Stateless Remote Environment Navigation with View Compression

Henning Biermann Aaron Hertzmann Jon Meyer Ken Perlin
Media Research Laboratory
Department of Computer Science
New York University
719 Broadway, 12th Floor
New York, NY 10003
{biermann,hertzman,meyer,perlin}@mrl.nyu.edu
http://www.mrl.nyu.edu/{biermann,hertzmann,meyer,perlin}
NYU Computer Science Technical Report

April 23, 1999

Abstract

We present a set of very low bandwidth techniques for navigating remote environments. In a typical setup using our system, a virtual environment resides on a server machine, and one or more users explore the environment from client machines. Each client uses previous views of the environment to predict the next view, using the known camera motion and image-based rendering techniques. The server performs the same prediction, and sends only the difference between the predicted and actual view. Compressed difference images require significantly less bandwidth than the compressed images of each frame, and thus can yield much higher frame rates. To request a view, the client simply sends the coordinates of the desired view and of the previous view to the server. This avoids the overhead of maintaining connections between the server and each client.

No restrictions are placed on the scene or the camera motions; the view compression technique may be used with arbitrarily complex 3D scenes or dynamically changing views from a web camera or a digital television broadcast. A lossy compression scheme is presented in which the client estimates the cumulative error in each frame, and requests a comprete refresh before errors become noticable.

This work is applicable to remote exploration of virtual worlds such as on head-mounted displays, Digital Television, or over the Internet.

1 Introduction

Within the next decade, users can look forward to a variety of compelling multimedia expereinces, made possible by the steady increase of computing power. One oft-stated goal is the development of shared virtual worlds and entertainment broadcasts that allow consumers to remotely explore 3D spaces. However, the speed of the Internet and other broadcast media cannot keep up with the demand for available bandwidth, if thousands of users are to have high-fidelity access to remote worlds. To address this issue, we introduce a class of compression schemes designed to significantly reduce the bandwidth required for remote navigation.

In a typical setup using our scheme, a user explores a virtual world on a client machine. This machine requests views of the world from a server machine. Sending the entire model over the network in advance is extremely slow, or impossible for dynamic scenes; one solution is to send each camera view from the server to the client for each frame as a compressed image. This solution will still require high network bandwidth to display video at interactive frame rates.

This paper presents a novel compression scheme that predicts the appearance of new views from previous views, using the known camera motion and image-based rendering techniques. This allows the server to send only incremental amounts of information for each frame, greatly reducing the bandwidth required for remote navigation. Unlike most image compression schemes, this method is cooperative: the client and server can communicate to determine the transmission for each frame that maximizes quality of service.

It is assumed that available network resources in the coming decade will lag far behind increasing processor power, and will be the limiting factor on navigation frame rates. Thus, some additional computation required for each frame is acceptable.

2 Related Work

This paper is based on image-based rendering, in which images are used as primitives rather than 3D models. (See Kang [12] for a survey.) Many authors have described image-based techniques for utilizing temporal coherence to reduce rendering latency [5, 22, 19, 7, 20].

Regan and Pose [18] and Mark et al. [16] use an image-based approach to overcome high network latencies. In these systems, reference images are generated and sent to the client system. The client then reprojects these images to generate new views at interactive rates until the next set of reference images arrive. Conner and Holden [6] discuss techniques for hiding the effects of latency in a shared world. These address latency rather than bandwidth, are are orthogonal to the approach presented in this paper.

Two commercial products, Apple QuickTime VR 3.0 [2] and LivePicture ImageServer [15] send panoramas over the network in pieces, so that the client may view the scene without having to receive the entire panorama. However, a large portion of the panorama must be downloaded before much of the panorama may be viewed. Also, these systems are not easily extensible to handle dynamic imagery or camera translations.

