Height Field Haptics

Kristin Potter kpotter@cs.utah.edu
David Johnson dejohnso@cs.utah.edu
Elaine Cohen cohen@cs.utah.edu
School of Computing, University of Utah

Abstract

We present a system for haptically rendering large height field datasets. Inasmuch as, height fields are naturally mapped to piecewise bilinear patches. We develop algorithms for intersection, penetration depth, and closest point tracking using bilinear patches. In contrast to many common haptic rendering schemes for polygonal models, this approach does not require preprocessing or additional storage. Thus, it is particularly suitable for the large scale datasets found in geographic and reverse engineering applications.

1 Introduction

Haptic interfaces augment the experience of a computer user with touch and feel. Scenes are graphically displayed, and with a haptic interface such as the SensAble Phantom, the user can feel the surface of the model through force feedback. Such an interface enhances the experience of the user beyond the visual by allowing another avenue of exploration. The user feels features and properties of the model that may not be detected visually.

Current work in haptic rendering has concentrated on polygonal models. To meet the fast update rates, typically one kilohertz, required by haptic interaction, preprocessing and additional storage is required. Geographic datasets may contain tens of millions of data points, so general polygonal approaches are too slow to render these models haptically.

Alternatively, we treat height fields as collections of bilinear patches. Bilinear patches naturally provide a smooth force response across the patch and suffer from fewer artifacts across patch boundaries such as tessellation artifacts associated with polygonal models.

To successfully employ bilinear patches in a haptic rendering system, this paper develops fast techniques for approximating the intersection of rays with patches, computing penetration depths between a point and a patch, and tracking the closest point on a mesh of patches during the movement of the haptic interface point. The grid structure of the height field allows the necessary computations to be locally constrained to patches in a small neighborhood of the haptic endpoint.

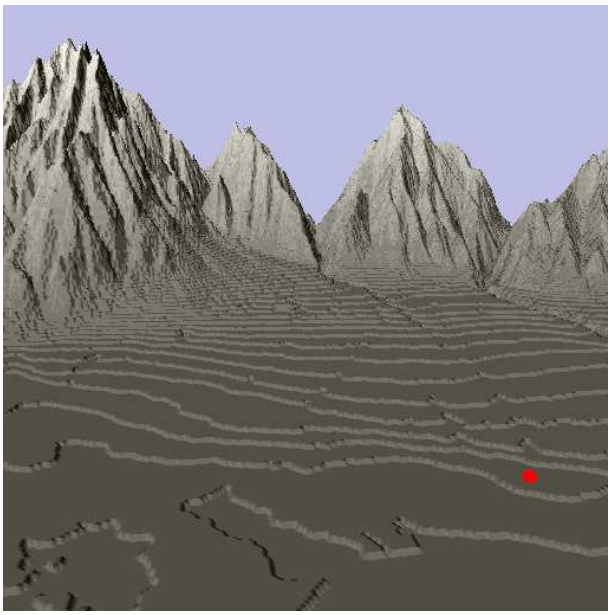

Figure 1. A screen shot of the system using the Sugarhouse dataset. The red sphere is the haptic endpoint.

These techniques are then combined into a haptic rendering system for large height field datasets. A screen shot of the system using a geographical dataset is depicted in Figure 1. Herein, we establish that bilinear patches are a viable alternative to polygonal representations of height field datasets, and produce com-

parable performance by minimizing the number of collision detection calculations and allowing for fast point tracking.

2 Background

The previous work in haptic rendering examined here is separated into model representation, haptic contact models, and the haptic display of surface properties.

