Haptic Rendering of Dynamic Volumetric Data

Karljohan Lundin Palmerius, Matthew Cooper and Anders Ynnerman

Lundin Palmerius, K., Cooper, M., Ynnerman, A., (2008), Haptic Rendering of Dynamic Volumetric Data, *IEEE Transactions on Visualization and Computer Graphics*, 14(2), 263-276.
https://doi.org/10.1109/TVCG.2007.70409

Haptic Rendering of Dynamic Volumetric Data

Karljohan Lundin Palmerius, Matthew Cooper and Anders Ynnerman

***Abstract*— With current methods for volume haptics in scientific visualization, features in time-varying data can freely move straight through the haptic probe without generating any haptic feedback — the algorithms are simply not designed to handle variation with time but consider only the instantaneous configuration when the haptic feedback is calculated. This article introduces haptic rendering of dynamic volumetric data to provide a means for haptic exploration of dynamic behaviour in volumetric data. We show how haptic feedback can be produced that is consistent with volumetric data moving within the virtual environment and with data that, in itself, evolves over time. Haptic interaction with time-varying data is demonstrated by allowing palpation of a CT sequence of a beating human heart.**

***Index Terms*— direct volume haptics, time-varying data, changing model transform, scientific visualization**

I. Introduction

Volume rendering has become an effective tool in scientific visualization for exploring static as well as time-varying volumetric data. Using sequences of data the user can explore dynamic phenomena and processes which can otherwise be difficult to understand, such as swirl propagation or shockwave folding in a CFD simulation, the valve movements in a CT scan of a beating human heart, or the changes in blood flow in a Doppler MRI.

With the overwhelming amount of data provided by time-varying volumes, it can be beneficial to augment the direct graphical representation with other representations delivered via alternative senses. Several independent research efforts[1]–[5] have shown that haptics has the potential to significantly increase both speed and accuracy of human-computer interaction and provide extra information about the data. This natural mode of interaction makes effective use of our fundamental sense of touch and allows for synergistic effects between multiple modes of exploration of the volume data[6]–[8].

Direct volume haptics[6]–[13], by analogy with direct volume rendering, provides a means to present the full data through force feedback without restricting the exploration to a subset of the data presented through an intermediate representation, such as an extracted surface. Previous methods for direct volume haptics, however, have difficulty in incorporating dynamic data in the haptic representation since they cannot capture changes in the data, which should affect the haptic probe. If, for example, the user is holding the haptic probe stationary in a volume of data through which a shock-front is moving and the shock-front lies on one side of the probe in one data sample, but in the next it lies on the other side of the probe then it should, obviously, have affected the probe while passing though it. While it is possible to see how a haptic model based on surface extraction would allow the tracking of the surface and the use of collision detection methods to detect this inter-frame interaction of the data with the probe, current schemes for direct volume haptics cannot capture this interaction. The absence of this effect, where a feature should push against the probe but does not, results in a counter-intuitive and potentially confusing error in the haptic feedback.

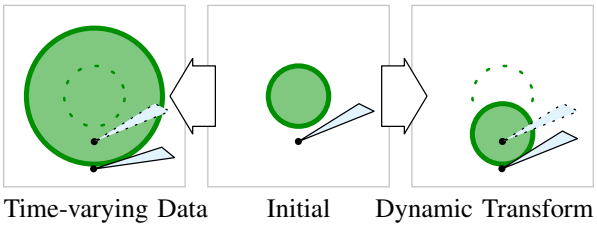


Fig. 1. Centre to right shows interaction with time-varying data while centre to left shows haptic interaction with data affected by a changing model transform. The solid probe shows the correct haptic response, obtained through the methods presented in this article, and the dotted probe shows the response without these methods.

This problem can also occur in interaction with time-invariant data which moves in the scene, for example if the user employs the common approach of using a separate interactor, such as a 3D mouse, to create a two-handed operation interface. Using the 3D mouse to change the model transform, and so control the movement of the data being explored, while the data are defining the force feedback can result in the effect where the user rotates or translates the data and a feature in the data lies on opposite sides of the (stationary) probe in two consecutive positions of the data. The feature should have transitioned across the probe and have affected it but, again, this interaction is not captured by current techniques for direct volume haptics.

The approach presented in this paper introduces two methods, each resolving one of the two problems described above, see figure 1. These can be combined to permit direct volume haptics to capture arbitrary changes in both the position of a time-invariant data volume and the position of features in a time-varying one. In this way this approach can capture all of the time-varying effects which can affect the data volume and hence the haptic interaction with the data. The method uses a combination of preprocessing and real-time techniques to ensure that the problem remains sufficiently computationally simple to sustain the required haptic frame rate and so full direct volume haptic interaction, reflecting the dynamic features of the data, can be maintained as the data evolves over time and is rotated and translated by the user.

The remainder of this article is structured as follows. The

Manuscript received December 21, 2006; revised.

The authors are affiliated with Norrköping Visualization and Interaction Studio, Linköping University, Sweden

next section gives an overview of the most important related research including an overview of the haptic primitives method on which the dynamic volume haptics presented in this article has been based. Section III shows how the changes in dynamic transforms are handled for consistent feedback and section IV describes how the feedback is made consistent with the internal dynamics of sets of pre-generated frames of time-varying data. Steps that are needed to correctly handle the simultaneous rendering of multiple data sets are then described in section V. The implementation is discussed in section VI and the results are presented in section VII. The article is finally concluded with a discussion on future work, in section VIII, and conclusions, in section IX.

II. Related Work

There are essentially two ways of producing haptic representations of volumetric data. One way is by first extracting some intermediate geometrical representation of the data, for example an isosurface, and then applying a haptic surface rendering algorithm. Some key methods for surface rendering are described in section II-A. Extracting an intermediate representation of the data requires preprocessing and also provides representation only of a subset of the original data, such as the isosurface subspace.

The other way to introduce haptic feedback from volumetric data is by generating the feedback directly from the volumetric data, *Direct Volume Haptics* (DVH) in which the force feedback is generated directly from the local data at a probed position. This approach allows for the haptic representation of the full data, rather than just a subset. The methods for DVH can, in turn, be divided into two branches: force functions and constraint-based methods, see II-B and II-C below.

A. Surface Rendering

Surface haptics algorithms can be considered to have evolved through three steps[14]. The original *penalty method* produces a force directed towards the closest point on the closest surface of a penetrated object. The penetration depth multiplied by a stiffness constant gives the strength of the force. Lacking a memory of where the haptic probe penetrated an object, the penalty method produces discontinuities when the closest surface polygon changes. The memory needed to avoid this is introduced in the *god object-based method*, through the "god-object"[15], a point that slides on the surface of the objects. Due to the limitation in floating point precision inter-polygon edges have small gaps through which the point-sized god-object can slip. The problem is overcome by using a finite-sized sphere object, used by the *proxy-based method*, introduced in [14].

Both the god object and the proxy can be considered internal representations of the haptic probe, a point that can be moved by a haptic algorithm to mimic the behaviour of the probe, but in a stable and fully controlled manner. The surface haptics algorithms move this point to simulate the interplay between polygons and a virtual spring-damper coupling the probe and the proxy, see figure 2. The feedback, $\vec{f}_{\text{fb}}$, is estimated through

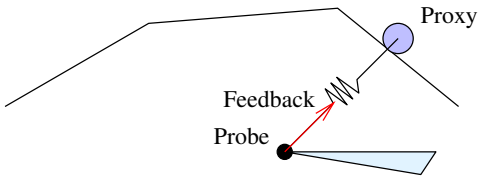


Fig. 2. The proxy is an internal representation of the haptic probe. It is controlled by the haptic algorithm to simulate the interplay between polygons and the virtual spring-damper that specifies the force feedback.

the spring-damper equation,

$$\vec{f}_{\text{fb}} = -k_s\left(\vec{x}_{\text{probe}} - \vec{x}_{\text{proxy}}\right) - k_d\left(\dot{\vec{x}}_{\text{probe}} - \dot{\vec{x}}_{\text{proxy}}\right) \quad (1)$$

where k_s is a stiffness constant and k_d is a damping term, and the proxy point functions as a memory of the point of palpation for the next time the feedback is estimated.

B. Force Function Methods

Without an intermediate polygon model of the data, the haptic algorithm has to directly convert the local volumetric data at the probed position to a force feedback. The most straightforward approach to DVH is the force function-based approach. Here the haptic feedback is expressed as a vector-valued function, $\vec{F}$, of the volumetric scalar or vector data at the position of interaction, the probe position, and the velocity of the probe

$$\vec{f}_{\text{fb}} = \vec{F}(\mathbf{\Lambda}, \vec{x}_{\text{probe}}, \dot{\vec{x}}_{\text{probe}}) \quad (2)$$

where $\vec{f}_{\text{fb}}$ is the force feedback, $\vec{x}_{\text{probe}}$ is the probe position and $\mathbf{\Lambda}$ is some property of the volumetric data. This property could be the data value itself, the gradient of a scalar data value, or any other suitable extracted value. Various haptic effects from this approach have been used to produce palpable representations of the data and to guide the user to interesting areas. For example, pushing the haptic instrument in the direction of the local gradient in CT data,

$$\vec{f}_{\text{fb}} = C\vec{\nabla}\mathbf{\Lambda}(\vec{x}_{\text{probe}}) \quad (3)$$

where C is a positive or negative force scaling constant and $\mathbf{\Lambda}$ is the scalar data, has proved useful in certain exploratory tasks[6], [7], [16]. The haptic instrument is then either pulled towards high density regions in the data, or pushed away from it towards low density data. This haptic effect has also been applied to other data than CT, for example for guidance in MRI data[17].

For vector data the most obvious approach is to use the vector as force feedback[6] although more advanced approaches have been proposed. In [8], for example, a vector representing the direction towards the centre of local vorticity is extracted and used to produce a haptic representation of the vorticity and a guidance force which helps the user follow the path of any vortex trail.