The approach presented in this paper is an image-compression scheme, based on specialized a priori knowledge about the images. It is similar to the multiscale compression schemes [3, 21] which use prediction and difference. This work is designed for a cooperative client-server approach, similar to Levoy [14]. The MPEG compression scheme [13] uses estimated motion vectors to predict video frames to compress prerecorded video. Similarly, Guenter et al. [8, 24] compress video using the known motion vectors from synthetic scenes. Chang et al. [4] use foveation, a spatially-varying compression scheme for remote viewing of very large 2D images. They use a generalization of wavelets, that allows an image to be displayed at different resolutions at different locations. In this system, the server sends only the image coefficients necessary to update the view of a static, 2D space.

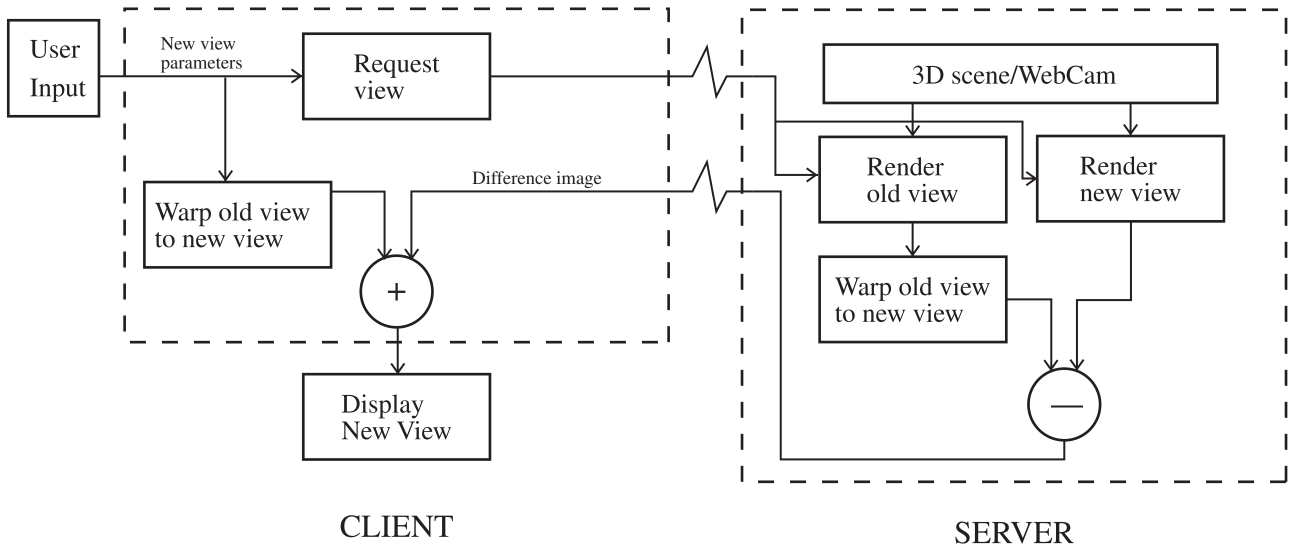


Figure 1: View compression for remote navigation. Rather than send every view of the virtual world over the network separately, the client can predict a desired view from previous views. The server needs only to send small corrections, instead of entire views.

3 Remote Navigation System

3.1 View Compression

Our remote navigation scheme enables a user, on a client machine, to interactively explore a 3D environment stored on a remote machine (the server). The client and server are connected by a network. Operation begins when the server sends the image of the view from a user's starting location in the 3D world.

The operation of the system is illustrated in Figure 1. For each frame, the client system requests the view for the new camera position, orientation, and focal length, by sending the coordinates of the new view V_1 and the coordinates of the old view V_0. While waiting for a response, the client predicts the appearance of the new view, by reprojecting the old view to the new view (Figure 2). The reprojection operation is described in Section 3.4. In the presence of network latency, predicted views can also be used to show intermediate frames [16].

The server renders both views V_0 and V_1, reprojects V_0 to the camera view of V_1, and computes the difference between the actual and predicted view. The difference image is then sent back to the client. The client then reconstructs the actual view by summing the prediction and the difference image.

