User-Centric Viewpoint Computation for Haptic Exploration and Manipulation

Miguel A. Otaduy and Ming C. Lin
Department of Computer Science
University of North Carolina at Chapel Hill
{otaduy,lin}@cs.unc.edu
http://www.cs.unc.edu/∼geom/HView

Abstract: We present several techniques for user-centric viewing of the virtual objects or datasets under haptic exploration and manipulation. Depending on the type of tasks performed by the user, our algorithms compute automatic placement of the user viewpoint to navigate through the scene, to display the near-optimal views, and to reposition the viewpoint for haptic visualization. This is accomplished by conjecturing the user's intent based on the user's actions, the object geometry, and intra- and inter-object occlusion relationships. These algorithms have been implemented and interfaced with both a 3-DOF and a 6-DOF PHANToM arms. We demonstrate their application on haptic exploration and visualization of a complex structure, as well as multiresolution modeling and 3D painting with a haptic interface.
CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques - Interaction Techniques, Visualization

1 Introduction

Three-dimensional (3D) interaction has been explored in computer graphics, virtual reality (VR), user interface and scientific visualization. A number of techniques for 3D interaction have been developed, including object selection [PFC+97a], flying, grabbing and manipulating [RH92], worlds in miniature [PBBW95], combination of different modes of speech, gesture and gaze at the interface to allow real-time interaction with a graphics display, two-handed interaction [ABF+97, CFH97], and exploiting proprioception [MBS97]. Among them, haptic visualization, as an augmentation to visual display, has the potential to further increase the understanding of complex datasets by enabling another modality of communication [AS96, Bur96, Che99, DMW+98, Gib95, IN93, LLPN00, MPT99].

3D manipulation of virtual objects or massive datasets can be simplified if the viewing of interaction is presented using multiple 2D views. This technique is commonly used in many commercial CAD/CAM and data visualization software systems. Typically the mouse cursor is free to move among the three or more views and constrained to act in only the two dimensions shown in each view. This requires the user to synthesize multiple 2D views to visualize the manipulated objects or datasets. For most of the work in VR, 3D interaction techniques and haptic visualization, the user's viewpoint remains fixed at the same location, unless the user is head- or eye- tracked or has a dedicated input device (e.g. joystick or spaceball) to specifically indicate the direction of travel.

Viewpoint locations have a direct impact on the quality of the resulting graphical display accompanying the haptic visualization. However, to the best of our knowledge, the issues related to determining appropriate viewpoints in the graphical display that will be most suitable for haptic exploration and manipulation tasks have not been addressed. The disconnection between the graphical display and haptic visualization can result in poor or inconsistent presentation of information. For example, as the haptic probe moves between time steps while the viewpoint remains fixed, the probe may be occluded by other objects and not be visible from the viewpoint (as shown in Fig. 1). A new camera position needs to be computed, in order to properly view the neighborhood under haptic exploration.

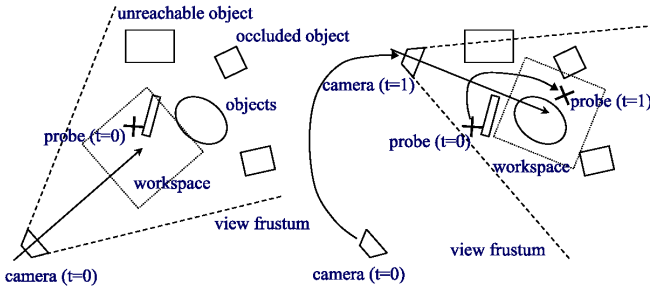

Figure 1: As the haptic probe moves from its location at $t = 0$ to the new location at $t = 1$, it becomes occluded by the ellipsoidal object, viewing from the camera position at $t = 0$. In order to properly view the probe and its nearby neighborhood, the camera must be repositioned at $t = 1$.

1.1 Main Contribution

In this paper, we present algorithms for *user-centric* viewing of 3D virtual objects or datasets for haptic exploration and manipulation, by taking into account the user's intention in determining the viewpoint locations. In addition to using the force feedback device for haptic rendering, we also use it *implicitly* as a mechanism for the users to express

their intent for the viewpoint location in graphical display, while *simultaneously* performing force display. Our algorithms generate automatic placement of the user viewpoint to navigate through the scene, to display the near-optimal views, and to reposition the viewpoint. This is accomplished by conjecturing the user's intent based on the user's actions, the object geometry, as well as inter-object and intra-object occlusion relationships. Our viewpoint computation techniques offer the following advantages:

- Simple 3D interface to reset the view location for haptic visualization;
- Proper viewing of the interesting events (e.g. inter-object interaction) in the scene or explored salient features of the datasets at all time;
- Automatic adjustment of the viewpoint locations without having the user to switch between haptic manipulation and camera repositioning.

1.2 Organization

The rest of the paper is organized as follows. Section 2 gives a brief survey of related work. Section 3 presents an overview of our approach. Our algorithm for view navigation of large, complex scenes for haptic exploration and visualization is explained in section 4. Section 5 discusses techniques to resolve the object-object occlusion problems and visibility issues in choosing near-optimal views. Section 6 presents a mechanism to automatically reposition the viewpoint for better viewing of manipulation results. Section 7 describes our prototype implementation and demonstrates their application on 6-DOF haptic interaction of complex structures, as well as haptic modeling and 3D painting.