C. Constraint-based Methods

The constraint-based approach, first introduced in [9] and subsequently refined and applied in [10]–[13], [18], has been

developed to provide a more intuitive representation of the volumetric data compared to force function-based methods. A constraint-based method uses the local data to define local passive constraints that represent the data at any position in the volume. By letting the constraint yield when subjected to a force exceeding some material-specific magnitude, occlusion is avoided — the user can push through an occluding surface or other distinct feature. In this way a continuous representation of the data can be produced, removing the need for the user to activate and deactivate the haptic feedback to probe different regions. Using yielding constraints means that features in the data are represented by shapes, an effect that is naturally connected to human perception[19].

For example, the gradient vector in CT data has been used to generate a yielding constraint that represents surface-like features in the data[9], [10]. The interaction is similar to surface haptics, giving a distinct sense of position and shapes, in contrast with the 'magnetism-like' pulling and pushing of a force function rendering of the same data. The feedback can be assigned tissue specific material properties, such as friction and the strength needed to penetrate the surface. The constraint-based approach has also been used to produce haptic feedback from vector data, by generating an anisotropic friction that produces a resistance when moving the probe perpendicular to the field[10], [12], or tube-like representations of vortices[12].

Intuitively, constraints should move with moving data or with animated features in the data. In the implementations published to date, however, no consideration has been taken for the possibility of the data moving. Thus, the haptic feedback from moving or time-varying data reflects only the static configuration of the instantaneous data at the palpated position. A moving volumetric object can freely move straight through the haptic probe without generating any haptic feedback — the user can push against the haptic features, but a moving feature can never push against the haptic probe.

D. Proxy-based Volume Haptics

The constraint-based methods for volume haptics take their basic principles from state-of-the-arts methods for surface haptics, and thus make use of a proxy point. These methods, however, simulate the interplay between the virtual spring-damper, coupling the probe and the proxy, and implicit constraints representing the volumetric data at the proxy position. Thus, such algorithms can be generalized into three steps, also shown in figure 3.

1) extract data at proxy position, $\vec{x}_{\text{proxy}}$,
$$\Lambda = \mathbf{\Lambda}(\boldsymbol{T}^{-1}\vec{x}_{\text{proxy}}) \tag{4}$$
where $\mathbf{\Lambda}$ is the volumetric data property of interest and $\boldsymbol{T}$ is a transform describing the size and position of the data volume in world coordinates
2) use data to estimate proxy movements,
$$\vec{x}'_{\text{proxy}} = \vec{x}_{\text{proxy}} + \vec{F}(\Lambda,\ \vec{x}_{\text{proxy}}) \tag{5}$$
where $\vec{F}$ is an algorithm-specific function that moves the proxy in a manner that simulates the interaction between constraints and the probe
3) calculate the feedback, $\vec{f}_{\text{fb}}$, from the displacement of the new proxy point relative to the probe using equation 1

These three steps are similar in different proxy-based algorithms and essentially only $\vec{F}$ in the second step differs. This full procedure — extracting data, finding the new proxy position and calculating the force feedback — is generally performed in a dedicated haptic thread, at a rate of 1 kHz to ensure low latency and high fidelity feedback. The support for dynamic data is introduced through preprocessing of the proxy position before extracting data and calculating the proxy movements that represent these data, as described in sections III and IV.

In the simplest approach, used in [10]–[12], the extracted data is used to control the orientation and strength of an orthogonal set of simple single dimensional constraints. The proxy is then, in turn, moved in the direction of each constraint, j, according to

$$\vec{x}'_{\text{proxy}} = \vec{x}_{\text{proxy}} + \hat{q}_j \max(0,\ \vec{q}_j \cdot (\vec{x}_{\text{probe}} - \vec{x}_{\text{proxy}}) - s_j/k_s) \tag{6}$$

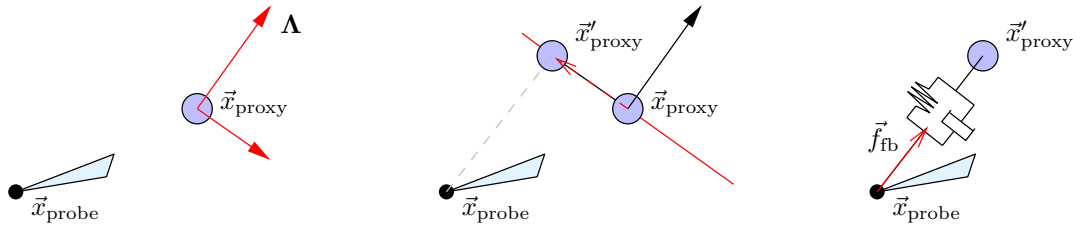
where $\vec{q}_j$ is the constraint direction, s_j is the strength of the constraint and k_s is the stiffness of the virtual coupling. Defining the orientation, $\vec{q}_j$, as the normalized gradient of the data, for example, the surface effect mentioned above is obtained. The value of s_j is generally controlled through a transfer function from some scalar property, for example the gradient magnitude for surface simulation.

E. Primitives-based Volume Haptics

Recently an algorithm based on haptic primitives has been developed that captures the potential effects of both force functions and constraint-based methods in a single framework[13]. This algorithm is a proxy-based algorithm, in principle defining the function $\vec{F}$ of equation 5, but hides this implementation behind an abstraction layer, haptic primitives. Since this is our preferred method for volume haptics it is this approach that has been used for the implementation of the dynamic volume haptics. Most of the techniques presented in this article could, however, be implemented using any proxy-based method for volume haptics. Some features for volume haptics are currently only supported through haptic primitives, such as support for non-orthogonal constraints required for the simultaneous handling of multiple data sets. The special considerations required for handling dynamics of multiple data sets, section V, are therefore discussed in the context of haptic primitives.

The haptic primitives form an abstraction layer for the implementation of haptic interaction schemes and also provide a means of calculating the feedback. Constraints are represented using primitives restricting the motion in one, two or three degrees of freedom: *plane*, *line* and *point*, respectively. Active forces and other force functions are included through a fourth primitive: *directed force*. Superpositions of these four primitives have been found to be sufficient to represent any force feedback scheme encountered to date.

The haptic primitives are characterized by simple vector-valued force functions of the proxy position, with parameters strength, s_j, position, $\vec{x}_j$, and direction, represented by a unit



(i) In the first step the data at the palpated position, $\vec{x}_{\text{proxy}}$, are examined.

(ii) In the second step the proxy is moved to reflect the extracted data.

(iii) Finally, the feedback is estimated through a virtual spring-damper.

Fig. 3. The three steps used to update the proxy position and calculate the force feedback. These three steps are similar in different proxy-based algorithms and only the second step, the moving of the proxy, differs.

vector, $\vec{q}_j$. The force fields of the primitives are illustrated in figure 4. Each constraint primitive generates a force of magnitude s_j towards the closest point on the primitive defined by the position and orientation. The plane primitive is semi-directed so that it provides feedback only when pushing the probe against a plane surface and not when moving away from it. Two such primitives can then be combined to produce a bi-directional planar constraint.

- Plane, a directed force which exists only on one side of the plane defined by $\vec{x}_j$ and $\vec{q}_j$:

$$\vec{f}_j\left(\vec{x}_{\text{proxy}}\right) = \begin{cases} 0, & \text{if } (\vec{x}_{\text{proxy}} - \vec{x}_j)\cdot\vec{q}_j \geq 0 \\ s_j\vec{q}_j, & \text{if } (\vec{x}_{\text{proxy}} - \vec{x}_j)\cdot\vec{q}_j < 0 \end{cases} \tag{7}$$

- Line, an attractor towards the closest point on a line:

$$\begin{aligned} \vec{f}_j\left(\vec{x}_{\text{proxy}}\right) &= \begin{cases} \vec{0}, & \text{if } |\vec{m}| = 0 \\ s_j\frac{\vec{m}}{|\vec{m}|}, & \text{if } |\vec{m}| \neq 0 \end{cases} \\ \vec{m} &= \vec{q}_j\left[\vec{q}_j\cdot(\vec{x}_{\text{proxy}} - \vec{x}_j)\right] - (\vec{x}_{\text{proxy}} - \vec{x}_j) \end{aligned} \tag{8}$$

- Point, an attractor to point $\vec{x}_j$ in space:

$$\begin{aligned} \vec{f}_j\left(\vec{x}_{\text{proxy}}\right) &= \begin{cases} \vec{0}, & \text{if } |\vec{x}_j - \vec{x}_{\text{proxy}}| = 0 \\ s_j\frac{\vec{m}}{|\vec{m}|}, & \text{if } |\vec{m}| \neq 0 \end{cases} \\ \vec{m} &= \vec{x}_j - \vec{x}_{\text{proxy}} \end{aligned} \tag{9}$$

- Directed force, a position-independent force:

$$\vec{f}_j\left(\vec{x}_{\text{proxy}}\right) = s_j\vec{q}_j \tag{10}$$

The proxy position is then found by balancing the force feedback, $\vec{f}_{\text{fb}}$, from the coupling equation (equation 1) against the force from the primitives, by minimizing the residual $\vec{\varepsilon}$ in

$$\vec{\varepsilon} = -\vec{f}_{\text{fb}} + \sum_j \vec{f}_j\left(\vec{x}_{\text{proxy}}\right) \tag{11}$$

with respect to $\vec{x}_{\text{proxy}}$. In the current implementation this is carried out using a numerical solver. When the minimum has been found the new proxy position perfectly represents, through the coupling equation, the haptic effect from the selected primitives. The residual will be zero unless the proxy point lies on a constraint primitive, in which case it will be non-zero. This is because of the discontinuity in the equations at the border of each constraint primitive. All primitive parameters (position, strength and orientation) are constant when estimating $\vec{\varepsilon}$.

