \boldsymbol{a} + \boldsymbol{c} + \boldsymbol{d} + \boldsymbol{g}$
- At (1,1,1), $\boldsymbol{F} = \boldsymbol{F}_8 = \boldsymbol{a} + \boldsymbol{b} + \boldsymbol{c} + \boldsymbol{d} + \boldsymbol{e} + \boldsymbol{f} + \boldsymbol{g} + \boldsymbol{h}$

These eight equations can be solved for $\boldsymbol{a}$, $\boldsymbol{b}$, …, $\boldsymbol{h}$, which yields the *forcegrid interpolation function*:

$$\begin{aligned}\boldsymbol{F} = {} & \boldsymbol{F}_1 + x(\boldsymbol{F}_2-\boldsymbol{F}_1) + y(\boldsymbol{F}_3-\boldsymbol{F}_1) + z(\boldsymbol{F}_4-\boldsymbol{F}_1) \\ & + xy(\boldsymbol{F}_5-\boldsymbol{F}_3-\boldsymbol{F}_2+\boldsymbol{F}_1) + xz(\boldsymbol{F}_6-\boldsymbol{F}_4-\boldsymbol{F}_2+\boldsymbol{F}_1) \\ & + yz(\boldsymbol{F}_7-\boldsymbol{F}_4-\boldsymbol{F}_3+\boldsymbol{F}_1) \\ & + xyz(\boldsymbol{F}_8-\boldsymbol{F}_7-\boldsymbol{F}_6-\boldsymbol{F}_5+\boldsymbol{F}_4+\boldsymbol{F}_3+\boldsymbol{F}_2-\boldsymbol{F}_1)\end{aligned}$$

Even when the forces $\boldsymbol{F}_1$ through $\boldsymbol{F}_8$ are exactly the reaction forces exerted by E when the operational point is respectively located at each of the eight vertices of c, the above function is still a linear approximation of the actual value of $\boldsymbol{F}$ at other locations of P in c. However, in most situations, the actual force $\boldsymbol{F}$ varies smoothly when P moves, so that the above approximation is satisfactory. For instance, in a mass-spring model, forces at the mesh vertices are differentiable functions of the positions of the vertices. The accuracy can be improved by increasing the resolution of G.

The forcegrid interpolation function defines a continuous force field over W. Indeed, one can verify that, if P is located in a face bounding two cells or in an edge shared by four cells, then the above formula gives the same value of $\boldsymbol{F}$, independent of the cell used. Although the first derivative (gradient) of $\boldsymbol{F}$ is not continuous at the cell boundaries, we have been unable to feel such discontinuities on the haptic device. If needed, one could use higher-degree interpolation functions.

5. Filling a Forcegrid

We now propose two techniques to fill the vertices of a forcegrid G with forces. Both techniques work online, while the haptic device is interacting with an object E. The first technique – called the *silent method* – is the simplest, and only uses the data computed (at ~30Hz) by the graphic simulator. The second technique – called the *request method* – requires a second copy of the simulator to be available on a separate processor. Only the silent method has been implemented and tested so far.

Silent method – The silent method tries to make the best possible use of the limited flow of information provided by the graphic simulator.

The simulator computes at 30Hz the new shape of E along with the reaction force $\boldsymbol{F}$ at the current location of P. At any one time during haptic interaction, each vertex v of G contains a force $\boldsymbol{F}_v$ with a certain weight w_v. Initially (when P makes contact with E), every vertex of G contains the null force with weight 0. Next, every new force $\boldsymbol{F}$ computed by the simulator is used to update the forces and weights stored in G. We use a "candle light" technique in which the current location of P is seen as a punctual source of light illuminating the forcegrid with an intensity that decreases with distance. In other words, the influence of $\boldsymbol{F}$ is stronger on the vertices of G that are close to P than on those that are further away.

