Depth Images: Representations and Real-time Rendering

Pooja Verlani, Aditi Goswami, P. J. Narayanan
Centre for Visual Information Technology
IIIT, Hyderabad 500032 INDIA
pjn@iiit.ac.in

Shekhar Dwivedi
GE JFWTC
Bangalore

Sashi Kumar Penta
nVidia Corporation
Bangalore

Abstract

Depth Images are viable representations that can be computed from the real world using cameras and/or other scanning devices. The depth map provides 2-$\frac{1}{2}D$ structure of the scene. A set of Depth Images can provide hole-free rendering of the scene. Multiple views need to blended to provide smooth hole-free rendering, however. Such a representation of the scene is bulky and needs good algorithms for real-time rendering and efficient representation. In this paper, we present a discussion on the Depth Image representation and provide a GPU-based algorithm that can render large models represented using DIs in real time. We then present a proxy-based compression scheme for Depth Images and provide results for the same. Results are shown on synthetic scenes under different conditions and on some scenes generated from images. Lastly, we initiate discussion on varying quality levels in IBR and show a way to create representations using DIs with different trade-offs between model size and rendering quality. This enables the use of this representation for a variety of rendering situations.

1. Introduction

Image Based Rendering (IBR) has the potential to produce new views of a real scene with the realism impossible to achieve by other means. It aims to capture an environment using a number of cameras that recover the geometric and photometric structure from the scenes. The scene can be rendered from any viewpoint thereafter using the internal representations used. The representations used fall into two broad categories: those without any geometric model and those with geometric model of some kind. Early IBR efforts produced new views of scenes given two or more images of it [5, 19]. Point-to-point correspondences contained all the structural information about the scene used by such methods. Many later techniques also used only the images for novel view generation [15, 11, 8, 20]. They require a large number of input views – often running into thousands – for modeling a scene satisfactorily. This makes them practically unusable other than for static scenes. The representation was also bulky and needs sophisticated compression schemes. The availability of even approximate geometry can reduce the requirements on the number of views drastically. The use of approximate geometry for view generation was a significant contribution of Lumigraph rendering [8] and in view-dependent texture mapping [6]. Unstructured Lumigraph [4] extend this idea to rendering using an unstructured collection of views and approximate models.

The *Depth Image (DI)* representation is suitable for IBR as it can be computed from real world using cameras and can be used for new view generation. A Depth Image consists of a pair of aligned maps: the image or texture map $\mathbf{I}$ that gives the colour of all visible points and a depth map $\mathbf{D}$ that gives the distance to each visible point. The image and depth are computed with respect to a real camera in practice though this does not have to be the case. The calibration matrix $\mathbf{C}$ of the camera is also included in the representation giving the triplet $(\mathbf{D}, \mathbf{I}, \mathbf{C})$. This is a popular representation for image-based modelling as cameras are cheap and methods like shape-from-X are mature enough to capture dense depth information. It has been used in different contexts [14, 16, 23, 13, 24]. The Virtualized Reality system captured dynamic scenes and modeled them for subsequent rendering using a studio with a few dozens of cameras [16]. Many similar systems have been built in recent years for modeling, immersion, videoconferencing, etc. [22, 3]. Recently, a layered representation with full geometry recovery for modeling and rendering dynamic scenes has been reported by Zitnick et al. [23]. Special scanners such as those made by CyberWare have also been used to capture such representations of objects and cultural assets like in the Digital

Michelangelo project [2, 1].

Depth Images have been used for IBR in the past. McMillan used it for warping [14] and Mark used an on-the-fly Depth Image for fast rendering of subsequent frames [13]. Virtualized Reality project computed them using multibaseline stereo and used them to render new views using warping and hole-filling [16]. Zitnick et al [23] use them in a similar way with an additional blending step to smooth discontinuities. Waschbusch et al [24] extended this representation to sparslely placed cameras and presented probabilistic rendering with view-independent point-based representation of the depth information.