2.1 Model Representations

Today, the most common geometrical representation of models in computer graphics is a set of triangles, or a triangular mesh. The pervasiveness of this representation has influenced computer haptics and much work has focused on the haptic interaction with triangular meshes. Problems are encountered while haptically rendering triangular models when the mesh contains cracks and holes, or when the topological connectivity between triangles is absent. Haptic traversal of the model is susceptible to failure in these instances because inconsistencies in topology will effect the forces applied to the haptic interface. Haptic rendering algorithms which present solutions to these types of problems, like [6] and [11], have treated the contacted triangles as constraint planes in order to reliably move across a surface. With large models however, the haptic interface point may need to cross many triangles to reach a local minimum, and the number of constraint planes may exceed what can be computationally sustained at haptic rates [10].

If triangle connectivity information is available, a local gradient search can replace the more complicated constraint plane solution [2]. This approach is used to interact haptically with high resolution triangular models. The local search is achieved in nearly constant time given the temporal coherence of haptic interaction rates. However, the models still require a hierarchical bounding volume fitting and storage in order to efficiently perform the collision detection used to initialize the method.

In [10], a triangulated search algorithm was adopted to haptically render height field data. Rather than searching along the triangle faces, an edge-based search was used to increase the interaction speed with large datasets. Properties of grid data were used to reduce the collision detection time by considering only the triangles in the region of the interaction point. While this technique was shown to be effective, there remains the possibility of incorrect minimization from the edge search, and the introduction of discontinuity artifacts through the choice of how to facet the quadrilateral data. Also, it is not clear that the common technique of *force-shading* to smooth the force response over faceted data would be applicable to the edge search method.

While polygons are the most popular model representation used with haptic interaction, there has also been work on other model representations such as NURBS [8][9] and implicit surfaces [7]. These methods also utilize local tracking to reduce the complexity of the haptic computation, and can take advantage of the smoothness of the data to improve the haptic rendering.

2.2 Haptic Contact Models

Haptic contact models simulate the interaction between the endpoint of the haptic device and the model. This interaction can be thought of as a massless point moving along the object [11], a sphere proxy contacting the model [6], or two polygonal models coming in contact with each other [3]. The latter of these techniques is commonly employed in virtual prototyping environments to better understand the interactions between two objects. A similar approach models the haptic device as a ray rather than a single point to simulate the contact between the tip and side of the haptic device against an object [1]. The massless proxy technique however is most closely related to the work presented here to simulate the interaction between the user and the model.

The function of the proxy point [6] and the *god-object* [11] techniques is to keep a position constrained to surface facets of the model such that the distance from the haptic endpoint to the proxy point is minimized, and is thus the virtual position of the haptic interface on the surface. The haptic endpoint itself is not constrained, and may move about the object freely. The proxy point maintains a constrained position that does not penetrate the surface, and moves across the surface from the previous location of the haptic endpoint to the current position, to arrive at a local minimum distance. A main advantage of proxy type algorithms is the ability to haptically render infinitely thin objects such as planes or polygons. The proxy cannot slip through the polygon, and thus more freedom is given as to how the models are represented. Moreover, the amount of force to return to the user is directly proportional to the distance from the haptic endpoint to the proxy point.

The haptic contact model that this work is most closely related to is the proxy graph approach [10] which uses two modes of operation depending on the current relationship between the haptic endpoint and

the model. When the haptic endpoint is not in contact with the model and an intersection with the model does not occur, free space minimization allows the proxy point to proceed directly to the haptic endpoint. If an intersection does occur, the algorithm enters constrained minimization to determine a local minimum distance from the proxy point on the surface to the haptic endpoint. In this mode, the proxy point traces along vertices of the model to locate the edge or face on which the local minimum resides. Once the edge or face is found, the orthographic projection of the goal point (haptic endpoint) is verified to be on that edge or face.

The haptic contact model presented in this paper is similar to the proxy graph approach in that there are two different modes, a collision detection mode and a trace along the model to determine the proxy point when the haptic endpoint is in contact with the model. In contrast to the proxy graph algorithm, the approach presented here begins the search for the resting point of the proxy at the first location of possible intersection. This ensures that the local closest point found is the first location on the surface intersected by the haptic endpoint. If the local closest point resides on the interior of the patch, it is kept as the local minimum, else a neighborhood of patches is checked for a closest point, and the search moves to the patch closest to the goal location. This approach quickly determines a local minimum, and also allows the use of concave models which can cause naive local closest point searches to infinitely loop.