More precisely, let v be a vertex of G at Euclidean distance d of P. The force and weight contained in v are updated according to the *forcegrid update formulae*:

$$\begin{aligned}&\boldsymbol{F}_v \leftarrow [w_v\,\boldsymbol{F}_v + w(d)\boldsymbol{F}]/(\,w_v + w(d)), \\ &w_v \leftarrow w_v + w(d),\end{aligned}$$

where $w(d)$ is a decreasing function that goes from 1 to 0 when d grows from 0 to a given constant r, called the *distance of influence* of a reaction force. There are two motivations for bounding the influence of a reaction force: (1) the value of $\boldsymbol{F}$ is likely to become irrelevant at positions that are too far from the current P; (2) if G is dense, setting r small enough significantly reduces the cost of updating the forces in G. Many different weighting functions could be used. We use the following one:

$$\begin{aligned}w(d) &= 1/(1 + d^2) - (d/(r(1 + r^2)) && \text{if } d \le r, \\ &= 0 && \text{if } d > r,\end{aligned}$$

Our experiments showed that r should be set to approximately 2-3 times the distance that the operational point may traverse between two iterations of the graphic simulator and that the resolution of G should be such that the distance between two adjacent vertices be on the order of $r/10$ to $r/5$.

Note that if P undergoes small motions relative to the resolution of G during some extended period of time, the graphic simulator will compute reaction forces at many points forming a dense cluster in W. Despite the weighting mechanism described above, these forces could have, altogether, an abnormally high impact on the vertex forces within radius r of the cluster's center. Therefore, we set a threshold s on the number of reaction forces computed in a certain cell that can be used to update the forcegrid.

Request method – The request method explicitly asks the simulator to compute forces at specific locations of the operational point, called *query positions (QP)*. In most implementations, since the processor running the graphic simulator is already busy, a separate processor running a copy of the simulator (without graphic visualization) is needed to process the queries of the request method.

For each QP, the reaction force $\boldsymbol{F}$ must be computed. $\boldsymbol{F}$ is then used to update the vertex and their weights in the forcegrid, using the update formulae of the silent method.

The request method should select the successive QPs in order to best fill the forcegrid around the current P, and thus improve the accuracy of the forces rendered on the haptic device. One way of doing this is to consider all the cells whose centers are within some distance ρ of the

current P. Typically, ρ should be chosen roughly equal to the distance of influence r. The request method selects the center of the cell that has received the fewest reaction forces so far as the next QP. If there are several such cells, the closest center to the current P is chosen. Obviously, several variant schemes would achieve the same purpose.

If implemented, the request method would typically be run in complement to the silent method. Then its queries need not be answered at any particular rate, or even at a constant one.