2 Related Work

Camera control is a fundamental problem for 3D graphics applications. Several techniques on user interfaces for camera control have been proposed, including orbiting techniques mapping 2D motion into 3D interaction [CMS88, PBG92, Wer94, ZF99], use of image plane constraints [GW92, PFC+97b], and direct camera manipulation using a 6DOF input device [WO90]. Our approach differs from many of the existing techniques on using 2D input devices to directly manipulate the viewpoint. It shares some similarity with [MCR90] on moving the viewpoint toward a point of interest. However, our main focus here is to achieve automatic placement of viewpoint via implicit control based on user's manipulation of the haptic device. We also consider issues related to visibility and occlusion between objects.

Viewpoint computation is related to the problem of camera placement in computer vision, including sensor planning [TAT95], view planning [Pit99], and purposive viewpoint [KD95]. These problems have different task requirements (e.g. placement of multiple sensors and cameras) and performance constraints (e.g. the execution can be carried out offline). This problem is also important to image-based rendering [CL96, MB95], where the viewpoint computation is crucial to model acquisition and scene modeling.

Our viewpoint computation problem can be considered as an inverse of the sensor planning problems in computer vision. Given the user's actions in exploring the complex environment or datasets, i.e. sequences of the input positions and orientations of the haptic device (robot arm used in reverse), we attempt to compute the near-optimal viewpoint for viewing the manipulated objects and/or features on the datasets. In contrast, camera placement for sensor planning takes a *constraint based* description of the vision task to be performed and synthesizes a "generalized viewpoint", which incorporates sensor location, orientation and lens parameters. Using the geometric description of the workspace, it is sometimes possible to construct occlusion-free visibility regions for viewing a particular feature of a given model. This region can then be used to find the best viewpoint for planning and accomplishing a certain vision task [Tar91].

Furthermore, the optimization process to compute the best viewpoint for performing certain tasks can often be highly non-linear and take at least several seconds upto a few minutes to perform the computation. In addition, it sometimes requires human assistance to provide good starting points. This is acceptable for one-shot off-line planning, but not suitable for interactive applications, such as haptic visualization and manipulation of complex datasets and objects at KHz rates.

The viewpoint computation problem we address in this paper is also somewhat related to the art gallery problem [O'R87] in computational geometry where the number and positions of watchmen in a 2D polygonal environment need to be determined. But, here we are concerned with a 3D environment and the placement of a single viewpoint. These requirements introduce a much higher combinatorial complexity than the classic art gallery problem.

3 Preliminaries

3.1 Terminology

The position and directions are expressed using vector notation in bold face symbols. We use the quaternion algebra [Sho87] to specify the relative orientation and transformation from one coordinate system to another.

The viewpoint (or the camera setting) is characterized by the following parameters: (a) **e**: the location of the viewpoint; (b) **c**: the location where the viewpoint is looking at; (c) **u**: the up vector that determines the pan and tilt angles along the viewing vector determined by **e** and **c**.

We use the term "probe object" for describing the object which the probe picks up, and "target object(s)" for referring to the object(s) that the probe object or the probe is interacting with.

3.2 Overview of Our Approach

Given the haptic inputs (i.e. positions & orientations of the probe) from the user and the scene description, we wish to determine the viewpoint location that provides near-optimal viewing of the "region of interest". The region of interest is changing dynamically and normally refers to the areas where the haptic manipulation is taking place or where the salient features of the datasets are under exploration. Optimality in this problem is measured in terms of object-object and self occlusion, given the user controlled focus distance and field-of-view of the camera.

We have designed several viewing algorithms that automatically compute the new viewpoint location based on the type of haptic tasks performed on the scene or the datasets. Our viewpoint computation techniques take into considera-

tion the followings:

- user's intention based on his/her manipulation of the haptic probe;
- the object geometry and the scene description;
- intra- and inter- object occlusion;
- camera's field-of-view;
- camera's focus distance.

Given the haptic task, the viewpoint computation techniques we have developed include:

1. *View navigation* which provides the capability to recompute the viewpoint based on user's exploration of massive datasets (section 4);
2. *Near-optimal views* of the regions that the user is interacting with (section 5). This is applicable to all types of haptic exploration and manipulation;
3. *Automatic repositioning* of the viewpoint for any type of haptic interaction (section 6).

Next, we will describe each approach in detail.

4 View Navigation

In this section we present a view navigation technique for haptic exploration of massive datasets. Unlike traditional viewing systems for haptic visualization, the goal of this viewing functionality is to focus on the regions of potential interest in the virtual scene, as the user haptically explores different parts of the massive datasets in the environment.