2.3 Surface Properties

An advanced feature added to many haptic rendering systems is the ability to render the properties of the surface. A common such feature is force shading [4] which interpolates surface normals to make the surface feel smoother and reduce the effects of polygonal boundaries on the haptic feedback. In addition, frictional surfaces and textures can be haptically rendered to improve the feel of the model [2].

3 System Overview

The haptic rendering system presented here uses a SensAble Phantom interface device running on a dual processor Pentium 4 2.4 GHz Linux computer with a gigabyte of RAM and a GeForce 4Ti 440 graphics card. The haptic and graphics displays are separated into threads with the haptic display running at highest priority. Because of the large nature of the input data, the graphics display uses a level-of-detail approach to display the data at a lower resolution than used for the haptic display to ensure a fast update rate for visual display.

4 Bilinear Patch Approach

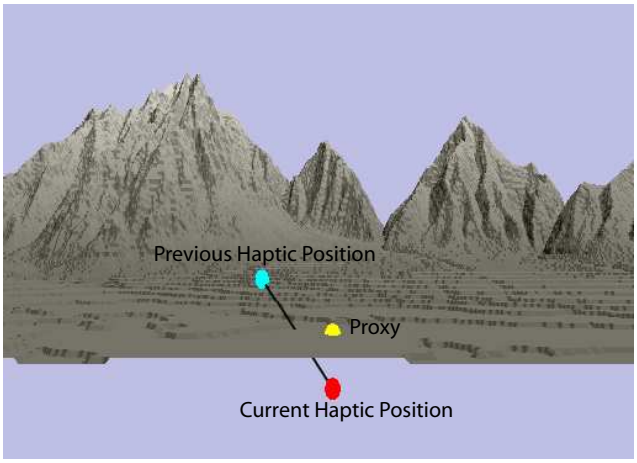

Figure 2. Cutaway of the system in use (the proxy is enlarged for viewing).

The approach presented here begins by building a bilinear patch grid from a height field which is typically an image containing depth information. Each bilinear patch interpolates among four neighboring pixels, creating a grid of patches which allows for fast indexing into the data. The haptic model maintains a proxy point which follows the haptic movement along the surface while the haptic endpoint is in collision with the model. Collision with the model is determined by detecting the intersection of the ray between the previous and current haptic positions with the surface of the model. The proxy as well as the ray from the previous to current haptic endpoints can be see in figure 2. Finally, forces proportional to the penetration depth are applied when the current haptic endpoint is determined to be in contact with the model.

4.1 Model Representation

The height field data consists of images with depth values encoded in the alpha channel. This data representation is a rectangular grid which allows direct representation using a bilinear patch grid. For every four data points in an image, a bilinear patch is constructed as illustrated in Figure 3. Thus, with the exception of

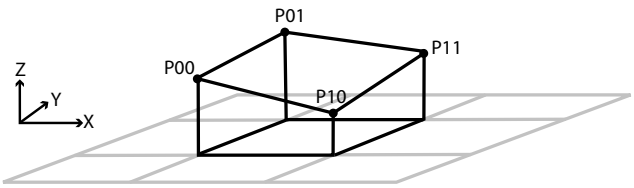

Figure 3. Points P_{00}, P_{10}, P_{11} and P_{01} are the four vertices of the center bilinear patch.

positions on the edges of the images, all data points influence four bilinear patches.