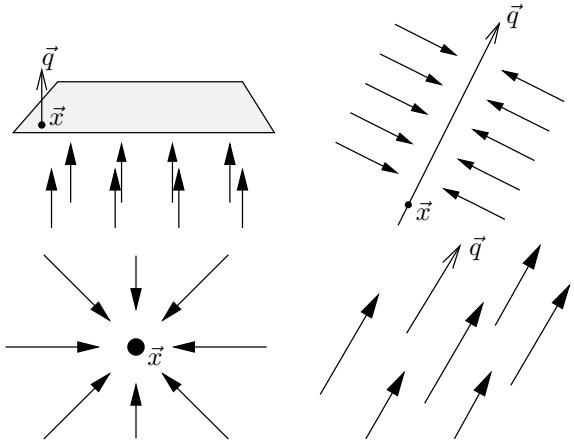


Fig. 4. The haptic primitives and their effects. From the top left: plane, line, point and force.

Using the primitives, schemes for haptic interaction can be handled as entities, *haptic modes*, simplifying the design and implementation of visio-haptic interfaces. Each haptic mode generates haptic feedback by placing one or more haptic primitives at the proxy position ($\vec{x} = \vec{x}_{\text{proxy}}$): during the haptic simulation each mode controls their parameters, such as orientation and strength, as functions of the local data at the proxy position, $\mathbf{\Lambda}(\boldsymbol{T}^{-1}\vec{x}_{\text{proxy}})$. In that respect the haptic mode acts as a link between the data and its haptic representation.

As an example, *surface-and-friction* feedback is implemented using two haptic primitives. A plane primitive oriented by the normalized gradient of the scalar data at the proxy position, $\vec{n}$, represents a surface. Friction is added using a line primitive with the same orientation, as shown in figure 5. The residual to minimize for this haptic mode then becomes, from equations 7, 8 and 11,

$$\begin{aligned} \vec{\varepsilon} = -\vec{f}_{\text{fb}} &+ \begin{cases} 0, & \text{if } (\vec{x}_{\text{proxy}} - \vec{x})\cdot\vec{n} \geq 0 \\ s_1\vec{n}, & \text{if } (\vec{x}_{\text{proxy}} - \vec{x})\cdot\vec{n} < 0 \end{cases} \\ &+ \begin{cases} \vec{0}, & \text{if } |\vec{m}| = 0 \\ s_2\frac{\vec{m}}{|\vec{m}|}, & \text{if } |\vec{m}| \neq 0 \end{cases} \end{aligned} \tag{12}$$

where $\vec{x}$ is set to the initial value of $\vec{x}_{\text{proxy}}$ and

$$\vec{m} = \vec{n}\left[\vec{n}\cdot(\vec{x}_{\text{proxy}} - \vec{x})\right] - (\vec{x}_{\text{proxy}} - \vec{x})$$

The strength of the surface is defined by the scalar data, Λ, at the proxy position using a transfer function, τ_{surf}, so that

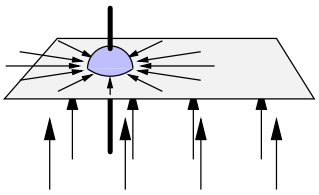


Fig. 5. Surface and friction simulation by using plane and line primitives at the position of the proxy point.

$s_1 = \tau_{\text{surf}}(\Lambda)$. The friction force is calculated from the normal force, estimated as the smallest value of the normal force, $k_s\vec{n} \cdot \left(\vec{x}'_{\text{proxy}} - \vec{x}_{\text{probe}}\right)$, and the current surface strength. The strength of the line primitive is then calculated by multiplying the normal force strength with a friction value obtained from a transfer function, τ_μ,

$$s_2 = \tau_\mu(\Lambda)\min(s_1,\ k_s\vec{n} \cdot (\vec{x}_{\text{proxy}} - \vec{x}))$$

Similarly the *follow-mode*, a mode that provides a guidance by generating a resistance to movement of the probe perpendicular to a vector field, is implemented using a single line primitive,

$$\begin{aligned} \vec{\varepsilon} &= -\vec{f}_{\text{fb}} + \begin{cases} \vec{0}, & \text{if } |\vec{m}| = 0 \\ s_3 \frac{\vec{m}}{|\vec{m}|}, & \text{if } |\vec{m}| \neq 0 \end{cases} \\ \vec{m} &= \Lambda\left[\Lambda \cdot (\vec{x}_{\text{proxy}} - \vec{x})\right] - (\vec{x}_{\text{proxy}} - \vec{x}) \end{aligned} \tag{13}$$

where Λ, here, is the local vector and s_3 is controlled through a transfer function, τ_{follow}, from the vector magnitude, $s_3 = \tau_{\text{follow}}(|\Lambda|)$.

Once defined, a haptic mode is calibrated through the haptic material transfer functions, τ. Multiple haptic modes defined in this manner can easily be combined. When two or more modes are used simultaneously, their individual primitives' force contributions are combined linearly in the balancing equation, as expressed by equation 11. Minimizing this residual formula then provides a new proxy position that represents the combined haptic effect of the arbitrary number of active haptic modes.

III. Dynamic Transforms

In this section we describe how the interaction point, and thus the volumetric features at that position, can respond to changes in the dynamic transforms of static volumetric data and so actively move the haptic probe. The feedback then responds to the collective motion of features in the data, described by the transform dynamics. This is a fundamental requirement for natural haptic interaction in a dynamic multimodal VR environment; the user can, for example, rotate the visualized data using a space mouse to explore it from an arbitrary direction, with consistent haptic feedback being generated throughout the movement.

A. Moving Feature Representations

So far the current static transform, $\boldsymbol{T}$, has been used to describe the location of the volumetric data in the virtual environment, and any changes between adjacent frames have been ignored. To produce haptic feedback that is consistent with movements in the scene, the proxy point should be moved according to changes in the transform of the data. This principle is shown in figure 6.

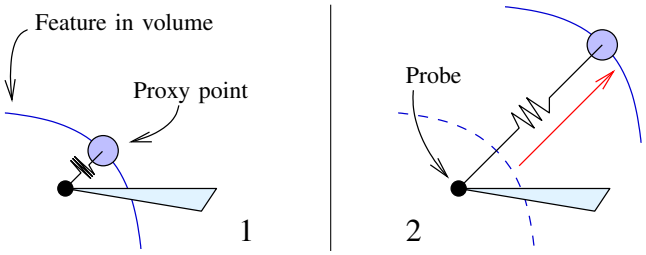


Fig. 6. The point of interaction, the proxy, is moved to reflect the movement of the volume of data.

The principle here is that the initial proxy position, before performing the three steps for proxy-based haptics calculation described in section II-D, should be the same in the current local data space as in the last local data space when the proxy position was last determined through equation 5. The changes of transformation between two frames are defined by the difference between the current transform, $\boldsymbol{T}_c$, and the previous transform, $\boldsymbol{T}_p$. Setting the proxy position, defined in world coordinates, equal in the current and previous local data spaces

$$\boldsymbol{T}_c^{-1}\,\vec{x}'_{\text{proxy}} = \boldsymbol{T}_p^{-1}\,\vec{x}_{\text{proxy}} \tag{14}$$

$$\vec{x}'_{\text{proxy}} = \boldsymbol{T}_c\,\boldsymbol{T}_p^{-1}\,\vec{x}_{\text{proxy}} \tag{15}$$

we obtain a new proxy position, $\vec{x}'_{\text{proxy}}$, representing the palpated position in the data after the changes in the model transform.

Using this new position as the initial proxy position when extracting data (equation 4) and calculating the haptic feedback (equations 5 and 1) produces a haptic representation at the updated position. In this way the point of interaction, and the feedback experienced, move relative to the transform difference which will produce an active pull in the direction of the data movement, as can be seen in figure 6.

B. Improving Haptic Frame-rate

Most systems for visio-haptic computer environments make use of two asynchronous threads, one for haptics and one for graphics, so that the haptic calculations can be executed at a higher rate and with lower latency than the graphical rendering. In a scene graph system the transforms are typically controlled in the same thread as the graphics, together with mouse events, game processing and other simulation event handling. Thus, the haptic process, rendered at a higher frame-rate, has to use the static knowledge between the graphics frames. Since the graphics frame-rate is low compared to the 1 kHz haptic frame-rate needed to produce smooth force feedback, the feature translation described above cannot be performed only when a new transform becomes available.

To obtain the inter-frame transform information needed for smooth feedback, we perform interpolation between the transforms available from the graphics thread, see figure 7.

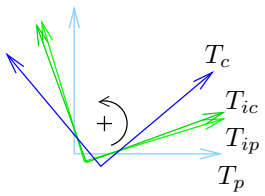


Fig. 7. Rotational transform example — the current (T_c) and the previous (T_p) graphics transforms of the haptic mode are recorded and used in the haptics thread to determine the current (T_{ic}) and previous (T_{ip}) interpolated haptic transforms.

Thus, instead of using the current and previous transforms ($\boldsymbol{T}_c$ and $\boldsymbol{T}_p$) for moving the interaction point (equation 15), the current ($\boldsymbol{T}_{ic}$) and previous ($\boldsymbol{T}_{ip}$) interpolated transforms are used,

$$\vec{x}'_{\text{proxy}} \quad = \quad \boldsymbol{T}_{ic}\, \boldsymbol{T}_{ip}^{-1}\, \vec{x}_{\text{proxy}} \tag{16}$$

The most recent interpolated transform, $\boldsymbol{T}_{ic}$, is then used to project to local space when extracting the local data, $\Lambda = \boldsymbol{\Lambda}(\boldsymbol{T}_{ic}^{-1}\vec{x})$.