A depth map is required for 3D reprojection. The same prediction/difference scheme is used for depth maps: the client predicts the depth map for a new view by transferring the current depth map, and a difference image is sent by the server. Since depth maps are usually smooth (except at object contours), their bandwidth requirements can be reduced by subsampling. We envision that our system will be used in conjunction with the latency compensation technique of Mark et al. [16], in which case the depth map is also required for generating intermediate frames.

The key observation of this paper is that previous views may be used to accurately predict new views. Ideally, the prediction will exactly match the new view in regions visible from the current view. In practice, small errors are introduced by sampling and changes to the underlying scene. Thus, the difference image will be highly compressible, requiring much less bandwidth than sending each new view in its entirety.

3.2 Stateless Navigation

Each new request contains all the necessary information for the server to create difference images. Thus, the system is *stateless*: it is not necessary to maintain dedicated connections between server and clients, nor does the server need to keep track of each client's history. The only requirement is that the server reconstruct previous views, either by caching or rerendering. For example, the server in a dynamically changing scene needs only to cache a reasonable number of previously rendered views.

The stateless framework allows the system to scale to an arbitrary number of clients, limited only by available network resources. There is no bookkeeping overhead as would be required for managing a large database of open connections.

3.3 Quality of Service

So far, we have described a system that uses lossless compression, i.e. each difference image is an exact difference, producing a perfect reconstruction on the client side. In cases of extremely limited bandwidth, it may be necessary to use a lossy compression scheme. In our experience, moderately lossy compression does not produce any noticable degradation in image quality over a few frames of transmission. However, errors tend to build up in the images when there is substantial overlap in a long sequence of consecutive views, such as when the user focuses on one small region of the schene.

If stateless navigation is not required, then the server can compensate for compression errors at each frame. The server keeps track of the view seen by the user, by performing the same reconstruction steps as the client. The difference images are then computed by reprojecting the user views. Thus any error present in the previous frame will be corrected by the difference image from the current frame. In this scheme, highly compressed difference images can be sent with no error build between frames.

In a stateless system, the responsibility falls to the client for keeping track of image error. Along with each difference image, the server could send the cumulative error (e.g. L_2 error) for each difference image. The client uses this information to keep a running estimate of the cumulative error for the image or for each region. When the image error exceeds some threshold, the client would request a complete new view, rather than a difference image. These periodic refreshes are analogous to the I frames used in MPEG [13]. The extra data sent is minimal (on order of a few bytes.)

3.4 Reprojection

The central operation of image-based rendering is "reprojection," where a sampled view of a 3D world is used to generate another view. If the views differ only in camera direction, then the operation is equivalent to enviroment mapping [9]. If the camera position moves, then additional depth or correspondence information is required [11, 1], and there may be ambiguities in the result. We use difference images to correct these ambiguities.

Reprojection can be performed with standard texture mapping hardware. Camera rotation without translation is performed by texture mapping the previous view onto a rectangle coinciding with the old view plane, and then rotating the camera. Camera translation requires additional depth information. The current view is mapped onto a textured depth mesh [7, 20], a mesh that represents the current view as a height field with the correct depth. The texture coordinates are chosen to prevent perspective correction [7, 10]. To reproject, the camera is rotated and translated, and the mesh is rendered. The depth mesh may be approximate, or subsampled; any errors in the prediction will be corrected by the difference image. Pixel-remapping [11, 1] algorithms can also be used, with similar results.

4 Experiments

We have implemented a prototype system using OpenGL [23]. The server and client run as applications on separate computers, and are connected by a TCP/IP socket. Image reprojection is performed as described in the previous section. Images are run-length encoded and then Huffman coded.