The general framework of rendering Depth Images with blending was presented in [17]. In this paper, we provide a GPU based algorithm for real-time rendering of a representation consisting of multiple Depth Images. We also present a study on the locality properties of Depth Image based rendering. We then present the basic compression techniques for a collection of Depth Images including results of a proxy-based compression [18]. Results on representative synthetic data sets are presented to demonstrate the utility of the representation and the effectiveness of the algorithms presented here. Lastly we present a preliminary discussion on quality levels in IBR, similar to the level of detail used in graphics. The Depth Image representation can provide smooth trade-offs between the size of representation and rendering quality.

2. Depth Image Rendering

The process of rendering Depth Images is summarized below. Depth Images can be rendered using splatting or implied triangulation. Splatting treats each depth/colour combination as a 3D point with a certain size in the world or the image. Implied triangulation imposes a triangle-grid structure on the raster-ordered depth/colour pairs and draws them using standard graphics hardware. The triangles on the depth discontinuities have a large difference in depth along some of their edges and are not drawn. Depth discontinuities can result in holes in the rendered views. These can be filled by rendering using another Depth Image that sees that part of the scene. When multiple DIs are rendered, they should be blended when representing the same scene region. Thus, a representation consisting of multiple Depth Images can provide a complete representation that can use standard graphics algorithms for view generation.

The algorithm to render and blend the set of DIs is given below [17]. The optical axis of the new view is given by $\mathbf{n}$ and that of $\mathbf{DI}_i$ is given by $\mathbf{n}_i$.

for each Depth Image $\mathbf{DI}_i$ **do**

1. If $(\mathbf{n} \cdot \mathbf{n}_i \leq 0)$ skip i.
2. Generate the new view using $\mathbf{D}_i$ and $\mathbf{I}_i$.
3. Read back image to $\mathbf{I}'_i$ and the depth buffer to $\mathbf{Z}'_i$.

end for

for each pixel $\mathbf{p}$ in the new view **do**

4. Compare the $\mathbf{Z}'_i(p)$ values $\forall i$.
5. Keep the views within a threshold Δz of the nearest z value.
6. Compute the angle θ_i at the 3D point of p between the ray from DI i and the novel view. Compute the weight $w_i(p) = f(\theta_i)$ as a function of the angle.
7. Assign $\sum_i w_i I'_i(p)$ as the colour of the novel view pixel $\mathbf{p}$.

End for

It should be noted that a different combination of DIs could be blended for each pixel of the new view, based on the visibility and angle of each DI at that point [17]. The algorithm involves reading the depth and image buffers back and performing the blending on the CPU. These are expensive operations and hence real-time rendering was not achieved. The algorithm was able to render a frame every 2-3 seconds on an AMD64 machine with 1GB RAM and an nVidia 6600GT graphics card with 128MB of video RAM. The synthetic scene used for the performance figures, similar to those given in Figure 3, was represented using 20 Depth Images with ten each located on a circle at two different heights and pointed inwards towards the scene.

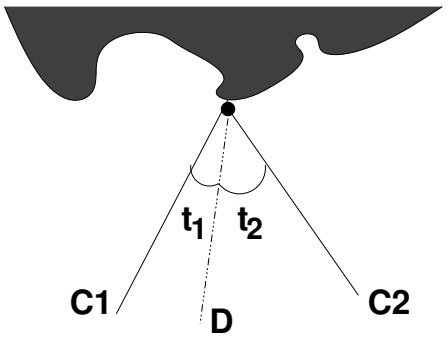

Figure 1. Rendered images from D using Depth Images C1 and C2 are blended based on the angles t_1 and t_2

The weighting function f ensures that the effect of a particular DI falls smoothly with novel view position. This avoids abrupt changes in colour values that can result if multiple DIs have different gains and offsets for

their images. Weighting functions like $\cos^k \theta$ or $e^{-k\theta}$ work for values of k of 2 or 3.

3. GPU Rendering of Depth Images

The read back of the framebuffer is the time consuming operation in the above algorithm. The modern GPUs have a lot of computation power and memory in them. If the read back is avoided and the blending is done in the GPU, the frame rate can possibly reach interactive rates.