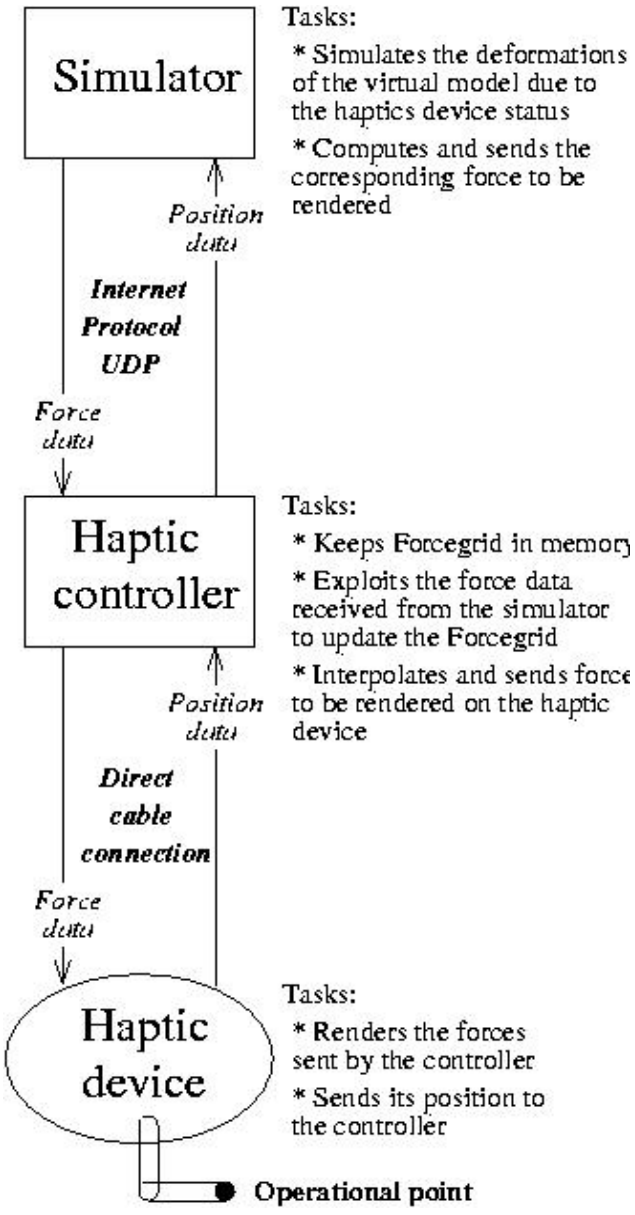

Figure 4: Architecture of the system implementing the forcegrid (silent method)

6. Implementation and Experiments

System – We have implemented the forcegrid with the silent method into a system whose architecture is shown in Figure 4. The haptic device is the PHANTOM Desktop of SensAble Technologies. This device must have extremely low friction and inertia to provide satisfactory interaction with an elastic object. Not all commercially available devices satisfy this requirement.

The forcegrid software (written in C++) runs on the 500MHz Intel Celeron processor of the controller of the PHANTOM. We set the update rate of the PHANTOM to 500Hz. Therefore, the forcegrid calculations must take no more than 2ms per haptic cycle.

The graphic simulator runs on a Sun Microsystems Ultra 60 workstation, and uses the mass-spring technique described in [6, 7] with either linear or non-linear springs. However, the forcegrid software is independent of the modeling and computational techniques used by the simulator.

The interaction protocol between the simulator and the forcegrid is the Internet protocol UDP, which does not require the forcegrid to acknowledge received data, nor the simulator to wait until acknowledgement has been received. The haptic device controller and the simulator can be connected by a simple serial line, a LAN, or through the Internet.

Initially, the forcegrid only contains null forces and the user moves the surgical instrument near deformable objects. A collision checker detects the occurrence of the contact between the instrument and any object [6] and returns the contact point. At this instant, the reaction force is null or very small. It takes 1/30 second for the simulator to return its first reaction force. During that short period of time the force rendered on the haptic device is the null force. As long as the haptic device is moved at reasonable velocity, this small delay is imperceptible to the user, since the deformation of the object and the actual reaction forces are small anyway. Once the simulator has returned a force, the forcegrid is filled with non-zero forces.

We experimented with various resolutions of the forcegrid and several values of the distance of influence r. With the PHANTOM Desktop, an adequate sampling of W was achieved with a grid of size $50\times50\times50$ with $r = 1/6$ of the size of W.

Examples – As usual, it is difficult to report on interactive haptic experiments in a paper. Since our mass-spring software allows us to easily describe new models of deformable objects, we have experimented with various models. Subjective haptic fidelity, as assessed by people internal and external to our group, was reported to be excellent. Forces were rendered smoothly and realistically, with no perceptible discontinuities or

tremors. In particular, we tested the forcegrid with a deformable model of a stomach. Figure 5 shows the stomach at rest (a) and a deformation generated by the soft-tissue deformation software (b). The underlying model of the stomach is a mesh of 4000 vertices (point masses) connected by 11,000 edges (springs). Figure 6 shows the stomach in a surgical training environment.