Each bilinear patch is constructed by interpolating between the four data points. Every data point specifies a vertex of the patch, so a combined linear interpolation is computed in the u-v parameters as follows:

$$P(u,v) = uvA + uB + vC + D \tag{1}$$

where

$$A = P_{11} - P_{10} - P_{01} + P_{00} \tag{2}$$

$$B = P_{10} - P_{00} \tag{3}$$

$$C = P_{01} - P_{00} \tag{4}$$

$$D = P_{00} \tag{5}$$

and P_{00}, P_{01}, P_{10}, and P_{11} are the four vertices of the patch, having 3-space coordinates. Taking advantage of the regularity of the grid allows for the simplification of the bilinear patch equation. Since the neighboring grid point are always 1 unit apart in x and y (we assume z to be the depth coordinate), the x and y equations for each patch can be simplified so that we solve for u and v separating the vector equation into respective components directly.

$$P_x(u,v) = u + D_x \tag{6}$$

$$P_y(u,v) = v + D_y \tag{7}$$

where D_x and D_y are the x and y components of equation (5).

Some of the benefits of using bilinear patches to represent height field grids are that bilinear patch grids are simple, direct representations of the height field grids that requires no preprocessing of the data and preserve all topological information. Bilinear patches also allow for a direct 1-1 mapping from 2 to 3 space and back and interpolation between data points is implicit in the representation.

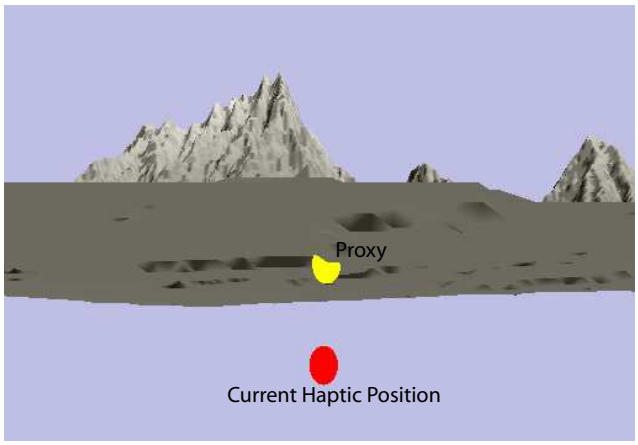

Figure 4. A close-up from underneath the model of the proxy and haptic position.

4.2 The Haptic Model

During the haptic exploration of the model, a ray from the previous to the current positions of the haptic endpoint is used to determine whether the haptic position collides with the model. Keeping track of the previous position provides the direction of penetration, and ensures stability of the computed intersection. If the haptic endpoint is determined to be in contact with a model, a proxy point tracks the position of the haptic endpoint on the surface of the model as shown in figure 4. The figure uses a sphere to visually represent the massless proxy. The proxy position is not allowed to penetrate the model, and once a collision is found, the proxy follows along the surface of the model until it rests at the closest location to the current haptic endpoint. If no collision is found, and the ray from the previous to current haptic position lies entirely outside the model, the proxy point is set coincident with the current haptic endpoint. Using a proxy point constrained to the surface of the model ensures that features of the model are not omitted once the haptic endpoint enters the model and also allows for infinitely thin models.

4.3 Haptic Update Algorithm

The following steps must occur during each update:

1. Determine if an intersection is possible.
2. Find the local point closest to the haptic endpoint.
3. Check if the haptic endpoint is inside the model.
4. If the endpoint is inside the model, apply forces.

4.3.1 Collision Detection

The search for a possible intersection begins at the previous haptic position and follows the projection of the haptic ray along the surface. This ensures that the first intersection found is the first possible location of a collision between the haptic endpoint and the model. If the state of the previous haptic position was already determined to be in contact with the model, the patch containing the local closest point from the previous iteration is used as the starting patch for the local closest point calculation, and no collision detection is preformed.

To determine the model location of potential intersections, the ray from the previous to the current haptic positions is projected into 2-space. This is a simple mapping of the x and y coordinates of the ray endpoints into the bilinear patch grid. Based on the scale and placement of the model, the x and y coordinates of the ray endpoints can be used to index into the bilinear patch grid. Once the bilinear patches containing the two endpoints of the ray are found, an incremental line drawing algorithm finds all bilinear patches through which the ray passes. This set of bilinear patches represents the possible locations of an intersection.