C. Transform Interpolation

The interpolated transform is obtained by first splitting the two transform samples into scaling ($\boldsymbol{T}^s$), rotation ($\boldsymbol{T}^r$) and translation ($\boldsymbol{T}^t$). Scaling and translation are interpolated linearly and the rotation is interpolated using Spherical Linear Interpolation (Slerp), and the parts are then rejoined into the complete interpolated transform,

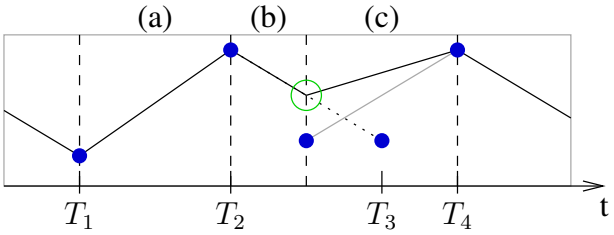
$$\boldsymbol{T}_{ic}^t \quad = \quad r\boldsymbol{T}_c^t + (1-r)\boldsymbol{T}_p^t \tag{17}$$
$$\boldsymbol{T}_{ic}^r \quad = \quad \text{Slerp}\left(\boldsymbol{T}_c^r,\ \boldsymbol{T}_p^r\ ;\ r\right) \tag{18}$$
$$\boldsymbol{T}_{ic}^s \quad = \quad r\boldsymbol{T}_c^s + (1-r)\boldsymbol{T}_p^s \tag{19}$$
$$\boldsymbol{T}_{ic} \quad = \quad \boldsymbol{T}_{ic}^t\, \boldsymbol{T}_{ic}^r\, \boldsymbol{T}_{ic}^s \tag{20}$$

where r is the interpolation ratio. Since the new transform is needed to perform a correct interpolation, the haptic rendering will lag by up to one graphics frame. It will then converge to the most current graphics transform.

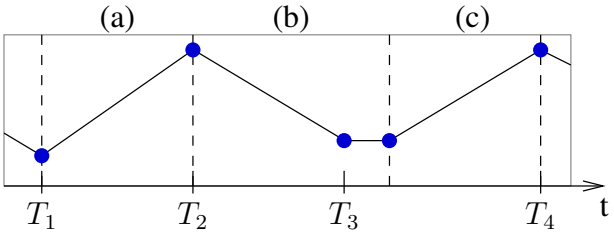
To perform the interpolation requires the current time into the graphics frame interval and the start and end times for that graphics frame. The time elapsed since the last graphics update is known, but not the remaining duration of the frame, when a new transform will become available — this needs to be estimated. The most obvious frame duration estimation is the mean of the time taken for the last few frames. Our tests have shown that the delay for rendering an individual frame is subject to noise and may vary significantly from the average because of interrupts in the operating system. Smoothing and removal of outliers is done by taking the inter-quartile mean (IQM) of the delay from the last N frames,

$$\Delta_{\text{IQM}} = \frac{2}{N} \sum_{i=N/4+1}^{3N/4} \Delta_i \tag{21}$$

for an ordered set, $\{\Delta_0,\ \Delta_1,\ \ldots,\ \Delta_N\}$, of frame delays. The size of N is a trade-off between an accurate mean and a rapid response to a long-term change in the graphics frame rate. In the implementation used in section VII, tests have shown that 12 frames gives a good balance.



(i) In frame interval (b) the interpolation between transforms T_2 and T_3 is not complete when T_4 becomes available earlier than expected. Thus, at the beginning of interval (c) there will be a discontinuity unless the current interpolated value (circled) is used to interpolate from, instead of T_3.



(ii) In this example transform T_4 becomes available later than expected and so the interpolation in frame interval (b) has finished before the new transform appears, resulting in a hold in the transform dynamics.

Fig. 8. The effect of different misestimations of the frame duration. In frame interval (a) T_2 is available as the new transform and T_1 is used to interpolate from. With a correctly estimated frame duration, the interpolated value converges with transform T_2 at the end of the frame interval.

Since the frame time can only be estimated it is bound to result in too long or too short an estimate (see figure 8). Frames that take less time than the estimate will then produce a sudden jump at the end of the frame period. This is caused by the significant difference between the new $\boldsymbol{T}_{ic}$ value and $\boldsymbol{T}_{ip}$ from the previous frame (figure 8(i)). To reduce this problem, we use the last interpolated transform, $\boldsymbol{T}_{ic}$, as the old transform to interpolate from instead of the transform that was currently the target for the interpolation, see figure 8(i). Thus, as a new transform becomes available, $\boldsymbol{T}_p$ is set equal to $\boldsymbol{T}_{ic}$ instead of $\boldsymbol{T}_c$, providing a C^0 continuous transform interpolation.

Similarly, a frame that is shown for longer than the estimated frame duration will have a slight pause at the end of the interpolation, since the haptic interpolation will finish before a new transform is available (figure 8(ii)). This pause is already C^0 continuous.

D. Dynamic Transforms Summary

In this section the changes in the transformation of static volumes have been used to control the movement of the proxy point, prior to rendering shapes and other features in the data, to enable smooth haptic feedback consistent with the scene dynamics. This handling of changes in the scene is essential for the general handling of volume visualization in a larger virtual environment — the feedback will be consistent with the motion, rotation and scaling of the volumes. The following section discusses how similar principles can be applied to the

more general case where the data inside the volume changes between frames.

IV. Time-varying Volumetric Data

The principle used above to add support for active dynamic transforms can be expanded to also provide feedback that is consistent with changes in time-varying data. In this section we consider sets of pre-captured data frames, each consisting of a static data volume. This can be a series of CT volumes or data from CFD simulations, for example. Graphical rendering of such time-varying data is a matter of showing some visual representation of the data frames in sequence. When computing the haptic feedback, however, the changes between the data frames must also be taken into account so that the haptic probe can be pushed by a moving feature in the data and thus provide a tactile sense of the dynamics in the data.

In section III the changes in the model transforms are used to determine the movement of the data, and thus also the collective movement of the features it contains, between frames. In sets of data volumes the changes between adjacent data frames are not known, so some explicit information about these changes is needed. Without this, there is no way of knowing how features are moving within the volume and so it is impossible to perform the proxy movements described above. It is, therefore, assumed that for each data frame there exists a motion field, $\vec{\boldsymbol{M}}$, describing with a vector for each position how that particular data point moves to a new location in the following data frame. This motion field may, in some cases such as with simulation data, be readily available, or can be computed as discussed in section IV-E.

A. Animation Control

We begin by assuming that the animation of the time-varying data is controlled by a fractional frame pointer that specifies the current position in time. In the graphics thread, this value is used to control the visual playback of the animation. By extracting the integer part of the pointer value, the current frame to show is determined.

In the haptic thread the fractional frame pointer is used to control the haptic animation. Two values are of importance: the fractional frame pointer in the current haptic frame interval, F_{ic}, and the value from the previous frame interval, F_{ip}. Note that to perform a smooth haptic interpolation between data frames, the frames must be at known points in time, but these points do not have to be equally spaced.

B. Moving Feature Representations

In visual rendering the same data frame can be shown for the full duration of that frame — the visual sense performs an interpolation to provide a sense of continuity even at fairly low frame-rates. In haptics we need a much higher frame-rate, preferably not less than 1 kHz, to give a smooth sense of continuity. To obtain this higher frame-rate, we treat the data frames as key-frames and perform a haptic interpolation between them.

In the case of dynamic transforms the interpolated transform is used to determine the data movements during haptic rendering between graphics frames. In this new case, however, we do not have information about what is happening between data frames. The animation frames are samples that can be considered as hyperplanes in the space-time of the dynamic data, see figure 10. As long as the fractional frame pointer is not an exact integer value, the proxy cannot be considered to be "on a data frame" in time. Thus, to be able to extract volume data from one of the static volumes of the data sequence, the inter-frame proxy from the previous haptic calculations, $\vec{x}_{\text{proxy}}$, must first be back-tracked to its corresponding position, $\vec{x}^{P}_{\text{proxy}}$, in the data frame of the previous calculations, $\lfloor F_{ip} \rfloor$. The details on proxy back-tracking are presented in section IV-D.

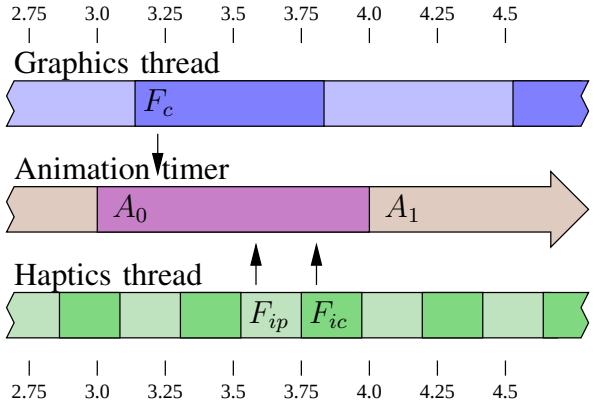


Fig. 9. The fractional frame pointer is used in the graphics thread to determine which data frame to show. In the haptics thread, the value is used to control the haptic interpolation.

The motion field vector at the data frame proxy position describes how the local features in the data move between the data frames. To derive the haptic effects of features moving between the individual frames we update the proxy, the point of interaction, to simulate a motion along the local motion field vector,

$$\vec{x}'_{\text{proxy}} = \vec{x}^{P}_{\text{proxy}} + (F_{ic} - \lfloor F_{ic} \rfloor)\, \vec{\boldsymbol{M}}(\vec{x}^{P}_{\text{proxy}}) \qquad (22)$$

where $F_{ic} - \lfloor F_{ic} \rfloor$ is the current fraction between the data frames. Back-tracking the proxy position using the previous fractional frame pointer, F_{ip}, and then moving the proxy using the current fractional frame pointer, F_{ic}, makes the haptic interaction point move with the motion field.

This new proxy position, $\vec{x}'_{\text{proxy}}$, is used in equation 5 when estimating the haptic effect generated by the data extracted from the volume. These data must, like the motion vector in equation 22, be extracted from the data frame using the data frame proxy position, $\vec{x}^{P}_{\text{proxy}}$. In this way the interaction point is interpolated between the data frames while the data controlling the effects of the haptic feedback are extracted from the corresponding position in the volume frames of the data sequence.