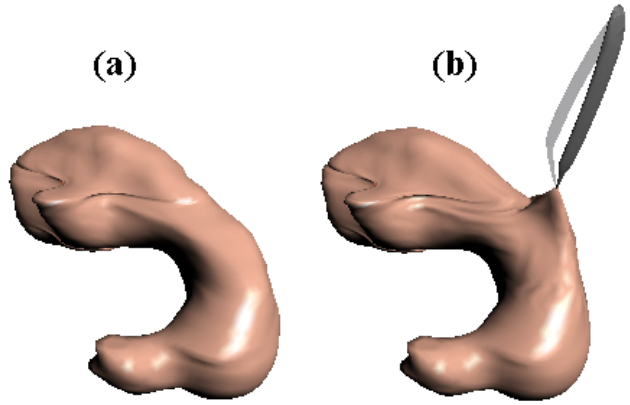

Figure 5: Stomach at rest (a) and deformed stomach (b)

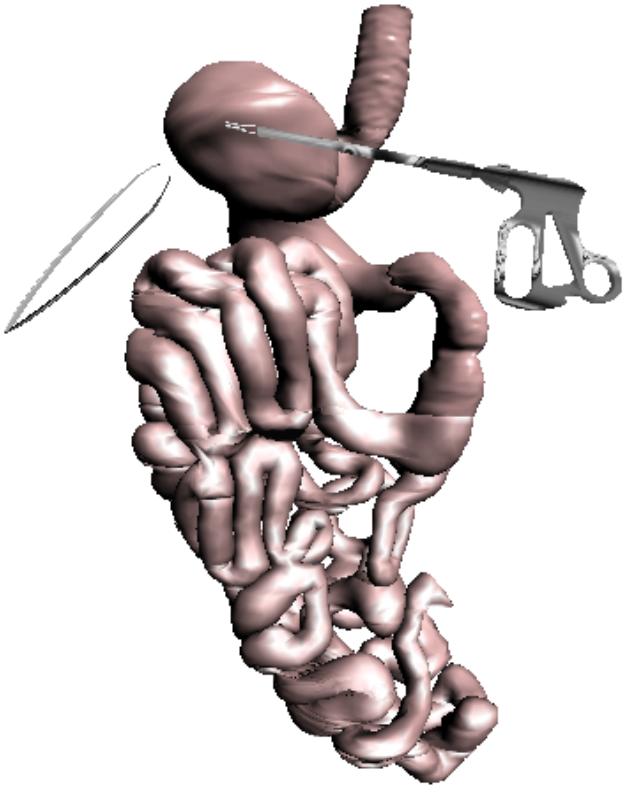

Figure 6: Stomach in a surgical training environment with other organs and surgical tools.

In another series of experiments, we input three different models having the same geometry at rest, but distinct mechanical parameters. In one model spring stiffness was uniformly set higher than in another model, while in the third model, stiffness varied greatly across the mass-spring mesh. Users were consistently able to correctly distinguish among these three models with haptic interaction and no visual feedback. In contrast, they were unable to do so with only visual feedback.

Quantitative evaluation – We measured the running times of the modules implementing forcegrid interpolation and filling. Forcegrid interpolation, including the localization of P in the grid and the evaluation of the interpolation function, takes 5µs. Forcegrid filling takes time linear in the number of grid vertices within the influence distance r from P. Updating the force of one vertex – including all the necessary computations – takes 1µs. In a 50×50×50 grid with r set to 1/6 of the size of the workspace, forces must be updated at roughly 3,000 vertices. So, a complete forcegrid filling operation takes 3,000 x 1µs = 3ms. To keep a high haptic rate, we split these 3,000 updates over consecutive force-rendering cycles. To achieve a haptic refresh rate of 500Hz, we can update up to 2,000 vertices per haptic cycle (2ms). On the other hand, a 30Hz simulator sends a new reaction force to the haptic controller every 33ms. So, splitting the force updates in the forcegrid by groups of 2,000 vertices makes it possible to update up to 33,000 vertices in one simulation cycle, which is an order of magnitude more than what we need in practice.

