Higher Precision in Volume Haptics Through Subdivision of Proxy Movements

Karljohan Lundin Palmerius George Baravdish

Visual Information Technology and Applications
Linköping University, Sweden
{karlu,geoba}@itn.liu.se

Abstract. Volume haptics has become an increasingly popular way of adding guidance and improving information bandwidth in scientific visualization. State-of-the-art methods, however, use linear equations, which allows for a precision that can be insufficient in some circumstances. This paper describes how step-length subdivision can be used to improve precision even though these methods do not use integration steps in its usual meaning.

1 Introduction

Haptic feedback from volumetric data, so called volume haptics, has become an increasingly popular way of adding guidance and improving information bandwidth in scientific visualization. State-of-the-art algorithms are capable of representing features in the data as shapes with high stability, while at the same time avoiding haptic occlusion or obstruction by letting the shapes yield to data specific forces. This is, today, done using linear equations. By locking the rate of the haptic loop, typically to 1 kHz, or simply not delivering the probe position more often than that, haptic systems limit the precision of the linear approximation. The approximation error is accumulated over time potentially causing artifacts in the haptic feedback, such as fall-through of surfaces in the data.

This paper describes how step subdivision can be performed within the time budget for each haptic loop, to shorten the step length and thereby improve the precision of any algorithm for constraint-based volume haptics. This approach does not require a derivative or a higher update rate for the haptic loop or the probe position, making it suitable for incorporation in readily available systems.

2 Related Work

Haptics has successfully been applied in scientific visualization to enhance speed in specific tasks and information bandwidth[1–7]. State-of-the-art methods for volume haptics in scientific visualization apply a *constraint-based* approach[4–7], thereby avoiding the stability issues associated with force functions while representing the data by intuitive shapes. By letting the constraint yield to a material specific force, obstruction or occlusion is avoided removing the need for the user to explicitly select the region to probe. The concept has also been developed into an approach based on *haptic primitives*[8]: plane, line and point primitives providing yielding constraints, and a force primitive providing integrated force function feedback.

To enable the yielding effect, these methods use linear equations, which in some cases does not provide sufficient precision in the haptic interaction. Ikits et al.[5] use higher order integration to improve the precision, but are thereby forced to remove the support for yielding constraints. A paper by Lundin et al.[9] describes a method for improving the precision while retaining the yielding effect. The method, however, only improves the numerical precision within the limits of linear approximation.

3 Proxy-based Volume Haptics

The constraint-based methods apply a decoupling scheme where the probe, $\boldsymbol{x}_\mathrm{p}$, of the haptic instrument is internally represented by a proxy point, $\tilde{\boldsymbol{x}}_\mathrm{p}$, describing the point of interaction[10, 11]. This *proxy-based* approach follows three steps: 1) extract the volumetric property of interest at the proxy position, 2) move the proxy point to a new position, $\tilde{\boldsymbol{x}}'_\mathrm{p}$, according to these data, and 3) calculate the feedback from the probe's displacement relative the proxy,

$$\mathit{\Lambda} = \boldsymbol{\Lambda}(\tilde{\boldsymbol{x}}_\mathrm{p}) \tag{1}$$

$$\tilde{\boldsymbol{x}}'_\mathrm{p} = \mathcal{F}(\mathit{\Lambda},\, \tilde{\boldsymbol{x}}_\mathrm{p},\, \boldsymbol{x}_\mathrm{p}) \tag{2}$$

$$\boldsymbol{f}_\text{feedback} = -k_s\left(\boldsymbol{x}_\mathrm{p} - \tilde{\boldsymbol{x}}'_\mathrm{p}\right) - k_d\left(\dot{\boldsymbol{x}}_\mathrm{p} - \dot{\tilde{\boldsymbol{x}}}'_\mathrm{p}\right) \tag{3}$$

where k_s and k_d are the stiffness and damping of the virtual coupling. These steps are common for the constraint-based methods and only $\mathcal{F}$ in (2) differs.