In the typical viewing system for force display without head-tracking, as the user moves the haptic probe, the viewpoint normally stays fixed in the same location. The user often does not have the best viewing of the neighborhood where the haptic probe is exploring. We propose to adaptively change the viewpoint based on user's haptic exploration of the object(s) or datasets in a large, complex scene. The motion of the haptic probe provides excellent hints of the user's intentions. As the position and orientation of the haptic probe are changed based on user's gestures, viewing of the scene can be adapted to display the neighborhood that the virtual probe is interacting with, while preserving view coherence with the probe motion in the workspace.

The basic idea for our view navigation techniques is to apply appropriate transformations, taking into account the position of the haptic probe relative to a given reference point. Then, we attach the camera at the pre-defined offset distance (characterized by the camera parameters as explained in section 3) from the haptic probe to ensure proper viewing of the regions under haptic exploration. The position and orientation of the haptic device at each time instance are used as "velocity commands" for the virtual probe. The transformation between the local reference system and the global reference system is modified taking into account the velocity commands.

In a static situation, the motion of the virtual probe is confined to a virtual workspace. We use user's gestures to modify the position and orientation of that virtual workspace in the entire virtual world, thus allowing complete visual and haptic accessibility. Three invariants must be maintained

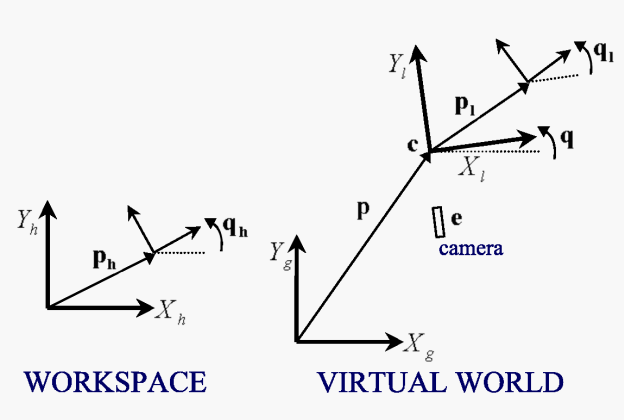

Figure 2: 2D workspace (LEFT), global and local reference frames in the virtual world (RIGHT).

in order to preserve coherence between the operations performed in the real workspace and the visual and haptic feedback:

1. The position and orientation of the virtual probe in the local reference system of the virtual workspace must mimic the position and orientation of the real haptic probe;
2. The camera must be anchored in the local reference system;
3. Force and torque must be displayed in local coordinates.

INVARIANT 1: aligning the local reference system with the real workspace

Given the position, $\mathbf{p_h}$, and orientation, $\mathbf{q_h}$ (expressed as a quaternion), of the haptic device in its workspace (see Figure 2), we compute the position, $\mathbf{p_g}$, and orientation, $\mathbf{q_g}$, of the virtual probe in the global coordinates of the virtual world. The first step is to align the position, $\mathbf{p_l}$, and orientation, $\mathbf{q_l}$, of the virtual probe in its local reference frame with the state of the haptic device in the workspace. That is,

$$\mathbf{p_l} = \mathbf{p_h},$$

$$\mathbf{q_l} = \mathbf{q_h}.$$

Next, the state of the haptic probe in the global reference system of the virtual world is computed. We denote the transformation from the local coordinates to the global coordinates with $\mathbf{p}$ for the translation and $\mathbf{q}$ for the rotation. The conjugate of $\mathbf{q}$ is expressed as $\mathbf{q}'$.

$$\mathbf{p_g} = \mathbf{p} + \mathbf{q} * \mathbf{p_l} * \mathbf{q}',$$

$$\mathbf{q_g} = \mathbf{q} * \mathbf{q_l}.$$

Computing the transformation between local and global reference systems

The velocity command is computed in terms of the difference between the current state of the haptic probe in its local reference frame (or the *local state*) and a *virtual base state*

($\mathbf{p_b}$ and $\mathbf{q_b}$). Emulating the operation modes of a joystick, we distinguish two possible modes that the user can select to manipulate the haptic probe and view the scene: $position$ mode and $velocity$ mode. In the position mode, the virtual base state is updated every frame and matches the current local state, so the velocity command is zero.

$$\mathbf{p_b} = \mathbf{p_l},$$

$$\mathbf{q_b} = \mathbf{q_l}.$$

In the velocity mode, the virtual base state remains static. The sub-indices i and $i-1$ indicate values of the variables at the current time step and at the previous time step.

$$\mathbf{p_{b,i}} = \mathbf{p_{b,i-1}},$$

$$\mathbf{q_{b,i}} = \mathbf{q_{b,i-1}}.$$

This implies that the base state "travels" along with the local state until we switch from the position mode to the velocity mode. At that moment the base will be anchored in the local reference system, and the difference between the local state and the virtual base state will set the velocity. This velocity is integrated at the current time step to yield an incremental transformation, $\Delta\mathbf{q}$ and $\Delta\mathbf{p}$. First, the rotation transformation is computed. Based on empirical observation, we conclude that the motion is more intuitive if the angular velocity is only allowed along the x and y axes (using the right-hand rule, y^+ being the positive vertical direction and z^+ coming out from the 2D workspace).