We also performed tests to estimate the errors in the rendered forces by comparing the forces rendered through different techniques. The model of the deformable object used in this experiment contains 10,000 vertices connected by 35,000 edges. We grabbed the surface of this object with an instrument, and then generated a sinusoidal movement of the instrument in a direction Δ orthogonal to the surface, with constant amplitude, but increasing speed. The simulator was running at 30Hz. We recorded simultaneously the position of the tool and the component of the reaction forces along Δ. Figure 7 shows the trajectory of the operational point (curve T) and three other curves representing the reaction forces obtained by three techniques. The instant when the contact is detected with the deformable object is $t = 0$. The first curve plots *streaming forces* $\boldsymbol{F}_s$, i.e. forces that the simulator computes and sends according to the position of the operational point it receives from the controller. The second curve represents *forcegrid forces* $\boldsymbol{F}_g$, forces rendered by our forcegrid software. The last curve represents *off-line forces* $\boldsymbol{F}_o$, the actual forces that would be computed by the simulator at the haptic rate if it was

fast enough (by necessity, these forces were computed off-line). F_s is actually the force that the controller receives in the forcegrid implementation, and with which it produces F_g. The curve representing F_s shows delay (due to the computation time of the simulator and the network delay) and jerkiness (due to the 30Hz simulation rate); it is subject to the inaccuracy of the simulator when the sinusoidal movement becomes too fast. After a short initial adaptation period, F_g eliminates all these problems. The curves representing F_g and F_o become very similar, regardless of the velocity of the operational point. Figure 8 displays the errors in F_g and F_s with respect to F_o. The error for F_g decreases over time, while the error for F_s increases as the operational point moves faster.

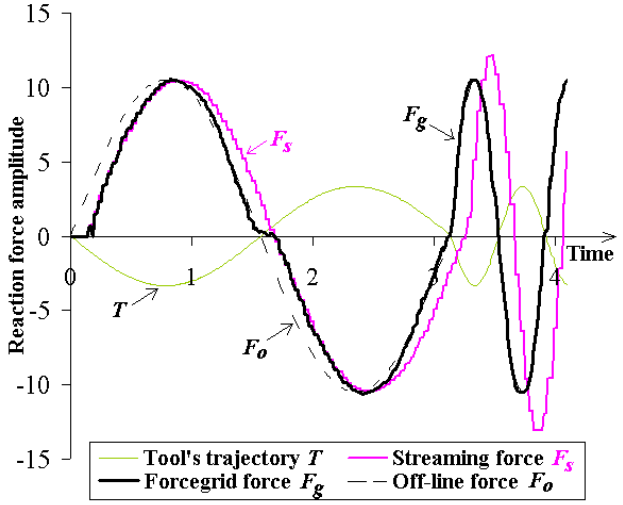

Figure 7: Trajectory of the operational point and forces rendered with different techniques

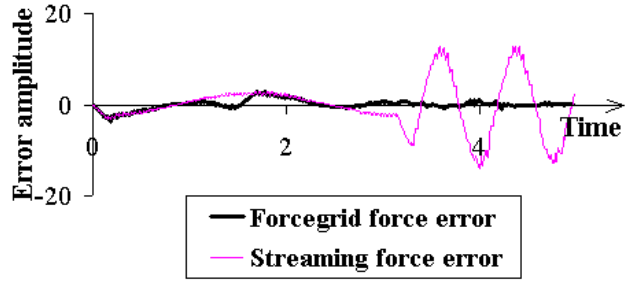

Figure 8: Errors of F_g and F_s with respect to F_o

7. Limitations and Remedies

The forcegrid presented in this paper offers a number of advantages over previous haptic rendering techniques. It is independent of the modeling and computational techniques used to simulate the deformations of a virtual elastic object. It is also independent of the cycle time of the simulator and the specific parameters of the physical model of the elastic object (mass, stiffness, damping, incompressibility, etc.). It is particularly easy to implement and the total compiled code is small (40Kbytes for the code and 1Mbytes for the grid), so that it can run on the local processor of any haptic device. It provides stable haptic behavior for a wide range of values of the forcegrid parameters. Experiments with many modeled objects have shown that it yields satisfactory haptic interaction.