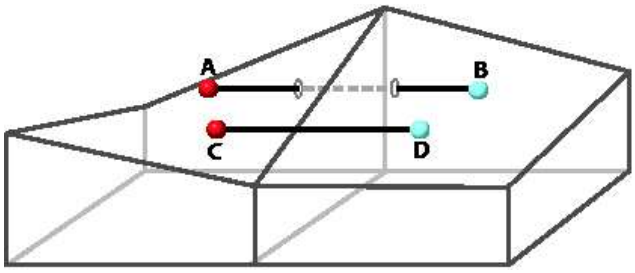

Figure 5. Both rays A-B and C-D pass the intersection test, but only AB intersects the model.

For every patch in the set of patches along the path of the ray, the maximum height of the patch is compared against the minimum height of the ray. The maximum height of the patch is the greatest z-value of the four patch vertices, and the minimum ray height is the smallest z-value between the two endpoints of the ray. The comparison of these two values reveals whether an intersection of the ray with the model is possible. If the minimum height of the ray is below the maximum height of the patch, an intersection may have occurred in this patch. Figure 5 illustrates two possible scenarios. Both rays in the figure satisfy the intersection condition, however the C-D ray does *not* actually intersect the model. This is not a problem however, because the final determination of a collision is based on the local closest point; this operation only estimates a starting position for the local closest point search. If the last patch in the ray path is reached and the intersection condition is not found to be true during this iteration, then there was no collision with the model.

4.3.2 Local Closest Point

The local closest point calculation finds the point on the surface of the model that is closest to the current haptic endpoint. The distance between a point and a parametric surface is described by

$$\mathbf{D}(u,v) = ||\sigma(u,v)||^2 \tag{8}$$

where

$$\sigma(u,v) = (\mathbf{S}(u,v) - \mathbf{P}). \tag{9}$$

Minimizing equation (8) corresponds to finding the parameter values of the local closest point on that surface and can be done by finding the simultaneous roots of the partial derivatives of 8, as in

$$\sigma(u,v) \cdot \mathbf{S_u} = \sigma(u,v) \cdot \mathbf{S_v} = 0. \tag{10}$$

Multi-dimensional Newton's method uses the Jacobian of the system of equations to solve. In the case of a bilinear surface, this Jacobian is simple to evaluate, as the higher derivatives are zero. The Jacobian to invert to find the simultaneous roots becomes

$$\begin{bmatrix} \mathbf{S_u} \cdot \mathbf{S_u} & \sigma(u,v) \cdot \mathbf{S_{uv}} + \mathbf{S_u} \cdot \mathbf{S_v} \\ \sigma(u,v) \cdot \mathbf{S_{uv}} + \mathbf{S_u} \cdot \mathbf{S_v} & \mathbf{S_v} \cdot \mathbf{S_v} \end{bmatrix}, \tag{11}$$

with partials of $\mathbf{S}(u,v)$ denoted by appropriate subscript. Because of the lack of higher derivatives on the bilinear patch, Newton's method is stable for these computations. A more detailed explanation of Newton's Method can be found in [5].

Newton's method may converge to a solution outside of the domain of the patch. If each coordinate of the point is between 0 and 1, then the local closest point is within the current patch and can be kept for collision determination and force calculations. If the coordinates of the point are outside the domain of the patch however, a search of a neighborhood of patches must occur to ensure that the local closest point actually resides on the model, rather than below or inside.