The incremental rotation is set as YX Euler angles and is applied in local coordinates. Let k_1 and k_2 be adjustable gain values, we have:

$$\Delta\mathbf{q} = \mathbf{q_y} * \mathbf{q_x},$$

$$\mathbf{q_y} = Rot(Y, k_1 * (Y(\mathbf{q_l}) - Y(\mathbf{q_b}))),$$
$$\mathbf{q_x} = Rot(X, k_2 * (X(\mathbf{q_l}) - X(\mathbf{q_b}))),$$
$$\mathbf{q_i} = \mathbf{q_{i-1}} * \Delta\mathbf{q}.$$

where $X(\mathbf{q})$, $Y(\mathbf{q})$ and $Z(\mathbf{q})$ is the XYZ Euler angle representation of the quaternion $\mathbf{q}$ and $Rot(V, \theta)$ denotes rotation θ about the vector V.

The incremental translation is transformed from the local to the global coordinates, to make it coherent with the user's view of the scene. Let k_3 be an adjustable gain value.

$$\Delta\mathbf{p} = k_3 * \mathbf{q} * (\mathbf{p_l} - \mathbf{p_b}) * \mathbf{q}',$$

$$\mathbf{p_i} = \mathbf{p_{i-1}} + \Delta\mathbf{p}.$$

In practice, the incremental transformation is only applied if the difference between the local state and the virtual base state is larger than a pre-defined threshold, and it is also clamped to a maximum value.

INVARIANT 2: setting the camera

After transforming the position and orientation of the virtual probe, we transform the settings of the camera and the output forces (and torque) so that coherence is maintained between the operations performed in the workspace of the device and their image in the local reference system of the virtual scenario. As described in section 3, the camera is characterized by means of its location, $\mathbf{e}$, where it is looking at, $\mathbf{c}$, and an up vector, $\mathbf{u}$, to define the pan or tilt angle. Every frame they are defined with the initial settings ($\mathbf{c_0}$, $\mathbf{e_0}$, $\mathbf{u_0}$):

$$\mathbf{c} = \mathbf{p} + \mathbf{q} * \mathbf{c_0} * \mathbf{q}',$$

$$\mathbf{e} = \mathbf{p} + \mathbf{q} * \mathbf{e_0} * \mathbf{q}',$$

$$\mathbf{u} = \mathbf{q} * \mathbf{u_0} * \mathbf{q}'.$$

Given this mathematical formulation, the viewpoint is naturally controlled by the haptic probe to ensure proper viewing of the regions under haptic exploration.

INVARIANT 3: transforming the forces

The forces and the torque are computed in the global coordinates, and they have to be displayed in local coordinates:

$$\mathbf{F_l} = \mathbf{q}' * \mathbf{F_g} * \mathbf{q},$$

$$\mathbf{T_l} = \mathbf{q}' * \mathbf{T_g} * \mathbf{q}.$$

5 Near-Optimal Views

As the viewpoint moves with the haptic probe using view navigation described in section 4, it is possible that one object may occlude the view of another in the scene, or the visibility of the region of interest degrades due to self-occlusion. In this section, we describe a technique to compute near-optimal views for haptic interaction and automatically provide the user with a proper view of the region of interest at every moment. This enables the user to manipulate target objects easily, without explicitly resetting the viewpoint location.

To present near-optimal views, we use two cameras to view the scene: the main camera and the detail camera. The main camera is typically positioned using the view navigation technique described in section 3; while the detail camera attempts to provide near-optimal views of the regions on the target object(s) that the probe is interacting with. An example is shown in Color Plate III and the corresponding views are shown in Color Plate IV.

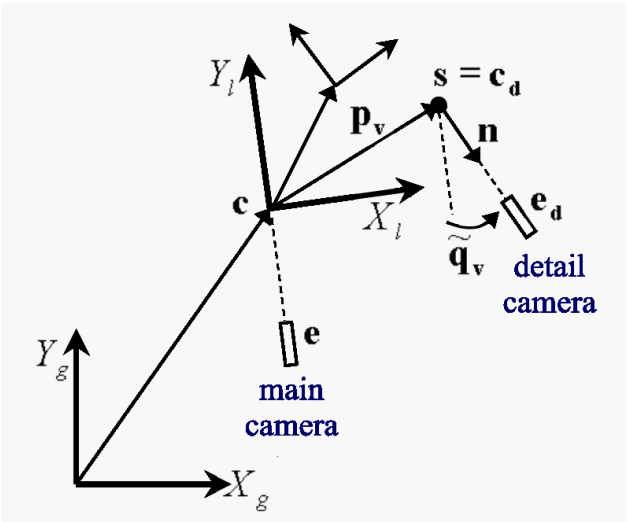

Figure 3: Placement of the cameras for near-optimal view.

We characterize the detail camera for projecting the near-optimal views in terms of a rotation $\mathbf{q_v}$ and a translation $\mathbf{p_v}$ from the main camera (see Figure 3).