However, in its current form, the forcegrid has some obvious limitations. A blatant one is that the haptic device must have three degrees of freedom and that there can be only one instrument (hence, a single operational point) interacting with the deformable object. If the haptic device had six degrees of freedom (e.g., to render both forces and torques), then the forcegrid would have to be a 6-D grid. The interpolation function in a 6-D grid is easily obtained using a recursive linear interpolation approach, but the size of a 6-D grid could become prohibitive. To overcome memory limitations, we can use an adaptive technique that maintains a high-density grid in a cube around the current position of the operational point and a low-density grid in the rest of the workspace. A 6-dof haptic device would be particularly useful to render contacts other than point contacts, such as multiple contacts and contacts over small surface areas. Similarly, if there were several haptic devices simultaneously interacting with the same object (each carrying a different surgical instrument), we could encode the positions of the various operational points in a single forcegrid – hence, if there are two haptic devices with three degrees of freedom each, the forcegrid would be a 6-D grid. Multiple haptic devices are needed to simulate two-hand or multi-surgeon surgery.

The forcegrid assumes that the behavior of the deformable object is smooth. In practice, however, discontinuities may occur. The occurrence of the contact between the surgical instrument and the deformable object is already one such discontinuity. Other discontinuities occur if, while deforming, the object makes contact with other objects (deformable or not) and with itself (self-collisions). As long as discontinuity events are sparse over time, it is possible to switch to a new forcegrid at each event. Some discontinuities, like the occurrence of new contacts, can be quickly detected by hierarchical [6] or feature-tracking techniques [18], or a combination of both [17]. When discontinuities occur over an extended period of time (e.g., while an instrument slides on the surface of the deformable object or cuts through it [10]), the forcegrid is no longer an appropriate tool. Indeed, previously computed forces are no longer valid, hence caching them is useless. A similar situation occurs if the elastic behavior of the deformable object has some hysteresis (but most existing simulators are unable to cope

with hysteresis anyway). One way to deal with this issue would be to "forget" old forces and update the forcegrid accordingly. However, the forcegrid would lose some of its appeal and its updating would be more costly. In those cases, the buffer model of [2] or the simpler interpolation method of [20] might be better choices.

Acknowledgements: This research was performed in the NASA-Stanford National Biocomputation Center at Stanford University. It was funded in part by grants from the National Science Foundation (IIS-9907060), the NIH National Libraries of Medicine (NLM-3506), and the National Aeronautics and Space Administration (NAS-NCC2-1010). This work benefited from discussions with Remis Balaniuk, Joel Brown, Cynthia Bruyns, Benjamin Lerman and Arnaud Tellier.

References

[1] N. Ayache, S. Cotin, and H. Delingette. Surgery Simulation with Visual and Haptic Feedback. In *Robotics Research*, Springer, 1998, 311-316.

[2] R. Balaniuk. Using Fast Local Modeling to Buffer Haptic Data. *Proc. 4th PHANToM Users Group Workshop (PUG'99)*, Boston, MA, Oct. 1999.

[3] J. Berkley, S. Weghorst, H. Gladstone, G. Raugi, D. Berg, and M. Ganter. Fast Finite Element Modeling for Surgical Simulation. *Proc. Medicine Meets Virtual Reality (MMVR'99)*, ISO Press, 1999, 55-61.