To summarize, the steps taken to produce feedback that is consistent with changes between data frames are:

1) back-track the inter-frame proxy to its corresponding position in the data frame

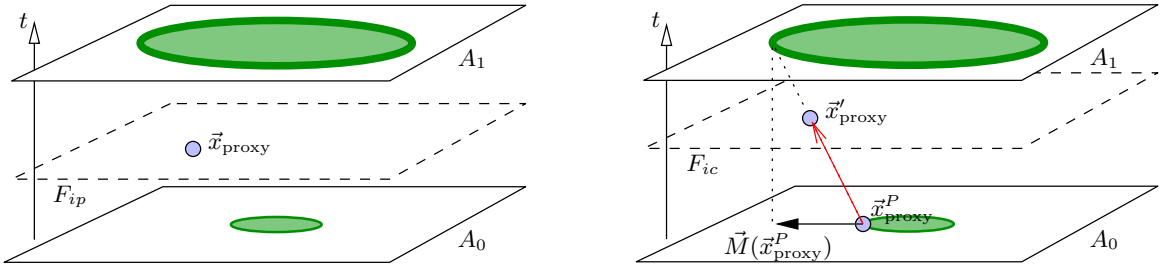


Fig. 10. The proxy can always be considered to be located between two adjacent data frames in space-time. Thus, the motion field has to be used to first back-track this position onto the current data frame for data extraction (left), and then move the proxy forwards to the current inter frame position (right) before the proxy movement that determine the haptic effects is calculated.

2) use this position to extract data that should control the effects from the haptic feedback (equation 4)
3) move the data frame proxy forwards to its new inter-frame position
4) use the new proxy position as initial state when calculating the haptic effect (equation 5)

Since the probe position is generally used as part of $\vec{F}$ in equation 5 to determine the proxy movements that should produce the desired haptic effect, the user input can always potentially modify the state and, for example, make the proxy move sideways over a moving surface or even penetrate the currently palpated feature. Thus, the proxy will not, in reality, move straight along the line specified by $\vec{\boldsymbol{M}}$, which makes the back-tracking and forward-moving necessary each time the feedback is calculated.

C. Data Value Interpolation

In time-varying CT data the features in the data generally have similar scalar values in adjacent data frames, but not at the same location in the frames. This characteristic does not apply to all types of data. For example, in CFD data the pressure of a shockwave will decrease as the wave front propagates through the volume. To avoid temporal discontinuity in such cases, we introduce inter-frame temporal data interpolation.

When the data frame proxy position, $\vec{x}^P_{\text{proxy}}$, has been determined as described above, its corresponding position in the next data frame is determined through

$$\vec{x}^{P'}_{\text{proxy}} = \vec{x}^P_{\text{proxy}} + \vec{\boldsymbol{M}}(\vec{x}^P_{\text{proxy}}) \tag{23}$$

This position can then be used to extract data from the next data frame. The data used to control haptic effects, Λ, are thus estimated by interpolating between the data extracted from the current data frame and the next data frame, A_0 and A_1 respectively (figure 10),

$$\Lambda = (1-r)\boldsymbol{\Lambda}_0(\vec{x}^P_{\text{proxy}}) + r\boldsymbol{\Lambda}_1(\vec{x}^{P'}_{\text{proxy}}) \tag{24}$$
$$r = F_{ic} - \lfloor F_{ic} \rfloor \tag{25}$$

where $\boldsymbol{\Lambda}_0$ and $\boldsymbol{\Lambda}_1$ are the volumetric data property of interest from frames A_0 and A_1.

D. Position Back-tracking

The back-tracking is needed to find the proxy position in the current data frame that corresponds to the current inter-frame proxy position. This can be expressed as finding the point $\vec{x}^P_{\text{proxy}}$ for which

$$\vec{x}^P_{\text{proxy}} + \gamma\vec{\boldsymbol{M}}(\vec{x}^P_{\text{proxy}}) = \vec{x}_{\text{proxy}} \tag{26}$$

where $\gamma = F_{ip} - \lfloor F_{ip} \rfloor$, a value in the range $[0, 1)$. This equation can be rewritten as a fixed-point problem in 3D space, $\vec{g}(\vec{x}) = \vec{x}$, where

$$\vec{g}(\vec{x}) = \vec{x}_{\text{proxy}} - \gamma\vec{\boldsymbol{M}}(\vec{x}) \tag{27}$$

The solution to fixed-point problems can be found by iteratively applying the function, starting from an initial estimate. We use the data frame proxy position, $\vec{x}^P_{\text{proxy}}$, from the estimation for the last haptic frame as an initial estimate, $\vec{x}_0$. This value is not correct since the proxy has been moved relative to the data frame in response to user actions since then, but since the movement between two haptic frames is generally small, this is an adequate first estimate.

The Banach fixed point theorem (1922) states that a necessary and sufficient condition for successfully finding the fixed-point in a compact space, a closed and bounded subset of $\mathbb{R}^n$, is $d(Tx, Ty) \leq q\, d(x, y)$, where $q < 1$ and T is the mapping transform, in our case corresponding to $\vec{g}$ in equation 27. This gives us, from equation 27 with γ set to one corresponding to the worst case at the end of each data frame,

$$\left|\left(\vec{x} - \vec{\boldsymbol{M}}(\vec{a})\right) - \left(\vec{x} - \vec{\boldsymbol{M}}(\vec{b})\right)\right| \leq q\left|\vec{a} - \vec{b}\right| \tag{28}$$
$$\Downarrow$$
$$\left|\vec{\boldsymbol{M}}(\vec{b}) - \vec{\boldsymbol{M}}(\vec{a})\right| < \left|\vec{a} - \vec{b}\right| \tag{29}$$

where $\vec{a}$ and $\vec{b}$ are points in the compact space.

We let the current local region, where the solution is sought, be our compact space. This is defined by the initial estimate ($\vec{x}_0$), the data frame proxy position ($\vec{x}^F_{\text{proxy}}$) and the immediate neighbourhood, the region containing the intermediate values from the iteration. This indicates that the critical factor for finding the correct solution to equation 26 is the temporal resolution of the animation relative to the speed of the changes in the volume — high temporal resolution reduces the magnitude of $\vec{\boldsymbol{M}}$, while slow dynamic behaviour lowers the difference between $\vec{\boldsymbol{M}}$ at neighbouring positions. Moreover, large movements of the haptic instrument enlarge the size of the local neighbourhood, which also has a negative impact on the conditions.

The back-tracking should move the proxy point into the current data frame. If, in the interval between the previous

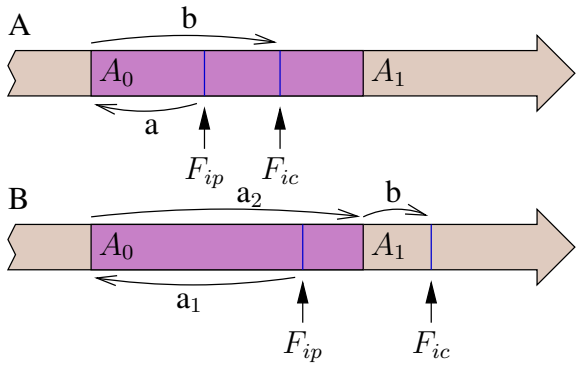


Fig. 11. Two different cases are considered when using the motion vector information to produce interpolated haptic feedback: A) both the current and previous fractional frame pointer refers to the same data frame interval and B) the current fractional frame pointer lies within a new frame interval.

and current haptic frame, a new data frame has been crossed then the proxy must be advanced by the full extent of the motion of the previous data frame, so that the proxy point is at the start of what is now the current data frame when extracting the data, see figure 11. Thus, we have two cases of back-tracking shown in figure 11: (A) when the current and the previous fractional frame pointer lie within the same data frame interval, and (B) when the current fractional frame pointer refers to a new data frame interval. This is also done with the initial estimate mentioned above. Note also that a wrap-around in a cyclic animation falls under case B.

If other position parameters expressed in world coordinates, such as the probe position, are to be used to extract data then these parameters must also be back-tracked to their frame-specific locations. Subsequently, if any parameter is also used when calculating the haptic effect, it must also be moved forward, as expressed by equation 22.

E. Estimating the Motion Field

The motion field, $\vec{\boldsymbol{M}}$, may be readily available when exploring time-varying data acquired from simulations. In the case of interaction with most scanned data, such as MRI or CT data, or with legacy data where motion information is unavailable, this motion field needs to be estimated from the set of static volumetric data frames.

The main requirement for the algorithm used to produce the motion field is that it generates a dense field, that is a field that defines a motion vector for each voxel. Data co-registration algorithms, for example, provide dense fields describing the relationship between data sets, so applying such algorithms on pairs of adjacent frames, $\mathbf{\Lambda}_0$ and $\mathbf{\Lambda}_1$, produces this type of data. These algorithms aim at minimizing the error, ε, in

$$\varepsilon^2 = \left\| \mathbf{\Lambda}_1(\vec{x} + \vec{\boldsymbol{M}}(\vec{x})) - \mathbf{\Lambda}_0(\vec{x}) \right\|^2 \tag{30}$$

that is, find a deformation that projects one data set into the other. An example is the Demons algorithm presented by Thirion in [20]. This algorithm is available as a filter in ITK[21], which we use to pre-process the data and subsequently to extract the motion field.

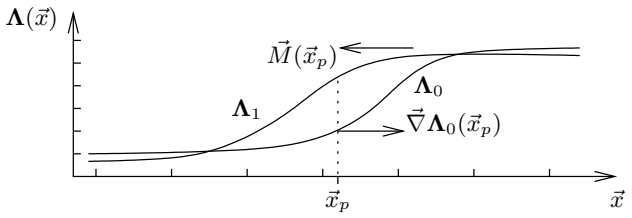


Fig. 12. This 1D example shows the implications of the instantaneous optical flow equation for two discrete time steps (equation 31) at a position $\vec{x}_p$.