In order to test the algorithm, we compare the sizes of compressed full views, to compressed difference images. The 3D scene used for the test is shown in the color plates. Each view is a 256x256 RGB image (192 kB raw). Each test consisted of a single camera motion. The full view and the difference view were each run-length encoded, and then Huffman coded. The compression ratio is computed as the ratio of the size of the compressed difference view divided by the compressed actual view. The results are shown in Figure 2. For small camera rotations, (i.e. less than 5 degress), the compression ratio is about 55%. This means that the difference scheme would require half the bandwidth of the simple scheme to achieve the same frame rate. As the new view moves further from the old view, less of the new view can be predicted from the old view. Figure 3 shows the same comparison, for viewpoint translation horizontal to the viewplane. Again, compression ratios of about 55% are achieved for small camera movements. When using a more sophisticated still image encoding scheme (JPEG), we get roughly the same compression ratios as with RLE and Huffman coding.

We believe that significant speedup can be attained by using UDP, a stateless protocol, instead of TCP/IP, a connection-based protocol. Our current system could also be improved by interleaving client and server computations, and by using mipmapped textures instead of point sampling.

5 Discussion and Future Work

A stateless, low-bandwidth method for remote environment navigation has been demonstrated. A client system is used to explore an environment contained on a server system. During navigation, the client can predict new views by reprojecting old views, using the known camera motion and image-based rendering techniques. The server system needs only to send the difference between the prediction and the actual view. Since the difference image is highly compressible, the system requires relatively little bandwidth. It is assumed that both systems are fast enough to perform rendering and differencing in real-time, and that the primary bottleneck is in the intervening network. It is also assumed that the server can perfectly replicate the client's reprojection operation.

We considered an alternative approach to view compression, in which the server sends pieces of an underlying image-based model. In the case of panorama navigation, the client requests pieces of the envi-

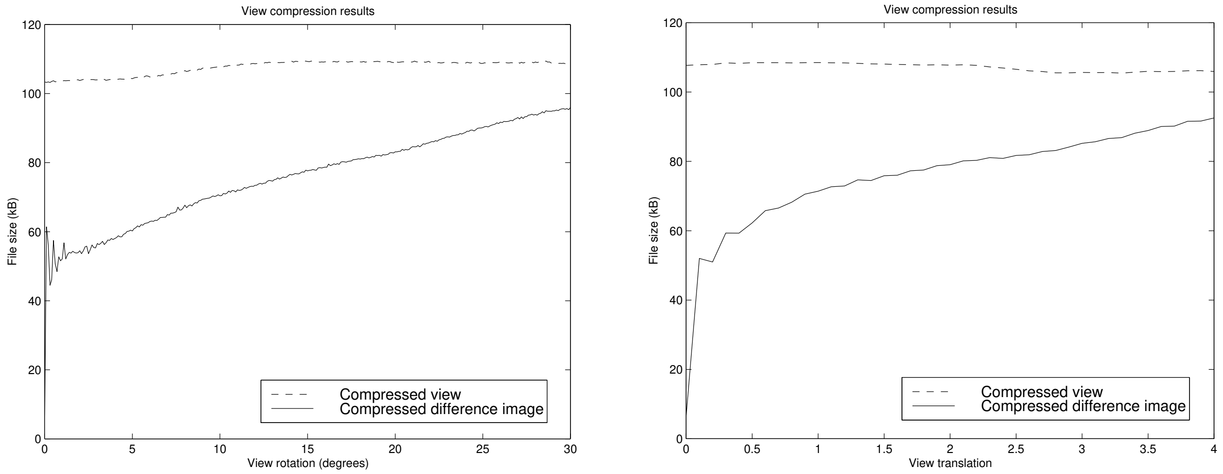


Figure 2: Comparison of frame sizes for compressed views vs. compressed difference images. The left plot shows pure rotations, and the right plot shows pure translations. For a translation of 4, about 25% of the new view is visible in the old view.

ronment map, rather than views, and stores them in a cache. This method has the advantage that the client can predict views based on all previous views, not just the last view. However, extending this method to full 3D navigation or to dynamically-changing scenes may require considerable bookkeeping effort.