The first step in placing the detail camera is the computation of the view center $\mathbf{c_d}$. Given an object touched by the haptic probe, we characterize the region of interest with a contact point, $\mathbf{s}$, and a contact outward normal, $\mathbf{n}$. We use penalty methods for the haptic rendering, and the contact normal is defined as the direction between the closest points in the virtual probe and the target object. The contact point is chosen to be the centroid of the contact area, when the probe is in contact with the object. When the haptic probe is free to move around, we set the contact point as the point on the target object that is closest to the probe. The contact point is selected as the view center. Then the translational vector from the main camera to the detail camera is expressed as:

$$\mathbf{p_v} = \mathbf{s} - \mathbf{c}$$

Next, we need to compute the center of projection or the viewpoint, $\mathbf{e_d}$. This is a problem with 3 degrees of freedom. A viewpoint location will be optimal if the contact region is visible from the user's view, at a user defined distance and at some offset angle from the normal direction. Figure 4 shows schematically the location for the viewpoint of the near-optimal view in a scenario where the probe is a cylindrical object.

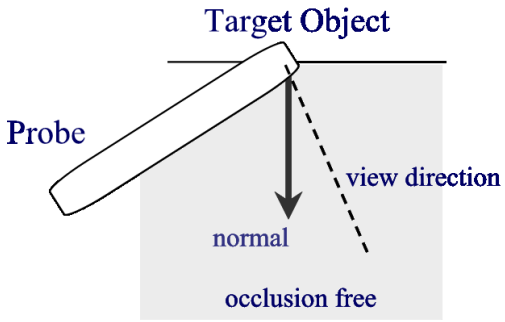

Figure 4: Optimal view direction for scenarios with cylindrical probes and no self-occlusion of the target object.

In some scenarios this problem may not have an optimal solution, due to occlusion between the target object and the probe object (or the probe itself). In such cases, we may need to zoom in arbitrarily close to the contact point to minimize the amount of occlusion. Furthermore, an optimal solution may require global visibility information, such as an aspect graph. However, its computation has a runtime complexity of $O(n^6)$, where n is the number of primitives (such as polygons). For most scenarios, it may not be possible at all to obtain the optimal solutions at interactive rates. Therefore, we do not attempt to compute a globally optimal view, but rather one that locally minimizes the amount of occlusion. We allow the user to interactively select the viewing distance r, thus adding one constraint to the problem. The camera can be positioned at any point on the surface of a sphere centered at the contact point with a radius of r. Assuming that our intention is to mainly visualize the target object, the location of the camera should be constrained to a cone around the contact normal. The angle of this cone is bounded to avoid tangential views of the contact location, which would not provide depth information (see Figure 5).

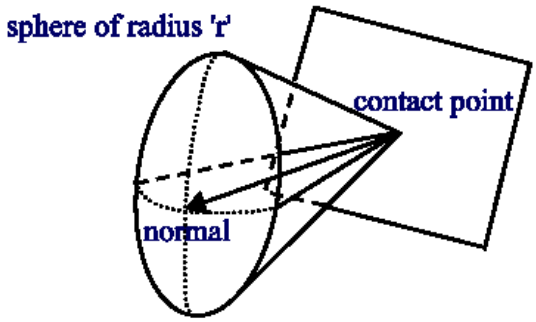

Figure 5: Location of the camera is constrainted to the surface of a sphere centered at the contact point with a radius r and to a cone around the contact normal.

The operation of placing the camera is decomposed into two steps: (a) an initial transformation where we place the camera along the contact normal, and (b) a local optimization that searches for the appropriate location. We call $\tilde{\mathbf{q}}_\mathbf{v}$ the "initial rotation". In local coordinates, the main viewing direction is aligned with the z axis, and the up vector is aligned with the y axis. The rotation from the main viewing direction to the normal direction of the contact can be found easily in local coordinates. This rotation $\mathbf{q_n}$ is computed as the composition of two rotations around the x and y axes. The rotation is completed transforming back to the global coordinates:

$$\tilde{\mathbf{q}}_\mathbf{v} = \mathbf{q} * \mathbf{q_n} * \mathbf{q}'.$$

The local optimization is based on the amount of occlusion from a successively refined camera position. The portions of the target and probe objects in front of the contact point are rendered separately from the current location of the detail camera. We compute amounts of overlap in two orthogonal directions in screen-space [ZMHH97]. The camera is repositioned along those two directions following a bisection scheme, until no further reduction in the amount of occlusion is obtained. We are considering different approaches, which compute the amount of occlusion rendering the objects from the view center c_d to reduce the number of rendering and readback operations. Eventually we obtain the rotation, $\mathbf{q_v}$, from the main camera to the detail camera.

The region of interest can change suddenly, as well as the optimal placement of the camera to avoid occlusions. The translation and rotation to be applied to the camera are filtered to smooth these discontinuities. After motion smoothing, we can compute the settings for the detail camera by:

$$\mathbf{c_d} = \mathbf{c} + \mathbf{p_v},$$

$$\mathbf{u_d} = \mathbf{q_v} * \mathbf{u} * \mathbf{q_v}',$$

$$\mathbf{e_d} = \mathbf{c_d} + r * \mathbf{q_v} * \frac{\mathbf{e} - \mathbf{c}}{\|\mathbf{e} - \mathbf{c}\|} * \mathbf{q_v}'.$$

where r is the distance between the contact point and the viewpoint.