The neighborhood of patches to search is based on the location of the bounded closest point. The bounded closest point is the point on the current patch closest to the goal location. This point is restricted to the boundaries (edges and vertices) of the patch and is calculated

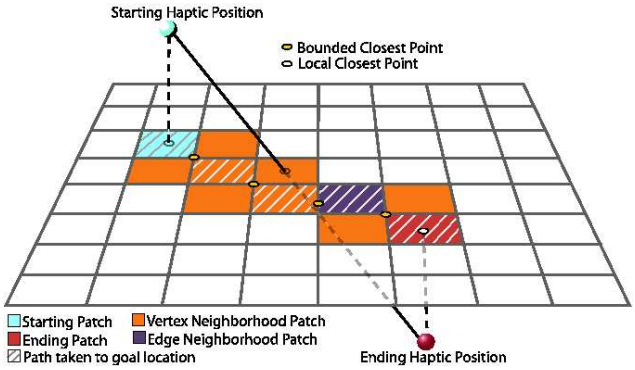

Figure 6. Bilinear patch neighborhoods are determined by the bounded closest point. A vertex point gives a three patch neighborhood, an edge point gives a single patch neighborhood.

by comparing and using the smallest of the minimum distances from the goal location to each edge and each vertex. If the location of the bounded closest point is on an edge, the patch that shares that edge is the only patch in the search neighborhood; if the point is on a vertex, the three patches sharing that vertex (if they exist in the model) are the search neighborhood. This is illustrated in Figure 6.

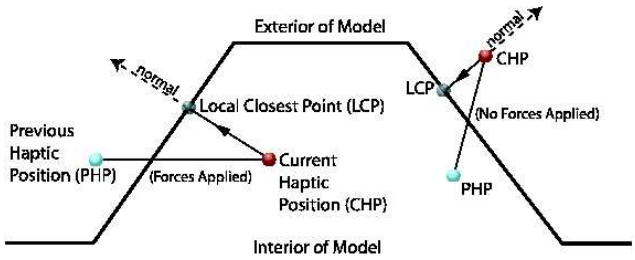

Figure 7. The dot product of the penetration vector (CHP-LCP) and the surface normal determines if a collision occurred.

Searching a neighborhood of patches allows the algorithm to terminate once a local closest point within the domain of the current patch is found, or if the bounded closest point of the current patch is closer than any point in the neighborhood of patches. The latter case often occurs in concave regions of a model because the local closest point resides on an edge or vertex. A naive local closest point search will infinitely loop between the patches that share the location of the local closest point. The neighborhood search eliminates the loop, and quickly finds the correct placement of the local closest point.

Model	Bilinear Patches	Update Rate
Sugarhouse	275,400	968 kHz
Moab	1,559,844	967 kHz
Devils Hole	1,514,496	975 kHz

Table 1. Model size and haptic update rates.

4.3.3 Collision Determination

The next step in the haptic update algorithm determines if the haptic endpoint is in contact with the model by finding the sign of the dot product of the penetration vector and the surface normal at the local closest point. The penetration vector is the vector from the current haptic position to the local closest point. The surface normal is either the normal to the surface at the local closest point, or if the bounded closest point was found to be located on a vertex or edge, the average of the surface normals at the edge or vertex of each adjacent patch. A positive dot product indicates that the current haptic endpoint is in contact with the model, and that forces should be applied. A negative dot product indicates that no forces should be applied, even if a collision was found. This is the case when the haptic endpoint leaves the model. Both cases are illustrated in Figure 7.

4.3.4 Force Calculation

The calculation of the force vector is based on the direction and magnitude of the penetration vector. The local closest point found in the previous step is used as the proxy position of the haptic interface on the surface of the model. From the proxy point, force is applied in the direction of the penetration vector, with strength proportional to the penetration magnitude. Because the neighborhood search for local closest point can restrict the point to an edge or vertex, the force vector will smoothly pass over bilinear patch boundaries.

5 Results

The datasets used by this system are depth images of topological data. The resolution of the data is 1 meter per unit vertically, 10 meters per unit horizontally. Thus, a single bilinear patch represents 10 square meters of land area which is why the elevation changes are so prominent in the images.