For the yielding constraints approach the local data, $\mathit{\Lambda}$, are used to define an orthogonal frame of two or three unit vectors, $\boldsymbol{u}_i$, representing the directions for the constraints. The proxy moving function, $\mathcal{F}$, is then defined as

$$\mathcal{F}(\mathit{\Lambda},\, \tilde{\boldsymbol{x}}_\mathrm{p},\, \boldsymbol{x}_\mathrm{p}) = \tilde{\boldsymbol{x}}_\mathrm{p} + \sum_i \boldsymbol{u}_i(\mathit{\Lambda}) \min\left(0,\, \boldsymbol{u}_i(\mathit{\Lambda}) \cdot (\boldsymbol{x}_\mathrm{p} - \tilde{\boldsymbol{x}}_\mathrm{p}) - s_i/k_s\right) \tag{4}$$

where s_i is the strength of the ith constraint. In the haptic primitives approach a haptic mode is defined by selecting haptic primitives and configuring them according to the the data. $\mathcal{F}$ then finds the proxy position by minimizing the difference between the force feedback and the primitives' force fields,

$$\mathcal{F}(\mathit{\Lambda},\, \tilde{\boldsymbol{x}}_\mathrm{p},\, \boldsymbol{x}_\mathrm{p}) = \underset{\tilde{\boldsymbol{x}}_\mathrm{p} \in \mathbb{R}^3}{\operatorname{argmin}} \left| \sum_i \boldsymbol{\mathcal{P}}_i(\mathit{\Lambda},\, \tilde{\boldsymbol{x}}_\mathrm{p}) - k_s\left(\tilde{\boldsymbol{x}}_\mathrm{p} - \boldsymbol{x}_\mathrm{p}\right) \right| \tag{5}$$

where $\boldsymbol{\mathcal{P}}_i$ is the force field of the ith primitive, as described in [8].

Common for these methods is that neither defines a derivative or an integration step that can be improved through, for example, Runge-Kutta integration.

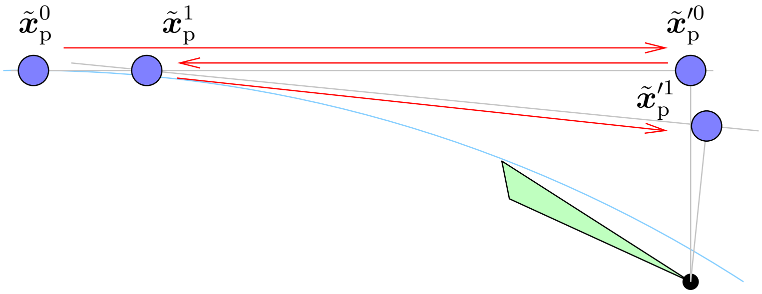


Fig. 1. Proxy motion subdivision. The old proxy position, $\tilde{\boldsymbol{x}}_{\mathrm{p}}^{0}$, and data at that position are first used to estimate the new proxy position, $\tilde{\boldsymbol{x}}_{\mathrm{p}}^{\prime 0}$. The proxy is the moved back to $\tilde{\boldsymbol{x}}_{\mathrm{p}}^{1}$, to simulate a shorter "step length", and the same procedure is performed again.

4 Proxy Movements Subdivision

With a modern computer and the fast algorithm for the haptic primitives-based method, the proxy update takes about 11 μs[9]. If the haptic loop is locked to 1 kHz, this leaves a dedicated CPU doing nothing (NOOPs) more than 98% of the time. A higher precision in the haptic interaction can be achieved by using this CPU time to perform more but smaller steps. With optimal use the remaining time should allow for up to $1/\left(1-98\%\right)=50$ additional steps.

4.1 Subdividing Proxy Motion

From the proxy moving functions in (4) and (5) can be concluded that these methods do not use step direction or step length, nothing that can be apparently shortened to implement a shorter step length. Instead, we need to simulate the integration steps. This is done by first applying the proxy moving function. The change in proxy position can be considered to constitute an integration step which can then be shortened, a posteriori.

Let t_{Δ} be the available time budget for estimating the haptic feedback and t_n be the time at which the nth step is evaluated. To perform a step the local data are first analyzed and the current proxy point, $\tilde{\boldsymbol{x}}_{\mathrm{p}}^{n}$, is moved to a new position, $\tilde{\boldsymbol{x}}_{\mathrm{p}}^{\prime n}$, through a call to the function $\mathcal{F}$,

$$\Lambda = \boldsymbol{\Lambda}\left(\tilde{\boldsymbol{x}}_{\mathrm{p}}^{n}\right) \tag{6}$$

$$\tilde{\boldsymbol{x}}_{\mathrm{p}}^{\prime n} = \mathcal{F}(\Lambda, \tilde{\boldsymbol{x}}_{\mathrm{p}}^{n}, \boldsymbol{x}_{\mathrm{p}}) \tag{7}$$