Once the camera placement is computed, the scene is then rendered. Since occlusion might be impossible to avoid, we use transparency to make the neighborhood about the region of interest visible at every moment. First the target object is rendered, clipped with a view pyramid with its apex at the eye and its base at the contact point. This makes portions of the target object in front of the contact point visible within a "visibility window" to ensure the contact region is visible regardless of the occlusion. The size of the view pyramid can be set in terms of the size of the visibility window and the distance from the center of projection to the contact point. The next operation is to render the probe object and the rest of the target object blended. We finally render the rest of the objects blended as well, to avoid possible occlusions within the region of interest.

6 Automatic Repositioning

In this section, we describe a viewing technique for automatically repositioning the viewpoint for any type of haptic tasks. During the haptic interaction, the region of interest or task space may become difficult to discern from the main view. And, the near-optimal view is intended as an aid for better viewing during haptic manipulation. However, due to the lack of coherence between the motion in the detail view and the motion in the workspace of the haptic device, the resulting graphical display may be misleading if the user tries to perform the operations based on the view projected on the detail camera.

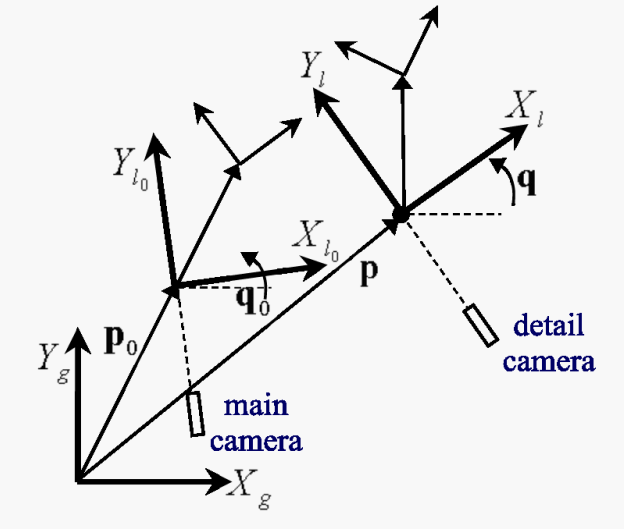

Figure 6: Automatic repositioning of the viewpoint and the virtual probe.

To solve these problems, we have designed a viewing functionality, where the main camera is progressively adapted to the location and orientation of the detail camera, allowing the user to automatically have a near-optimal and coherent vision of the task space (see Figure 6). In practice, this is carried out by composing the transformation from local to global coordinates with the transformation from the main camera to the detail camera. For the rotation, we use the initial rotation, $\tilde{\mathbf{q}}_\mathbf{v}$, instead of the actual rotation, because it is independent of the position and orientation of the virtual probe.

Under the user's command to carry out automatic repositioning (e.g. pressing a button), the current transformation is stored as $\mathbf{p_0}$ and $\mathbf{q_0}$. The target is set to be the desired transformation, and linear interpolation is computed, updating at every step the weight of each term.

$$\mathbf{q} = (1-t) * \mathbf{q_0} + t * \tilde{\mathbf{q}}_\mathbf{v} * \mathbf{q_0},$$

$$\mathbf{p} = (1-t) * \mathbf{p_0} + t * (\mathbf{p_v} + \mathbf{p_0}).$$

7 Implementation and Results

In this section we describe the platforms and test scenarios where we have applied our algorithms, as well as the results that we have been to able to achieve. We used two different haptic devices: the 6-DOF PHANTOM Premium 1.5 and the 3-DOF PHANTOM Desktop device, both designed by SensAble Technologies, Inc.

We have integrated our algorithms with two applications: six degree-of-freedom haptic rendering of polygonal models [GME+00] and an interactive multiresolution modeling and 3D painting with haptic interface [GEL00]. The latter was particularly valuable to test the performance of the automatic repositioning during a 3D painting or modeling operation, as opposed to the traditional technique of grabbing and repositioning the manipulated object. Using the framework for 6-DOF haptic rendering of arbitrary polygonal models we were able to test our view navigation algorithm and the near-optimal view computation. The algorithm for view navigation was tested on the haptic exploration of a massive model, the Auxiliary Machine Room (AMR) of a submarine. In order to test the algorithm for near-optimal view computation, we used the model of a digestive tract, which has a notably irregular surface, and a CAD model, which shows clear examples of self-occlusion.

Please see the accompanied MPEGs that demonstrate the application of our algorithms on some test scenarios. The video clips are also available at **http://www.cs.unc.edu/∼geom/HView**.

7.1 View Navigation

The AMR model consists of nearly half a million polygons and 3,000 parts. For the haptic exploration of the AMR we scaled the position of the haptic device in such a way that with the workspace of the haptic device we could only cover $1/200$ of the length of the AMR. This is a reasonable ratio for properly testing haptic exploration of the objects in the scenario using our view navigation algorithm. In Color Plate I we show a view of the haptic probe navigating along the AMR. In Color Plate II, we show the bird's eye view and the corresponding camera and probe positions for the situation in Color Plate I. View navigation is necessary if we want to be able to visualize all the objects in the scenario.