Table 1 gives the rate at which the system updates the haptic device for three different models. The rates reflected in the table describe the average rate of the various update states including the local closest point search, collision determination, and force reiteration based on non-movement of the haptic endpoint. The resolution of the models corresponds to a close investigation of the data, resulting in the crossing of only a few patches per update. Hence, the size of the model does not influence the update rate because the haptic update algorithm is performed over only a small number of patches. This is significant because both the local closest point search and collision detection have update rates of about 200 kHz. Thus, around 200,000 patches can be searched before updates rates become too slow for haptic rendering.

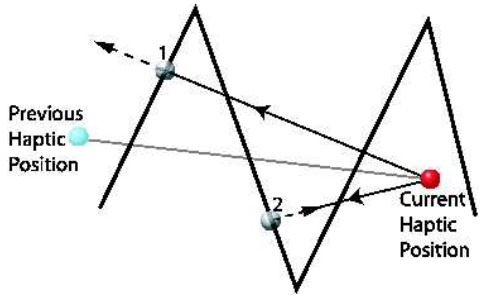

Figure 8. An error scenario in which the haptic ray passes through the left peak. The first iteration finds the local closest point at 1. The second iteration finds a closer and final local closest point at 2.

A possible problem with the approach presented here is that the algorithm can pass through high frequency data or the user can move the haptic endpoint fast enough so as to pass through features of the model. As figure 8 shows, the haptic ray moves through the two peaks in a single haptic update. Upon first iteration, the local closest point algorithm rests at position 1. This position would produce forces if used as the final position, however, the second iteration finds position 2 which is closer to the goal location. Position 2 does not result in applied forces, and thus the peak is passed through without resistance.

While this scenario is an obvious problem, its occurrence is rare, and a solution to the problem would be restrictive. Haptically rendering high frequency data would force the algorithm to apply forces from every bilinear patch in the patch of the haptic ray, thus forcing the haptic endpoint to halt at each patch. Not allowing the user to move the haptic endpoint through data in a single swipe would limit how fast the user could move the haptic device, and thus eliminate the possibility of haptic exploration of data at a high level. The typical haptic session however, displays the data such that only a few bilinear patches are crossed during a single time segment, and the problem is avoided almost always.

6 Conclusion

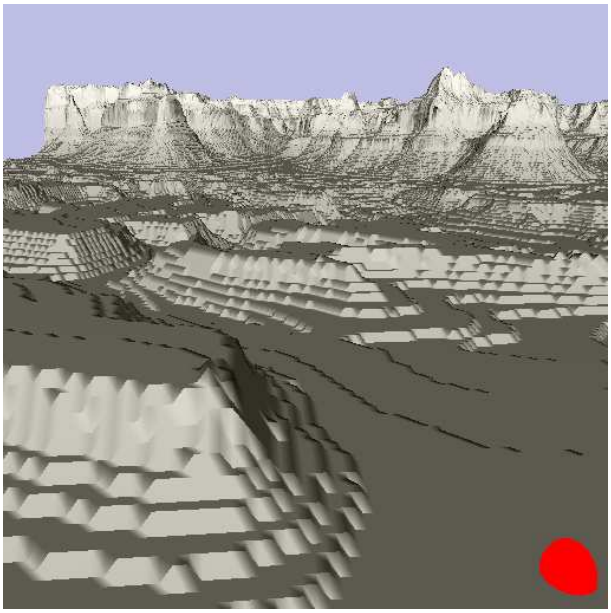

Figure 9. A screen shot of the Devil's Hole dataset.

Presented here is the essential structure of a new approach to haptically render large datasets using bilinear patches. The method minimizes the number of collision detection operations required per haptic update by taking advantage of the structure of bilinear patches and data. Additionally, bilinear patches do not require techniques to smooth tessellation, and offer a natural mapping for representing the grid data.