[4] C. Bosdogan, C. Ho, M.A. Srinivasan, S.D. Small, and S.L. Dawson. Force Interaction in Laparoscopic Simulation: Haptics Rendering of Soft Tissues. *Proc. Medicine Meets Virtual reality (MMVR'98)*, Jan. 1998, 28-31.

[5] M. Bro-Nielsen and S. Cotin. Real-Time Volumetric Deformable Models for Surgery Simulation Using Finite Elements and Condensation. *Proc. Eurographics'96*, 1996, 57-66.

[6] J. Brown, K. Montgomery, J.C. Latombe, M. Stephanides. A Microsurgery Simulation System. *Proc. Int. Conf. Medical Image Computing and Computer-Assisted Intervention*, Utrecht, The Netherlands, Oct. 2001.

[7] J. Brown, S. Sorkin, C. Bruyns , J.C. Latombe, K. Montgomery, and M. Stephanides. Real-Time Simulation of Deformable Objects: Tools and Application. *Proc. IEEE Conf. on Computer Animation*, Seoul, Korea, Nov. 2001.

[8] G. Burdea, G. Patounakis, V. Popescu, and R.E. Weiss. Virtual Reality Training for the Diagnosis of Prostate Cancer. *Proc. IEEE Symp. Virtual reality and Applications (VRAIS'98)*, Atlanta, GA, March 1998, 190-197.

[9] S. Cotin, H. Delingette, and N. Ayache. Real-Time Elastic Deformations of Soft Tissues for Surgery Simulation. *IEEE Tr. Visualization and Computer Graphics* 5(1),:62-73, 1999.

[10] S. Cotin, H. Delingette, 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.

[11] G. Debunne, M. Desbrun, M.P. Cani and A.H. Barr. Dynamic Real-Time Deformations using Space and Time Adaptive Sampling. Proc SIGGRAPH'01, 2001.

[12] H. Delingette. *Towards Realistic Soft Tissue Modeling in Medical Simulation*. Tech. Rep. No. 3506, INRIA, Sophia-Antipolis, France, Sept. 1998.

[13] D.H. House and D.E. Breen (eds.). *Cloth Modeling and Animation*. A K Peters, Natick, MA, 2000.

[14] D. James and D. Pai. A Unified Treatment of Elastostatic Contact Simulation for Real Time Haptics, *The Electronic J. of Haptics Research*, 2(1), Sept. 2001.

[15] A. Joukhadar and C. Laugier. Dynamic Simulation: Model, Basic Algorithms, and Optimization. In *Algorithms for Robotic Motion and Manipulation*, AK Peters, Natick, MA, 1997, 419-434.

[16] B. Jackson and L. Rosenberg. Force Feedback and Medical Simulation. In *Interactive Technology and the New Paradigm for Health-Care*, IOS Press, 1995, 147-151.

[17] M.C. Lin. Fast and Accurate Collision Detection for Virtual Environments. *Proc. IEEE Scientific Visual. Conf*., 1999.

[18] M. Lin and J. Canny. Efficient Algorithms for Incremental Distance Computation. *Proc. IEEE Int. Conf. Robotics and Automation*, 1991, 1008-1014.

[19] B. Marcus. Hands On: Haptic Feedback in Surgical Simulation. *Proc. of Medicine Meets Virtual Reality IV (MMVR'96)*, San Diego, CA, 1996, 134-139.

[20] G. Picinbono, J.C.Lombardo, H. Delingette, and N. Ayache. Anisotropy, Interaction and Extrapolation for Surgery Simulation. *J. Visualization. and Computer Animation*, 2001.

[21] M. Teschner, S. Girod and B. Girod, Efficient and Robust Soft Tissue Prediction in Craniofacial Surgery Simulation Using Individual Patient's Data Sets. *Proc. Int. Conf. Computer Assisted Radiology and Surgery (CARS'99)*, Paris, June 1999, 635-639.