The Demons algorithm is based on the instantaneous optical flow equation for two discrete time steps, which is designed to identify the local motion of objects in space,

$$\vec{\boldsymbol{M}}(\vec{x}) \cdot \vec{\nabla}\mathbf{\Lambda}_0(\vec{x}) = -\left(\mathbf{\Lambda}_1(\vec{x}) - \mathbf{\Lambda}_0(\vec{x})\right) \tag{31}$$

see figure 12. This equation is under-defined and needs an extra constraint to be solvable. This constraint is provided by projecting the deformation vector onto the gradient, resulting in an expression for the deformation or motion field, $\vec{\boldsymbol{M}}$,

$$\vec{\boldsymbol{M}}(\vec{x}) = -\frac{\left(\mathbf{\Lambda}_1(\vec{x}) - \mathbf{\Lambda}_0(\vec{x})\right)\vec{\nabla}\mathbf{\Lambda}_0(\vec{x})}{\|\vec{\nabla}\mathbf{\Lambda}_0(\vec{x})\|^2} \tag{32}$$

To make the expression less sensitive to small values for the image gradient, a value must be added to the denominator so that it does not approach zero. Thirion suggests the use of the volume value difference, resulting in

$$\vec{\boldsymbol{M}}(\vec{x}) = -\frac{\left(\mathbf{\Lambda}_1(\vec{x}) - \mathbf{\Lambda}_0(\vec{x})\right)\vec{\nabla}\mathbf{\Lambda}_0(\vec{x})}{\|\vec{\nabla}\mathbf{\Lambda}_0(\vec{x})\|^2 + \left(\mathbf{\Lambda}_1(\vec{x}) - \mathbf{\Lambda}_0(\vec{x})\right)^2} \tag{33}$$

This equation describes only a deformation determined from the *local* comparison between the two volumes $\mathbf{\Lambda}_0$ and $\mathbf{\Lambda}_1$. The full deformation is found by iteratively refining the deformation field, according to

$$\vec{\boldsymbol{M}}^{n+1}(\vec{x}) = \vec{\boldsymbol{M}}^{n}(\vec{x}) - \frac{\left(\mathbf{\Lambda}_1(\vec{x} + \vec{\boldsymbol{M}}^{n}(\vec{x})) - \mathbf{\Lambda}_0(\vec{x})\right)\vec{\nabla}\mathbf{\Lambda}_0(\vec{x})}{\|\vec{\nabla}\mathbf{\Lambda}_0(\vec{x})\|^2 + \left(\mathbf{\Lambda}_1(\vec{x} + \vec{\boldsymbol{M}}^{n}(\vec{x})) - \mathbf{\Lambda}_0(\vec{x})\right)^2} \tag{34}$$

where $\vec{\boldsymbol{M}}^{n}$ is the nth refinement's deformation field. To ensure connectivity in the resulting deformation field, a Gaussian filter is applied to $\vec{\boldsymbol{M}}$ after each iteration.

Since the range of the gradient is small relative to the changes in most sequences of data, the estimation of the motion field is preferably performed in multiple resolutions. The result from the deformation estimation of one level of resolution is used as the initial deformation for the estimation of a finer level of resolution, starting with a initial zero deformation at the lowest level. We perform this multi-resolution deformation estimation in size doubling steps, resulting in $\log_2 256 = 8$ levels for a volume of 256^3 voxels.

This procedure, applied to a scalar data property, produces a motion field describing the changes between one data frame and the next. It should be noted, however, that while the motion field for a scalar data set describing the location of objects in space (such as CT) may seem unambiguous, this

is not always the case. Different properties of a specific set of scientific data may move in completely different directions between two frames. For example in CFD the dynamics of interest may be the motion of features such as vortices, pressure regions or flow bifurcations. Each of these move in separate directions and would need a separate motion field. The $\mathbf{\Lambda}$ of equation 34 must therefore be carefully selected so that the resulting motion field describes the motion of the features represented by the current haptic mode. Thus, since two haptic modes may describe different features from the same data, these may need different motion fields to control the dynamics in that same data.

F. Time-varying Volumetric Data Summary

In this section the changes between captured data frames have been used to provide a smooth haptic feedback reflecting the changes in the data. To achieve this the current interaction point, the proxy, is first used together with a motion field to determine the interaction position relative to the current data frame. The local data at this position is used to control the haptic effects in the current visualization. The motion field is then used again, this time to move the proxy forwards, so that the new position represents the motion of features from the previous proxy position. This position is used when calculating the haptic effect. This produces a haptic interpolation of the motion or flow of features in the data between two data frames. These frames may even be seconds apart in animations with low temporal resolution or during slow playback.

V. Multiple Data Volumes

The primitives-based approach to volume haptics is designed so that multiple data visualizations can be simultaneously handled in a single virtual environment, and data provided by different modalities can be co-registered to produce a combined haptic effect. Each haptic mode that is part of the balancing equation, equation 11, is potentially assigned an individual data set and an individual transform, for example by being at a different location in a scene graph. Proxy, probe and primitive positions are described in world coordinates and the projection into the data space is described by the inverse of the transform of the haptic mode, $\boldsymbol{T}$. Thus, different parts of the sum in equation 11 may be subject to different transforms and data so that, for example, follow-mode feedback is provided by one volume at one position in space and surface feedback is generated by another volume at a different location.

In section III we describe how the proxy point must be translated in response to changes in the transforms to allow for haptic feedback that is consistent with these changes. Since the haptic modes may have individual transforms, this must be performed separately for each haptic mode, which leads to an individual proxy position for each haptic mode. These are temporary points, however, used only to define the mode-specific primitive parameters such as position and strength. When minimizing the balancing equation all primitive parameters are expressed in world coordinates, producing a single new proxy position again in world coordinates. The same considerations must also be taken into account when handling time-varying data (section IV). Since the haptic modes may have different data and thus also different motion fields, the data frame proxy position may be specific to each haptic mode.

VI. Implementation

The support for dynamic transforms and time-varying data has been integrated into the publicly available Volume Haptics Toolkit[18], our earlier implementation of the haptic primitives-based method. The toolkit is built on H3D API, an X3D-based and cross-platform scene graph system for multi-modal applications. It provides haptic modes, visual components and data handlers as scene graph nodes for fast and simple construction of advanced multi-modal applications. In the toolkit each haptic mode is represented by an individual scene graph node providing the haptic representation of data in a manner similar to the way that a graphics node may produce a visual representation controlled by transforms and parameters in the scene graph.

Each haptic mode node extends a super-class that handles transform caching, data interpolation and extraction, and data animation. A singleton core handles the collecting of haptic primitives from all active haptic modes. Since all primitive parameters are held constant after being set, these primitives by themselves fully define the haptic behaviour and no knowledge about the data or features are needed outside of the mode node. The core node also moves the proxy with transform changes, and performs the proxy back-tracking and movements following the approach presented here.

By applying the measures discussed in sections III and IV in sequence, the full support for dynamic data is implemented. By performing the forward movement on the primitives instead of the proxy, see figure 13, the haptic modes can work entirely in the space of the data frames. The primitives are then fed into a solver that solves equation 11 and returns the new proxy position. With this approach both the haptic modes and the primitives' solvers are unaware of the dynamics of transforms and data. This greatly simplifies the implementation of new haptic modes and solvers.

VII. Results

The implementation is run on a SenseGraphics IW Display equipped with 19" stereoscopic CRT, a Desktop PHANToM and a Magellan SpaceMouse. The display is driven by a dual Xeon 2.0 GHz PC with 256 MiB RAM and a Quadro FX 3000 graphics card. The system is calibrated for co-located and co-registered haptics and graphics, and the world coordinate system is calibrated with real-world metric units.

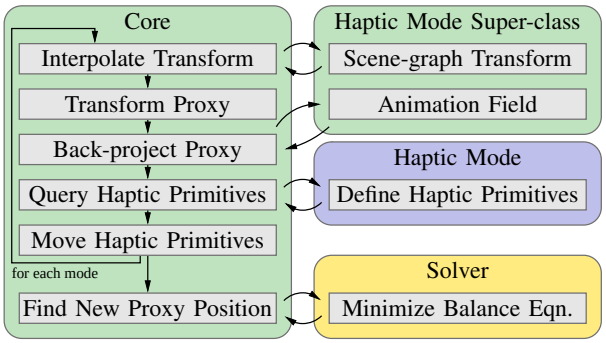
It is hard to communicate the effect of haptics in the static medium of an article, and dynamic haptics is even harder. The results below attempt to describe both objective observations, such as timings, tested data sets and experimental details, as well as subjective experiences by experienced users of haptic visualization.

A. Dynamic Transforms

We tested the dynamic transform support on a simple visualization of an analytical electropotential field of a

```
for each mode in list of haptic modes
  interpolate transform
  forward transform proxy
  back−track proxy to get frame proxy
  call get primitives in mode
    use frame proxy to get data
    use data to define primitives
    use frame proxy to position primitives
    return primitives
  end call
  move forward primitives' positions
end for
call get new proxy position in solver
return spring feedback
```

(i) This pseudo code shows the combined use of the steps presented in this article.



(ii) This structural figure of the current implementation shows how different parts communicate. All haptic modes have a common super class that handles the accumulated transform, the time-varying data and the motion field.

Fig. 13. The estimation of proxy position and haptic feedback that is performed for each haptic frame.

dichloroethane molecule, see figure 14. The data is rasterized into a 3D texture for the visual volume rendering while the haptic rendering is performed directly on the analytical data. The haptic feedback is generated using the follow-mode, see section II-E, applied to the gradient of the electropotential. This mode provides a feeling of both the orientation of the field by guiding the probe to follow the field lines, and the strength of the electropotential through the strength of the haptic feedback, the force needed to move the probe perpendicular to the field. Stream-tubes, interactively released in the virtual environment using the haptic instrument, provide a visual impression of the field used to control the haptic feedback. The molecule model can be freely rotated using a Magellan SpaceMouse and the dynamics support provides an active drag of the haptic probe that follows the changes in the transform.