The current implementation is structured for fast exploration of the terrain data. A more detailed investigation of a small region of the dataset requires a higher resolution of data to be displayed. Work in progress incorporates a multi-res approach to the graphics display with this haptics approach, thus allowing the user to carefully examine fine details in the data. Table 1 shows that this system's haptic update rates are fast enough to explore datasets much larger than the ones

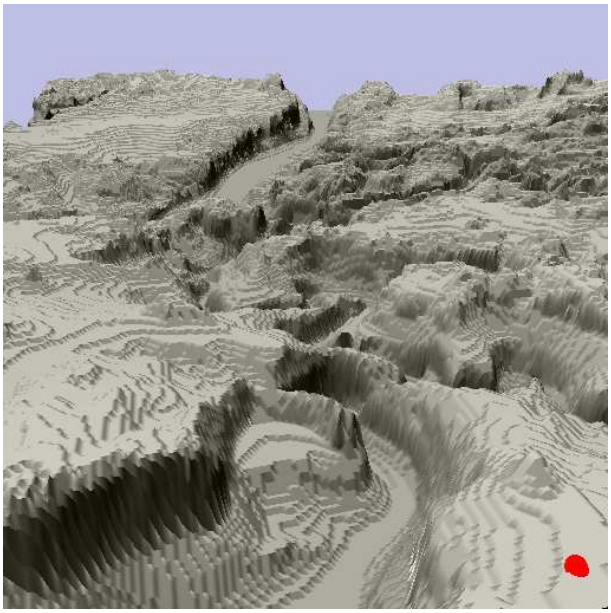

Figure 10. A screen shot of the Moab dataset.

we have been testing. However, to prepare for both rapid and fine level interrogation of datasets with sizes that are at least several orders of magnitude larger, we are extending the haptics algorithms and data structures to a multi-resolution approach as well.

7 Acknowledgements

This work was supported in part by ARO DAAD19-01-1-0013 and NSF DMI-9978603. All opinions, findings, conclusions or recommendations expressed in this document are those of the author and do not necessarily reflect the views of the sponsoring agencies.

References

[1] Basdogan, C., Ho, C.-H., and Srinivasan, M. A ray-based haptic rendering technique for displaying shape and texture of 3d objects in virtual environments. In *Proc. ASME Dynamic Systems and Control Division, DSC* (1997), vol. 61, pp. 77 – 84.

[2] Ho, C.-H., Basdogan, C., and Srinivasan, M. A. Efficient point-based rendering techniques for haptic display of virtual objects. *Presence 8*, 5 (1999), 477–491.

[3] Johnson, D., and Willemsen, P. Six degree-of-freedon haptic rendering of complex polygonal models. In *Haptics Symposium 2003* (March 2003), IEEE.

[4] Morgenbesser, H., and Srinivasan, M. Force shading for haptic shape perception. In *Proceedings of the ASME Dynamics Systems and Control Division* (1996), vol. 58, pp. 407–412.

[5] Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. *Numerical Recipes in C*. Cambridge University Press, 1988.

[6] Ruspini, D. C., Kolarov, K., and Khatib, O. The haptic display of complex graphical environments. In *SIGGRAPH 97 Conference Proceedings* (1997), pp. 345–352.

[7] Salisbury, J. K., and Tarr, C. Haptic rendering of surfaces defined by implicit functions. In *Proceedings of the ASME Dynamic Systems and Control Division* (1997), vol. DSC-Vol. 61, pp. 61–68.

[8] Stewart, P., and et al. Cad data representations for haptic virtual prototyping. In *Proceedings of DETC'97, 1997 ASME Design Engineering Technical conferences* (1997), pp. 14–17.

[9] Thompson II, T. V., Johnson, D. E., and Cohen, E. Direct haptic rendering of sculptured models. In *Symposium on Interactive 3D Graphics* (1997), pp. 167–176.

[10] Walker, S. P., and Salisbury, J. K. Large haptic topographic maps:marsview and the proxy graph algorithm. In *Proceedings of the 2003 symposium on Interactive 3D graphics* (2003), pp. 83–92.

[11] Zilles, C., and Salisbury, J. K. A constraint-based god-object method for haptic display. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, Human Robot Interaction, and Cooperative Robots* (1995), vol. 3, pp. 146–151.