The number of steps that there is still time left to calculate, N_{steps}, can be determined by considering how much of the time budget that is already spent and the time needed,

$$N_{\mathrm{steps}} = \left\lfloor \frac{t_{\Delta} - (t_n - t_0)}{\tilde{t}_{\delta}} \right\rfloor \tag{8}$$

where $\tilde{t}_\delta$ is the time it is estimated to take to calculate one integration step. If N_{steps} is at least one, the proxy is moved back towards the previous proxy position (see figure 1), providing a new proxy position, $\tilde{\boldsymbol{x}}_{\text{p}}^{n+1}$,

$$\tilde{\boldsymbol{x}}_{\text{p}}^{n+1} = \tilde{\boldsymbol{x}}_{\text{p}}^{n} + \frac{\tilde{\boldsymbol{x}}_{\text{p}}'^{n} - \tilde{\boldsymbol{x}}_{\text{p}}^{n}}{N_{\text{steps}} + 1} \tag{9}$$

for estimating a new proxy movement.

This procedure — (6), (7) and (9) — is iterated until N_{steps} is zero or less, meaning that the time budget is spent. At this point the last action is that defined by (7) giving an estimated proxy position, $\tilde{\boldsymbol{x}}_{\text{p}}'^{n}$. This is then used in (3) to estimate the final feedback. Since the intermediate proxy position from (9) is used to extract new data in (6) before the next step is performed by (7) this iterative process will follow the data much more accurately than taking just one single step.

4.2 Estimation of t_δ

The time, t_δ, needed to calculate one step is not known a priori but must be estimated, $\tilde{t}_\delta$. There are several methods available for providing an estimate from historical data, however the delay is noisy and has outliers caused by interrupts in the operating system which invalidates many of them. The estimation should also be computationally cheap to avoid unnecessary overhead.

For the work presented in this paper we use an approach where an estimate, $\tilde{t}_\delta$, is adjusted towards the last historical delay value, t_δ, throughout the simulation,

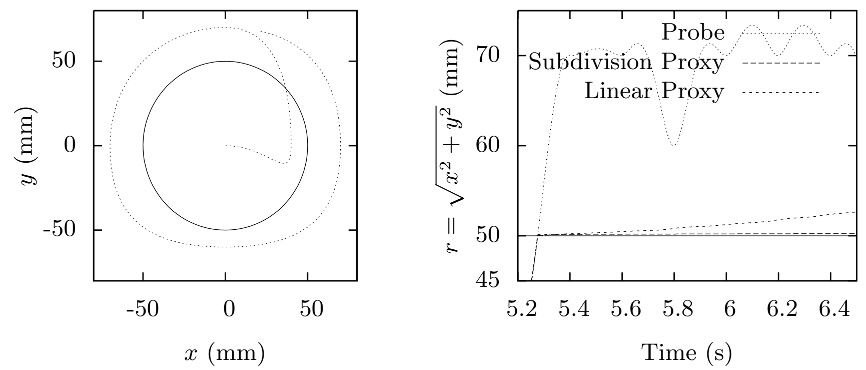
$$\tilde{t}'_\delta = \begin{cases} (1+\alpha)\,\tilde{t}_\delta, & \text{if } t_\delta > \tilde{t}_\delta \\ (1-\alpha)\,\tilde{t}_\delta, & \text{if } t_\delta \leq \tilde{t}_\delta \end{cases} \tag{10}$$

where α is a constant controlling the rate of change for the estimate. This estimation can be implemented very CPU efficiently and does not require saving historical data.

The value of α is set with a trade-off between a stable estimate and a rapid response to a long-term change in the time delay. With $\alpha \sim 0.05$ the estimate adjusts quickly while providing an accuracy within the normal variations of the delay.

5 Results

The subdivision algorithm has been integrated into the Volume Haptics Toolkit (VHTK) and H3D API, and tested on an analytical volumetric data set simulating a spherical cavity. A pre-defined spline controls the probe path with a velocity of 0.2 m/s. This relatively high speed is used to emphasize the numerical error. The probe starts inside the cavity, moves outwards and in a circle probing the inside with a surface simulating haptic mode, see figure 2(a). The simulation is running on an Ubuntu Linux machine with 1.83 GHz Dual Core CPU. H3D



(a) Probe path starting at the centre and probing the inside of the spherical cavity.

(b) The proxy distance from the centre over time with and without the movements subdivision.

Fig. 2. The probe and proxy motions in the simulation. The proxy should trace the spherical cavity at its radius of 50 mm. The solid line shows the border of the cavity in both graphs.