Without the support for dynamic transforms, the user will get no feedback from the rotation of the visualization: even if the data is rotating, the feedback will provide a static haptic representation of the data, so that the data at the probed position is experienced as static but with a changing value.

Activating the dynamic transform support, the feedback becomes consistent with the changes in the model transform.

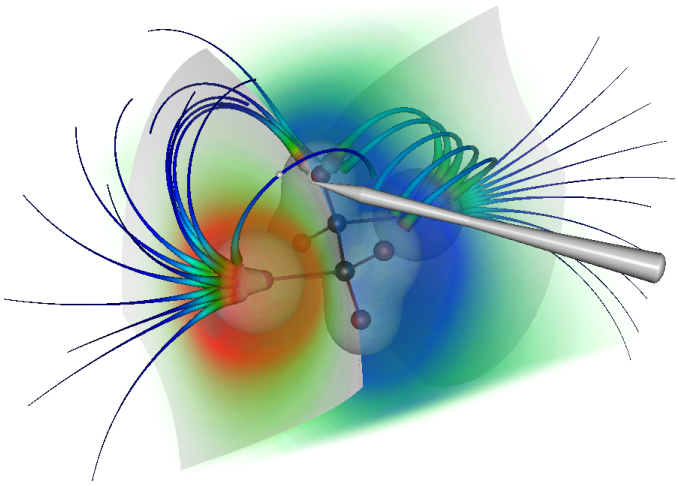


Fig. 14. The dichloroethane electropotential visualization with dynamic transforms controlled by a Magellan SpaceMouse. The haptic device is represented by a pen model with a small sphere in front of the pen at the probe position.

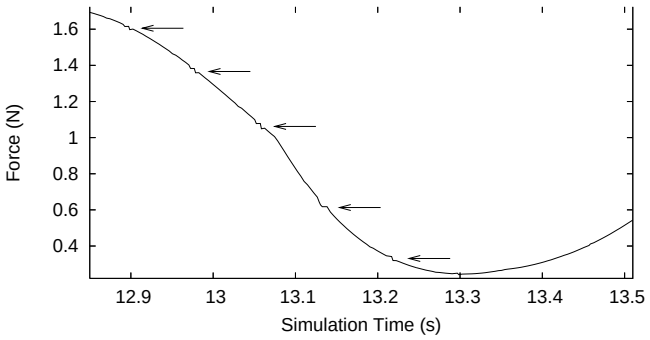


Fig. 15. In this graph, that shows the feedback force during interaction with a rotating data set, the artifacts from misestimated frame time can be seen as an unevenness in the force feedback at the end of every graphics frame.

For example, the moving haptic representations of the data pull the haptic probe with the rotation of the data. The lag of up to one graphics frame is unnoticeable at normal graphics frame-rates and the transform interpolation works well and produces smooth feedback. With large movements in combination with uneven frame-rate, however, the C^1 discontinuity mentioned in section III-C can be noticed as an irregularity in the feedback from, for example, a continuous rotation. This is experienced as a small tick at the transition between two graphical frames, see graph in figure 15. It should also be noted that controlling the transforms through a device outside of the haptic control loop will add energy to the system, which can lead to unstable behaviour.

B. *Time-varying Data*

The support for time-varying data has been tested in two configurations with different data sets, one synthetic with known changes over time and one authentic with unknown changes that need to be estimated. The authentic data set is a CT scan sequence of a beating human heart and the synthetic data set is a simulation of the same type of data.

1) Haptic Feedback: For haptic feedback from these data sets we apply the surface and friction mode, see section II-E. This mode extracts the gradient vector from the scalar data and uses this as an estimation of a local surface. The effect is the feeling of surfaces at locations where the scalar value in the CT data changes, typically at borders between tissues. The yielding constraints allow the user to apply a force exceeding the tissue specific strength and so penetrate the surface.

Exploring either of these animated data sets without the support for dynamics introduced here produces the impression of static data. The movements of, for example, the heart wall do not provide any haptic feedback. Only when the haptic instrument is moved by the user towards or over a tissue surface is the feedback exerted. The feeling is then of a static surface and as the animated surface moves away from the probed position, the feedback disappears.

2) Synthetic Data: This data set is of 128^3 voxels and both scalar data and motion field have been analytically estimated to provide a minimal error testing suite. The "heart" is a spherical shell of high scalar values surrounded by low values, and the "beating" consists of this shell repeatedly cycling through the phases small, intermediate, large and then back again to intermediate, four frames in total. The first three frames, from the expansion phase, are shown in figure 16 (leftmost, middle and rightmost). This setup has been tested at animation rates of 1–4 fps.

The feedback from the time-varying volumetric data is smooth and provides the same feature representation as interaction with static data, but now consistent with the changes in the data, providing a sense of moving haptic features. As the heart expands, the haptic features push the haptic instrument forwards and as the heart contracts, the features yield allowing the instrument to move inwards.

Since the animation frame-rate is independent of the graphics frame-rate of the running system, the difference between haptic and graphics feedback is more apparent here than with the dynamic transforms. With the haptic feedback interpolated between the data frames it can appear, particularly with low data frame-rates, as if the surface is pushing the probe away from the visual heart wall when the heart is expanding or yielding when the heart is contracting. The push effect can be seen in the expansion phase in figure 16, frames 2 and 4. Increasing the data frame-rate reduces this effect and with a rate experienced as smooth, the effect becomes unnoticeable.

3) Authentic Data: We have also tested the algorithm on an authentic CT of a beating heart, shown in figure 17. This data set, acquired through CT synchronized with EKG, is of 10 frames for the full beat cycle (to run at approximately 10 frames per second) with $512^2 \times 400$ voxels size and a resolution of about 0.5 mm/voxel. The data was downsampled to $128^2 \times 100$ to make the full animation fit in the available memory. The motion field estimation for this data set was implemented using ITK to perform a coarse-to-fine multi-resolution refinement as described in section IV-E.

The haptic feedback makes it possible to touch and track by touch the moving heart walls, and feel the movements. Touching the heart from the outside, the beating surface can be freely palpated and, by simply applying a force exceeding the strength defined for the heart muscle, the wall can be penetrated and the inside of the heart chamber can be examined.

In the case of the motion field being estimated from the set of frames, the accuracy of the haptic feedback depends heavily on the quality of this estimation. The required quality also varies for the different haptic modes available for scalar data due to their different nature. In the case of the surface and friction mode used here, the demands are high. Palpating regions where the motion field has been miscalculated could result in haptic artifacts such as 'pop-through' of the heart wall and temporal discontinuities in palpated shapes.

TABLE I

TIME NEEDED TO ESTIMATE THE HAPTIC FEEDBACK WITH AND WITHOUT SUPPORT FOR DYNAMIC TRANSFORMS AND ANIMATED VOLUMES.

Activated Support	CPU Time
Static Volumes	0.036 ms
Dynamic Transforms	0.045 ms
Dynamic Transforms & Time-varying Data	0.66 ms

C. Performance

To check the performance of the presented approach, the implementation has been timed with the authentic heart configuration described above. To estimate the CPU time the delay in estimating the haptic feedback was recorded for a typical exploration session. This was done for the same simulation and in similar conditions in three compilations of the software: without support for feedback from dynamics, with support for dynamic transforms but not animated volumes, and with support for both dynamic transforms and animated volumes. For each version the typical CPU time on the current platform is presented in table I. The time is independent of haptic rendering state and has $\mathcal{O}(N)$ time complexity with respect to the number of haptic modes used, since each mode potentially needs a separate processing of the proxy position.

The results show that adding support for dynamic transforms adds marginal extra load on the CPU. The support for haptic feedback from animated data, however, adds considerable extra time to the feedback estimation. It is primarily the back-tracking that adds to the computations. This procedure includes multiple vector data extraction from the motion field, each of which requires interpolations and world to local space transformations. The total time needed, however, is still generally below the limit of 1 ms required to keep the 1 kHz update rate.

VIII. FUTURE WORK

The results show that the presented algorithm is capable of allowing feedback that is consistent with dynamics in the data and that the haptic effects are both effective and predictable, however there remain some development areas for the support for animated data. Firstly the requirement of a high quality motion field makes this method heavily dependent on the success of algorithms for optic flow or registration. Since direct volume haptics is not connected to an isosurface, streamlines or some other object that can be exactly located on a frame to frame basis, errors in each haptic frame may

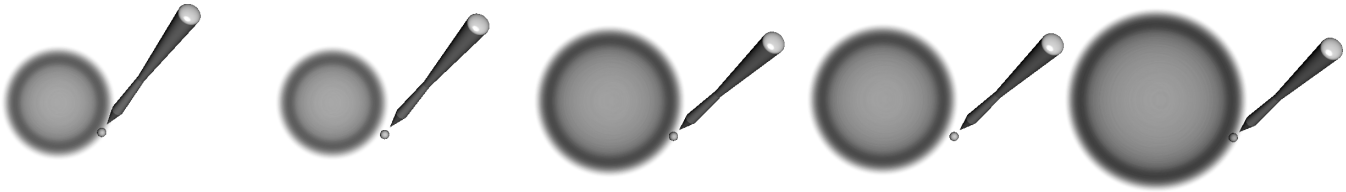


Fig. 16. A part of the sequence of a simulated CT of a beating heart. Images 1, 3 and 5 show the on-surface effect when haptic and graphics frames are in sync, while 2 and 4 show the effect of the haptic interpolation which gives the impression of the haptic probe lying on a virtual surface outside the visual surface.

be accumulated to eventually yield a noticeable error in the haptic behaviour. The Demons algorithm used in the work presented here follows a general optical flow-based approach that uses simply the intensity values and gradient of the scalar volumes. Other phenomena that are unique to specific volumetric data are not handled by optic flow algorithms, such as conservation of mass in CT. By considering the properties of the specific data at hand a more accurate, precise and complete definition of the motion field could potentially be extracted. Consequently, in the continuation of this project we wish to make use of specialized algorithms, for example those presented in [22]–[24]. This is an important line of research for more accurate and effective haptic interaction with a wider variety of dynamic data.