In this test scenario, the haptic device is efficiently used as an input device for tracking the intentions of the user, in addition to being a force feedback device. Both the haptic probe and the viewer follow the motion of the user's hand in a natural way.

7.2 Near Optimal View

The model of the digestive tract (Color Plate III) is composed of approximately 53,000 triangles. Its surface is irregular at some places. This implies that the optimal viewing direction for perceiving surface details can change noticeably as the haptic probe traverses the surface.

Color Plate V shows the detail views when touching a point at the duodenum with different orientations of the probe. A change in the orientation of the probe moves the detail camera to a new location, in search of a lateral and occlusion-free view of the region of interest (i.e. the contact point).

There are situations when object-object or self occlusion cannot be avoided. Also, in the transition between two locations for the near-optimal views, occlusion may occur, such as in the situation where the haptic probe explores the inner side of the CAD model (See Figure 7). The viewpoint location for the near-optimal views suddenly jumps trying to avoid self-occluding parts. Therefore, during the transition there is a lapse of time during which the contact point and the probe are occluded. In Figure 7 we show how this problem is handled by rendering the objects with different levels of opacity. The region of interest becomes visible, yet we still have sufficient information about the part of the object that occludes the contact neighborhood.

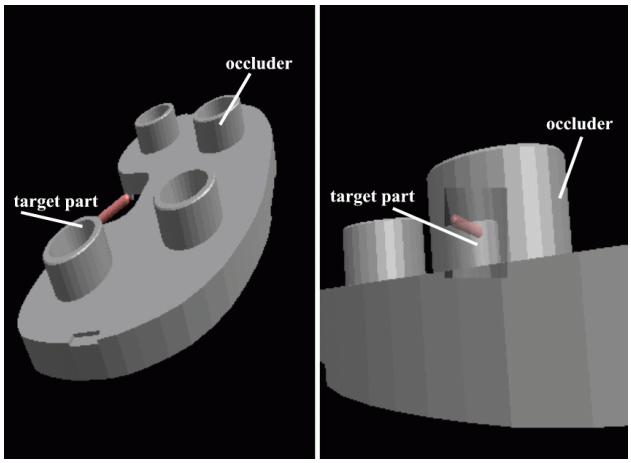

Figure 7: Rendering with different levels of opacity to avoid occlusion in the transition between two near-optimal views. Main view (left) and Detail view (right)

7.3 Automatic Repositioning

From the test scenarios using our interactive 3D multiresolution modeling and painting system, the users found that automatic repositioning is much easier to use than the tedious technique of grabbing the manipulated object and repositioning it whenever they want to modify parts of the object that are not visible.

During the manipulation task, the user can select a location on the target object at any time, make it the center of the workspace, and the viewpoint is automatically repositioned. As an example, we show two different views of an object after having repositioned the main view to move a point A in Color Plate VI and a point B in Color Plate VII to the center of the workspace. As it can be inferred from the effect of the lighting, in both cases the normal of the object at those points has become the viewing direction.

8 Summary and Conclusions

We have presented several simple yet effective techniques for adaptively recomputing user-centric viewpoint locations based on the user's intentions, for haptic exploration and manipulation. These viewing techniques also take into consideration the object geometry, occlusion and visibility issues and camera parameters. They enable the users to better view the manipulated subjects during haptic interaction with complex structures or datasets, without head- or eye-tracking and additional 3D input devices. For future research, we plan to apply our techniques to higher dimensional volumetric datasets.

Acknowledgement

This research is supported in part by a fellowship of the Government of the Basque Country, NSF DMI-9900157, NSF IIS-9821067, ONR N00014-01-1-0067 and Intel. We would like to thank Stephen Ehmann for his help on integrating collision detection libraries, SWIFT and SWIFT++, with our 6-DOF haptic rendering algorithm. We are also grateful to Dinesh Manocha and the anonymous reviewers for their feedback on the earlier drafts of this paper, and Vincent Scheib for assisting with video editing.

References

[ABF+97] Maneesh Agrawala, Andrew C. Beers, Bernd Fröhlich, Pat Hanrahan, Ian McDowall, and Mark Bolas. The two-user responsive workbench: Support for collaboration through independent views of a shared space. *SIGGRAPH 97 Conference Proceedings*, pages 327–332. 1997.

[AS96] R. S. Avila and L. M. Sobierajski. A haptic interaction method for volume visualization. *Proceedings of Visualization'96*, pages 197–204, 1996.

[Bur96] G. Burdea. *Force and Touch Feedback for Virtual Reality*. John Wiley and Sons, 1996.

[CFH97] L. Cutler, B. Frolich, and P. Hanrahan. Two-handed direct manipulation on the responsive workbench. *Proc. of 1997 Symposium on Interactive 3D Graphics*, pages 107–114, 1997.

[Che99] E. Chen. Six degree-of-freedom haptic system for desktop virtual prototyping applications. In *Proceedings of the First International Workshop on Virtual Reality and Prototyping*, pages 97–106, 1999.

[CL96] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. *SIGGRAPH 96 Conference Proceedings*, pages 303–312. 1996.