References

[1] Shai Avidan, Amnon Shashua. Novel View Synthesis in Tensor Space. *IEEE Computer Vision and Pattern Recognition*, June 1997.

[2] Apple Computer Inc., QuickTime VR 3.0. http://www.apple.com/quicktime.

[3] Peter Burt, Edward Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*. 31(4):532-540, April 1983.

[4] Ee-chien Chang, Chee Yap, T.J. Yen. Realtime Visualization of Large Images over a Thinwire. *IEEE Visualization '97* (Late Breaking Hot Topics). Tucson, AZ, 1997.

[5] Shenchang Eric Chen, Lance Williams. View Interpolation for Image Synthesis. *SIGGRAPH 93.* August 1993. p279–288.

[6] Brook Conner and Loring Holden. Providing a Low Latency User Experience In A High Latency Application *ACM Symposium on Interactive 3D Graphics,* Providence, RI, 1997, pp. 45–48

[7] Lucia Darsa, Bruno Costa, and Amitabh Varshney. Navigating Static Environments Using Image-Space Simplification and Morphing. *ACM Symposium on Interactive 3D Graphics,* Providence, RI, 1997, pp. 25–34

[8] Brian Guenter, Hee Cheol Yun, and Russell Mersereau. Motion compensated compression of computer animation frames. *SIGGRAPH 93*, (August 93) pp. 297–304

[9] Ned Greene. Environment mapping and other applications of world projection. *IEEE Computer Graphics & Applications*. 6(11):21–29.

[10] Kenneth E. Hoff, Understanding Projective Textures: Correcting the Double-Perspective Distortion. http://www.cs.unc.edu/~hoff/techrep/index.html

[11] Leonard McMillan, Gary Bishop. Plenoptic modelling: an image-based rendering system. *SIGGRAPH 95*, (August 95) pp. 39–46.

[12] Sing Bing Kang. A Survey of Image-based Rendering Techniques. Technical Report 97/4. Digital Equipment Corporation Cambridge Research Laboratory, August 1997

[13] D. Le Gall. MPEG: A Video Compression Standard for Multimedia Applications. *Communications of the ACM,* Vol. 34, No. 4, April 1991, p46-58.

[14] Marc Levoy. Polygon-Assisted JPEG and MPEG Compession of Synthetic Images. *SIGGRAPH 95*, (August 95), p21–28.

[15] Live Picture Inc., http://www.livepicture.com/.

[16] William R. Mark, Leonard McMillan, Gary Bishop. Post-Rendering 3D Warping. *Proceedings of the 1997 Symposium on Interactive 3D Graphics* (Providence, RI), April 27-30, 1997, p7–16.

[17] Matthew M. Rafferty, Daniel G. Aliaga, Voicu Popsescu, and Anselmo A. Lastra. Images for Accelerating Architectural Walkthroughs. *IEEE Computer Graphics and Applications*. Nov/Dec. 98, p38–45.

[18] Matthew Regan, Ronald Pose. Priority Rendering with a Virtual Reality Address Recalculation Pipeline. *Computer Graphics (SIGGRAPH 94 Conference Proceedings).* 1994

[19] Jonathan Shade, Dani Lischinski, David Salesin, Tony DeRose, John Snyder. Hierarchical Image Caching for Accelerated Walkthroughs of Complex Environments. *SIGGRAPH 96*, p75–82