Artifacts not removed by improving the quality of data, for example the irregularities from dynamic transforms and other artifacts from misestimated interpolation time, need to be reduced through additional measures. The possibilities of applying force smoothing in the transitions between interpolated frame intervals to reduce the artifacts, and its impact on the feedback quality are examples of future interests.

The most important future work in this project, however, is to investigate the back-tracking algorithm and alternatives, and find ways to guarantee graceful failure when no single solution is available. In the current implementation we address this by checking the calculated estimate and the initial estimate against equation 27 and choosing the best match. Our worst case scenario, the initial estimate, corresponds to ignoring the movement of the proxy point since the last estimation when reading off the movement vector.

The haptic support for time-varying data require only the data of the current data frame interval and an explicit description of the motion of local features from the current frame to the next. These data can also be provided by real-time simulation, allowing for dynamic volume haptics from, for example, interactive and volumetric tissue deformation. There are several issues in this process that we will examinate in our future work, for example motion field estimation and how to handle a delay in the data generation.

IX. Conclusions

Volume visualization is a powerful tool for exploring scientific data of a dynamic nature. Haptic feedback for volumetric data, capable of enhancing speed, accuracy and information flow in volume exploration, has been lacking methods for representing time-varying properties of dynamic data. This article has presented a technique to enable haptic feedback

Fig. 17. A frame from the animated CT sequence of a beating human heart. The haptic pen is probing the moving heart wall.

that reflects changes in volumetric data, which allows the introduction of haptic feedback in the exploration of dynamic behaviour in scientific visualization, and feedback consistent with changes in transforms. We have shown that the algorithm is effective in handling both dynamic transforms and animated volume data, providing smooth haptic interpolation and stable and predictable feedback, while still allowing for the required 1 kHz haptic update rate. Furthermore, we have shown that the method can be implemented in a way so that no additional considerations are needed when implementing haptic modes. New modes can be developed for static data sets and then be readily deployed on time-varying data.

Acknowledgements

This work has been supported by the Foundation for Strategic Research (SSF) under the Strategic Research Center MOVIII. The staff at the Center for Medical Image Science and Visualization (CMIV) at Linköping University are gratefully acknowledged for providing high quality data sets.

References

[1] Steven Wall and William Harwin, "Quantification of the effects of haptic feedback during a motor skills task in a simulated environment," in *Proceedings of Phantom User Research Symposium*, 2000.

[2] Arthur E. Kirkpatrick and Sarah A. Douglas, "Application-based evaluation of haptic interfaces," in *Proceedings of the 10th Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems*, 2002.
[3] P. J. Passmore, C. F. Nielsen, W. J. Cosh, and A. Darzi, "Effects of viewing and orientation on path following in a medical teleoperation environment," in *Proceedings of IEEE Virtual Reality*, 2001.
[4] Richard J. Adams, Daniel Klowden, and Blake Hannaford, "Virtual training for manual assembly task," *Haptics-e, The Electronic Journal of Haptics Research (www.haptics-e.org)*, vol. 2, no. 2, October 2001.
[5] Steven A. Wall, Karin Paynter, Ann Marie Shillito, Mark Wright, and Silvia Scali, "The effect of haptic feedback and stereo graphics in a 3D target acquisition task," in *Proceedings of Eurohaptics*. University of Edinburgh, United Kingdom, 2002.
[6] H. Iwata and H. Noma, "Volume haptization," in *Proceedings of IEEE 1993 Symposium on Research Frontiers in Virtual Reality*, October 1993, pp. 16–23.
[7] R. S. Avila and L. M. Sobierajski, "A haptic interaction method for volume visualization," in *Proceedings of IEEE Visualization*, October 1996, pp. 197–204.
[8] Dale A. Lawrence, Lucy Y. Pao, Christopher D. Lee, and Roman Y. Novoselov, "Synergistic visual/haptic rendering modes for scientific visualization," *IEEE Computer Graphics and Applications*, vol. 24, no. 6, pp. 22–30, 2004.
[9] Karljohan Lundin, Anders Ynnerman, and Björn Gudmundsson, "Proxy-based haptic feedback from volumetric density data," in *Proceedings of Eurohaptics*. University of Edinburgh, United Kingdom, 2002, pp. 104–109.
[10] Milan Ikits, J. Dean Brederson, Charles D. Hansen, and Christopher R. Johnson, "A constraint-based technique for haptic volume exploration," in *Proceedings of IEEE Visualization, pp. 263–269*, 2003.
[11] Erik Vidholm, Xavier Tizon, Ingela Nyström, and Ewert Bengtsson, "Haptic guided seeding of MRA images for semi-automatic segmentation," in *Proceedings of IEEE International Symposium on Biomedical Imaging*, 2004.
[12] Karljohan Lundin, Mattias Sillén, Matthew Cooper, and Anders Ynnerman, "Haptic visualization of computational fluid dynamics data using reactive forces," in *Proceedings of Conference on Visualization and Data Analysis, part of IS&T/SPIE Symposium on Electronic Imaging*, San Jose, CA USA, January 2005, pp. 31–41.
[13] Karljohan Lundin, Björn Gudmundsson, and Anders Ynnerman, "General proxy-based haptics for volume visualization," in *Proceedings of the IEEE World Haptics Conference*, Pisa, Italy, March 2005, pp. 557–560.
[14] Diego C. Ruspini, Krasimir Kolarov, and Oussama Khatib, "The haptic display of complex graphical environments," *Computer Graphics*, vol. 31, no. Annual Conference Series, pp. 345–352, 1997.
[15] C. B. Zilles and J. K. Salisbury, "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*, 1995, vol. 3, pp. 146–151.
[16] A. Mor, S. Gibson, and J. Samosky, "Interacting with 3-dimensional medical data: Haptic feedback for surgical simulation," in *Proceedings of Phantom User Group Workshop*, 1996.
[17] Dirk Bartz and Özlem Gürvit, "Haptic navigation in volumetric datasets," in *Proceedings of PHANToM User Research Symposium*, 2000.
[18] Karljohan Lundin, Matthew Cooper, Anders Persson, Daniel Evestedt, and Anders Ynnerman, "Enabling design and interactive selection of haptic modes," *Virtual Reality*, 2006, DOI: 10.1007/s10055-006-0033-7.
[19] Susan J. Lederman and Roberta L. Klatzky, "Hand movements: A window into haptic object recognition," *Cognitive Psychology*, vol. 19, no. 3, pp. 342–368, July 1987.
[20] Jean-Philippe Thirion, "Image matching as a diffusion process: an analogy with Maxwell's demons," *Medical Image Analysis*, vol. 2, no. 3, pp. 243–260, 1998.
[21] Luis Ibanez, Will Schroeder, Lydia Ng, and Josh Cates, *The ITK Software Guide: The Insight Segmentation and Registration Toolkit*, Kitware Inc, 2003.
[22] Samuel M. Song and Richard M. Leahy, "Computation of 3-D velocity fields from 3-D cone CT images of a human heart," *Transactions on Medical Imaging*, vol. 10, no. 3, pp. 295–306, September 1991.
[23] Ivana Mikić, Slawomir Krucinski, and James D. Thomas, "Segmentation and tracking in echocardiographic sequences: Active contours guided by optical flow estimates," *Transactions on Medical Imaging*, vol. 17, no. 2, pp. 274–284, April 1998.
[24] Samuel M. Song, Richard M. Leahy, Douglas P. Boyd, Bruce H. Brundage, and Sandy Napel, "Determining cardiac velocity fields and intraventricular pressure distribution from a sequence of ultrafast CT cardiac images," *Transactions on Medical Imaging*, vol. 13, no. 2, pp. 386–397, June 1994.

Karljohan Lundin Palmerius received his M.Sc. in Media Technology and Engineering in 2001 and his Ph.D. in scientific visualization in 2007. Since 2002 he has been involved in research projects in visualization, haptics and haptic visualization at Norrköping Visualization and Interaction Studio, Linköping University. During this time he has developed algorithms and concepts for the implementation of haptic interaction in several different disciplines, with particular focus on volumetric data.

Matt Cooper received his B.Sc. in Theoretical Physics from the University of Exeter in 1986, and his M.Sc. in Computer Science and Ph.D. in Theoretical Chemistry from the University of Manchester in 1987 and 1990, respectively. During the early 90's his postdoctoral research focussed primarily on high performance and parallel computing techniques for 'Grand Challenge' problems in the Physical Sciences but in 1996 he joined the Manchester Visualization Centre to focus more on the application of visualization and computer graphics methods across many disciplines. In 2001 he moved to the NVIS group in Linköping University's department of Science and Technology, where he has continued this work. His current research interests are in the visualization of large, time-varying and multivariate data, particularly through the application of virtual and augmented reality technologies to these problems.

Professor Anders Ynnerman received a Ph.D. in physics from Gothenburg University. During the early 90s he was doing research at Oxford University, UK, and Vanderbilt University, USA. In 1996 he started the Swedish National Graduate School in Scientific Computing, which he directed until 1999. From 1997 to 2002 he directed the Swedish National Supercomputer Centre and from 2002 to 2006 he directed the Swedish National Infrastructure for Computing (SNIC). Ynnerman is representing Sweden and the Nordic region in several international collaborations and policy bodies.

Since 1999 he is holding a chair in scientific visualization at Linköping University and In 2000 he founded the Norrköping Visualization and Interaction Studio (NVIS). NVIS currently constitutes one of the main focal points for research and education in computer graphics and visualization in the Nordic region. Ynnerman's current research interest lies in the area of visualization of large scale and complex data sets with a focus on volume rendering and multi-modal interaction.