API has been patched to use a FIFO real-time scheduler for the haptic thread and run at a *nice level* of -20 to reduce the effect of interrupts in the execution.

We use $\alpha = 0.05$ in (10) and set the time budget, t_{Δ}, to 0.9 ms thereby leaving some time for synchronizing the data with the haptics device before reaching the 1 ms limit in a 1 kHz update rate. With these settings the presented method shortens the average step length from 349 μm to 8.7 μm. This results in a significantly improved precision, as can be seen in figure 2(b). The algorithm performs at an average 46 sub-steps for each estimation of the haptic feedback which with linear approximation results in an equivalent reduction in the error. This number is not far from the theoretical 50 mentioned in section 4. The deviation from this number is caused by some overhead introduced in the subdivision of the proxy movements. Observe also that the step length reduction is not equivalent to the number of sub-steps. This is because the increased precision in the current example also gives rise to a change in the path of the proxy motion.

Because of natural variations in t_{δ}, the number of steps to perform will sometimes be overestimated resulting in a time budget overrun. It is therefore important that system interrupts are kept at a minimum when a higher percentage of the available time is used to estimate the proxy movements. With the real-time scheduler set to high priority and the t_{Δ} set to 0.9 ms this is not a problem — budget overruns are no more frequent with subdivision than without. With a normal thread scheduler running at a nice level of zero, however, the interrupts causes the time budget to be overrun for almost 2% of the haptic frames.

The haptic loop can on some systems be released from the 1 kHz rate lock and be executed at the highest possible rate. In such a case the presented subdivision approach still shortens the step length to one third. This is attributed to the fact that performing the subdivision bypasses parts of the full haptics loop, e.g. synchronizing and sending intermediate data to the haptics device.

6 Conclusions

The method presented in this paper subdivides the motion of the proxy to utilize the available computational power to increase the precision. This paper has presented subdivision for proxy-based volume haptics and shown that it can improve the precision by an order of magnitude. A side effect can be the occasional time budget overrun if the haptics thread is not sufficiently prioritized by the system. Also when the haptic loop is not locked to the rate of 1 kHz the presented method enables a step length of one third of that performed when not using the subdivision.

References

1. Iwata, H., Noma, H.: Volume haptization. In: Proceedings of IEEE 1993 Symposium on Research Frontiers in Virtual Reality. (1993) 16–23
2. Avila, R.S., Sobierajski, L.M.: A haptic interaction method for volume visualization. In: Proceedings of IEEE Visualization. (1996) 197–204
3. Lawrence, D.A., Pao, L.Y., Lee, C.D., Novoselov, R.Y.: Synergistic visual/haptic rendering modes for scientific visualization. IEEE Computer Graphics and Applications **24**(6) (2004) 22–30
4. Lundin, K., Ynnerman, A., Gudmundsson, B.: Proxy-based haptic feedback from volumetric density data. In: Proceedings of the Eurohaptic Conference, University of Edinburgh, United Kingdom (2002) 104–109
5. Ikits, M., Brederson, J.D., Hansen, C.D., Johnson, C.R.: A constraint-based technique for haptic volume exploration. In: Proceedings of IEEE Visualization, pp. 263–269. (2003)
6. Vidholm, E., Tizon, X., Nyström, I., Bengtsson, E.: Haptic guided seeding of MRA images for semi-automatic segmentation. In: Proceedings of IEEE International Symposium on Biomedical Imaging. (2004)
7. Vidholm, E., Nyström, I.: A haptic interaction technique for volume images based on gradient diffusion. In: Proceedings of the IEEE World Haptics Conference, Pisa, Italy (2005) 336–341
8. Lundin, K., Gudmundsson, B., Ynnerman, A.: General proxy-based haptics for volume visualization. In: Proceedings of the IEEE World Haptics Conference, Pisa, Italy, IEEE (2005) 557–560
9. Palmerius, K.L.: Fast and high precision volume haptics. In: Proceedings of the IEEE World Haptics Conference, Tsukuba, Japan, IEEE (2007) 501–506
10. Zilles, C.B., Salisbury, J.K.: A constraint-based god-object method for haptic display. In: Proceedings of IEE/RSJ International Conference on Intelligent Robots and Systems, Human Robot Interaction, and Cooperative Robots. Volume 3. (1995) 146–151
11. Ruspini, D.C., Kolarov, K., Khatib, O.: The haptic display of complex graphical environments. Computer Graphics **31**(Annual Conference Series) (1997) 345–352