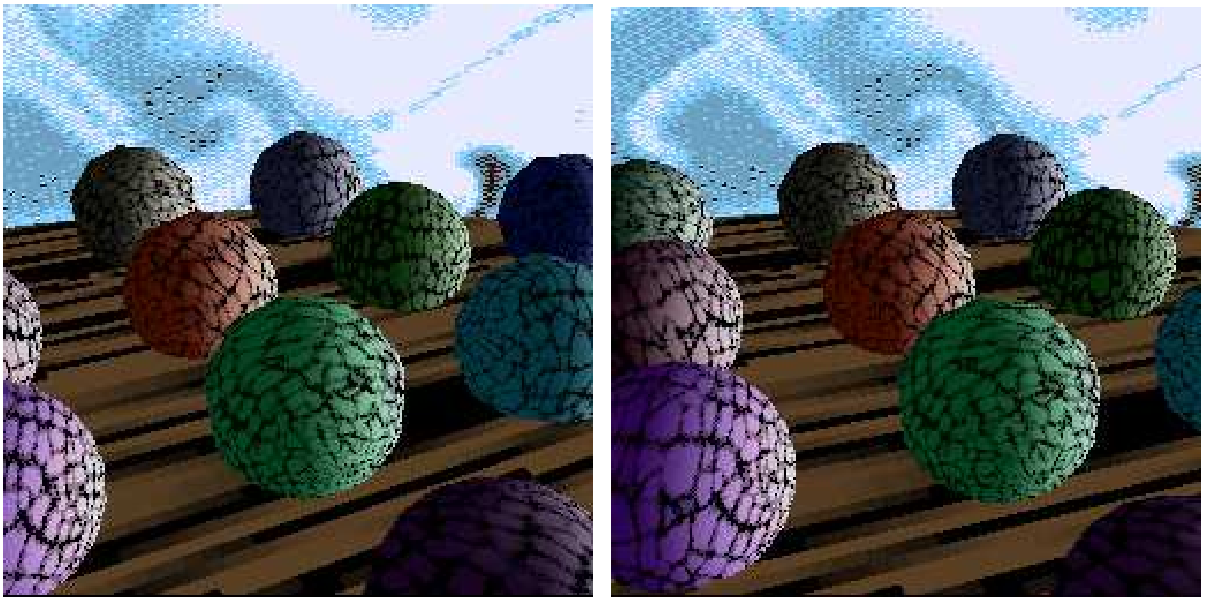
[20] François Sillion, George Drettakis, and Benoit Bodelet. Efficient Impostor Manipulation for Real-Time Visualization of Urban Scenery. Computer Graphics Forum (Proc. of Eurographics '97). Sep, 1997. p207–218.

[21] Wim Sweldens. The lifting scheme: A custom-design construction of biorthogonal wavelets. Technical Paper 1994:7, Industrial Mathematics Initiative, Department of Mathematics, University of South Carolina, 1994.

[22] Jay Torborg, Jim Kajiya. Talisman: Commodity Realtime 3D Graphics for the PC. *SIGGRAPH 96*. 1996. p353–363.

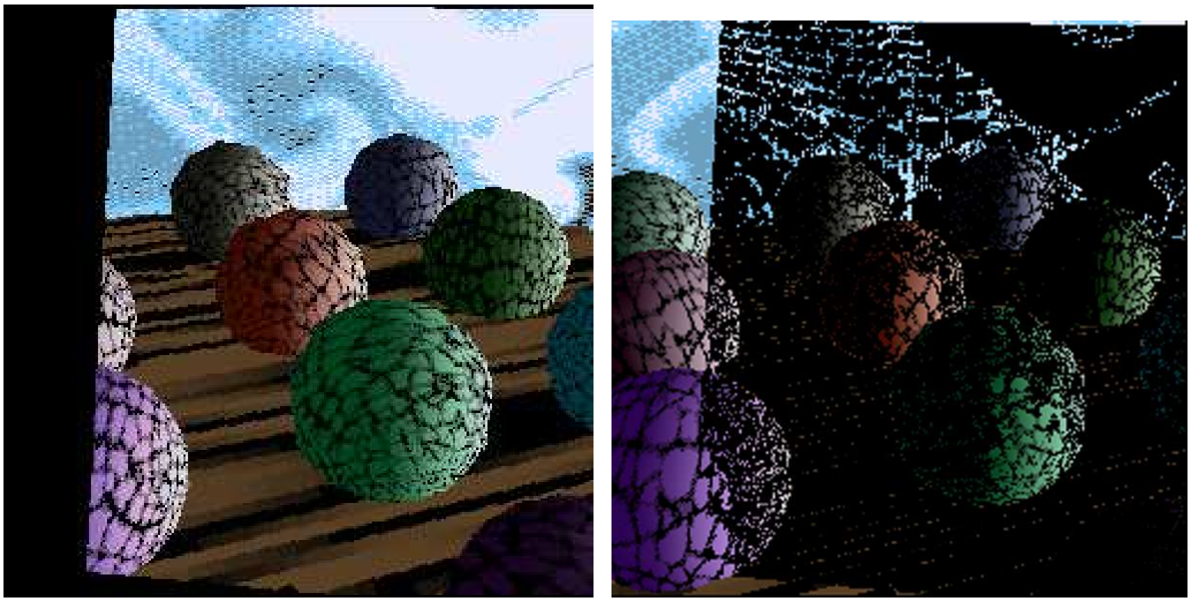
[23] Mason Woo, Jackie Neider, Tom Davis. *OpenGL Programming Guide, Second Edition*. Addison-Wesley, 1997.