[CMS88] Michael Chen, S. Joy Mountford, and Abigail Sellen. A study in interactive 3-D rotation using 2-D control devices. *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 121–129, 1988.

[DMW+98] L. Durbeck, N. Macias, D. Weinstein, C. Johnson, and J. Hollerbach. Scirun haptic display for scientific visualization. *Phantom Users Group Meetings*, 1998.

[GEL00] A. Gregory, S. Ehmann, and M. C. Lin. *inTouch*: Interactive multiresolution modeling and 3d painting with a haptic interface. *Proc. of IEEE VR Conference*, pp. 45-52, 2000.

[Gib95] S. Gibson. Beyond volume rendering: Visualization, haptic exploration, an d physical modeling of element-based objects. In *Proc. Eurographics workshop on Visualization in Scientific Computing*, pages 10–24, 1995.

[GME+00] A. Gregory, A. Mascarenhas, S. Ehmann, M. C. Lin, and D. Manocha. 6-dof haptic display of polygonal models. *Proc. of IEEE Visualization Conference*, pp. 139-146, 2000.

[GW92] Michael Gleicher and Andrew Witkin. Through-the-lens camera control. *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 331–340, 1992.

[IN93] H. Iwata and N. Noma. Volume haptization. *Proc. of IEEE VRAIS*, pp. 16-23, 1993.

[KD95] K. N. Kutulakos and C. R. Dyer. Global surface reconstruction by purposive control of observer motion. *Artificial Intelligence*, 78:147–177, 1995.

[LLPN00] D. A. Lawrence, C. D. Lee, L. Y. Pao, and R. Y. Novoselov. Shock and vortex visualization using a combined visual/haptic interface. *Proc. of IEEE Visualization*, pp. 131-137, 2000.

[MB95] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. *SIGGRAPH 95 Conference Proceedings*, pages 39–46. ACM SIGGRAPH, 1995.

[MBS97] Mark R. Mine, Frederick P. Brooks, Jr., and Carlo H. Séquin. Moving objects in space: Exploiting proprioception in virtual-environment interaction. *SIGGRAPH 97 Conference Proceedings*, pages 19–26. 1997.

[MCR90] Jock D. Mackinlay, Stuart K. Card, and George G. Robertson. Rapid controlled movement through a virtual 3D workspace. *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 171–176, 1990.

[MPT99] W. McNeely, K. Puterbaugh, and J. Troy. Six degree-of-freedom haptic rendering using voxel sampling. *Proc. of ACM SIGGRAPH*, pages 401–408, 1999.

[O'R87] J. O'Rourke. *Art Gallery Theorems and Algorithms*. The International Series of Monographs on Computer Science. Oxford University Press, New York, NY, 1987.

[PBBW95] Randy Pausch, Tommy Burnette, Dan Brockway, and Michael E. Weiblen. Navigation and locomotion in virtual worlds via flight into Hand-Held miniatures. *SIGGRAPH 95 Conference Proceedings*, pages 399–400. 1995.

[PBG92] C. Phillips, N. Badler, and J. Granieri. Automatic viewing control for 3d direct manipulation. *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 71–74, 1992.

[PFC+97a] J. Pierce, A. Forsberg, M. Conway, S. Hong, R. Zeleznik, and M. Mine. Image plane interaction techniques in 3d immersive environments. *Proc. of 1997 Symposium on Interactive 3D Graphics*, pages 39–44, 1997.

[PFC+97b] J. Pierce, A. Forsberg, M. Conway, S. Hong, R. Zeleznik, and M. Mine. Image plane interaction techniques in 3d immersive environments. *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 39–44, 1997.

[Pit99] R. Pito. A solution to the next best view problem for automated surface acquisition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21:1016–1030, 1999.

[RH92] Warren Robinett and Richard Holloway. Implementation of flying, scaling, and grabbing in virtual worlds. *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, volume 25, pages 189–192, 1992.

[Sho87] Ken Shoemake. Quaternion calculus and fast animation, computer animation: 3-D motion specification and control. *SIGGRAPH 1987 Tutorial*, pp. 101–121, 1987.

[Tar91] Konstantinos Tarabanis. *Sensor Planning and Modeling for Machine Vision Tasks*. PhD thesis, Columbia University, 1991. Department of Computer Science.

[TAT95] K. A. Tarabanis, P. K. Allen, and R. Y. Tsai. A survey of sensor planning in computer vision. *IEEE Trans. Robotics and Automation*, 11:86–104, 1995.

[Wer94] Josie Wernecke. *The Inventor Mentor*. Addison-Wesley, 1994.

[WO90] C. Ware and S. Osborne. Exploration and virtual camera control in virtual three dimensional environments. *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 175–183, 1990.

[ZF99] Robert Zeleznik and A. Forsberg. Unicam 2d gestural camera controls for 3d environments. *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 169–173, 1999.

[ZMHH97] H. Zhang, D. Manocha, T. Hudson, and K. Hoff. Visibility culling using hierarchical occlusion maps. *Proc. of ACM SIGGRAPH*, pp. 77-88, 1997.