[24] Hee Cheol Yun, Brian Guenter, and Russell Mersereau. Lossless compression of computer generated animation frames. *ACM Trans. on Graph.* 16, 4 (Oct. 1997), pp. 359–396



(a) (b)

Figure 3: (a) Old scene, before a camera motion. (b) New scene, after motion



(a) (b)

Figure 4: (a) New scene predicted from the old scene (b) Image showing where difference values exceed a threshold.

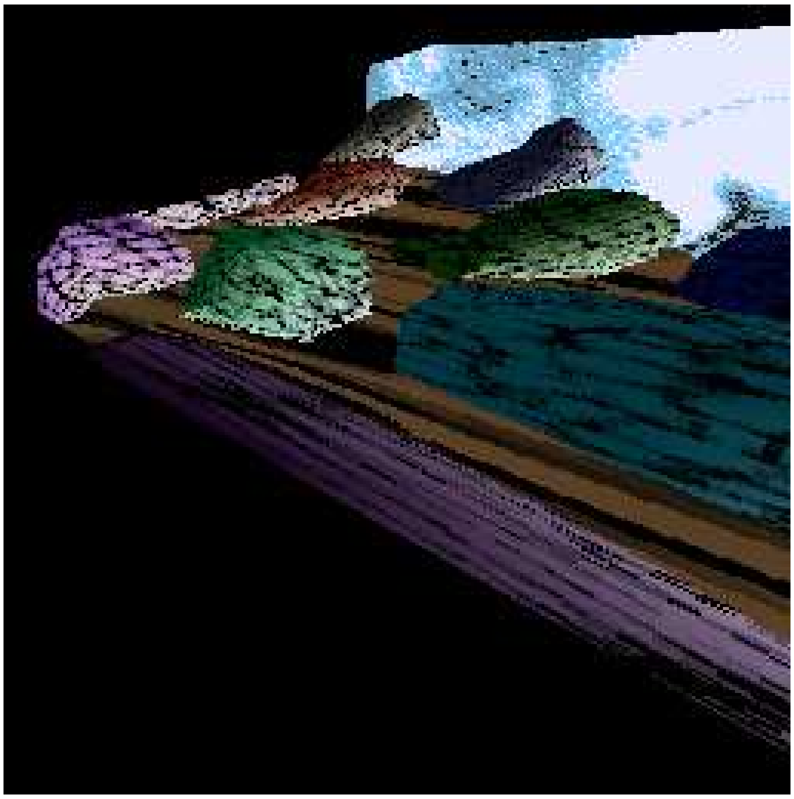


Figure 5: Textured depth mesh from the previous example, seen a side view.

(a) (b)

Figure 6: (a) View of a cylindrical environment map. (b) Difference image after a small horizontal rotation.