We devised a 2-pass algorithm to render multiple DIs with per-pixel blending. The first pass determines for each pixel which views need to be blended. The second pass actually blends them. The property of each pixel blending a different set of DIs is maintained by the new algorithm. The overview of the algorithm is given in Figure 2.

Pass 1:

1. Enable z-buffering, disable lights, shading.
2. Clear depths.
3. **for** each Depth Image $\mathbf{DI}_i$ **do**
   (a) If $(\mathbf{n} \cdot \mathbf{n}_i \leq 0)$ skip i.
   (b) Render $\mathbf{D}_i$. Offset each point by Δz away from the novel view camera
   end for

Pass 2:

4. Enable lighting, shading, z-test. Disable z modification.
5. Clear colour buffers RGBA.
6. **for** each Depth Image $\mathbf{DI}_i$ **do**
   (a) If $(\mathbf{n} \cdot \mathbf{n}_i \leq 0)$ skip i.
   (b) Render $\mathbf{D}_i$ and $\mathbf{I}_i$ to new view normally.
   (c) At each framebuffer pixel p, compute the angle θ_i between DI_i and novel view and the weight $w = f(\theta_i)$.
   (d) Set colour $c(p)$ at p to $(A(p)c(p) + wI_i'(p))/(A(p) + w)$ where $I_i'(p)$ is the colour from rendering the DI i.
   (e) $A(p) = A(p) + w$
   (f) Leave the image in the buffer for next DI.
   end for

The first pass leaves z_m, the closest z value, in the Z-buffer for each pixel. The value is offset by Δz so that all pixels with depth less than $z_m + \Delta z$ will succeed the depth test and will be blended. The offsetting in eye space is done using a suitable vertex shader program. Lighting, shading and updating of the colour buffers is disabled to speedup the computations.

The second pass performs the blending. This is performed using a pixel shader that runs on the GPU. For each pixel, the shader has access to the novel view and DI parameters and the results of previous rendering using a Frame Buffer Object (FBO). Depending on which DIs had values near the minimum z for each pixel, a different combination of DIs can be blended at each pixel. The colour values and alpha values are kept correct always so there is no post-processing step that depends on the number of DIs blended. The algorithm also ensures there will be no exceeding of the maximum range of colour values that is a possibility if the summing is done in the loop followed by a division at the end.

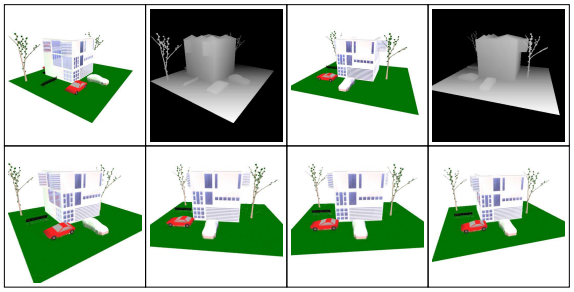

Figure 3. Top row: Depth Image pair for a synthetic view. Bottom row: New views generated using them.

The GPU algorithm used Vertex Buffer Objects and vertex arrays to store the DIs as triangulated models. The above algorithm was able to achieve a frame rate of 40 fps for the scene involving 10 DIs. The Depth Images have a resolution of 512×512, which is quite high. The frame-rate increased to 90 fps when the resolution is changed to 256×256 by dropping alternate rows and columns of the depth map. The video memory on the GPU was saturating and affecting the performance. The frame rate on a 20 DI scene was 10 and 35 for the higher and lower resolutions respectively. Typically, 4-5 DIs were blended for each new viewpoint.

3.1. Limited Set Rendering

Blending of all available and relevant DIs to generate a new view produces good results but could be wasteful in computational effort. It could also produce bad results if the DIs are inconsistent or noisy, as is often the case with those captured from a real scene. An alternative is to use a limited set of DIs within a specified distance of the new view for rendering. The Virtualized Reality system used 3 DIs to enclose the novel viewpoint from all directions [16]. Zitnick et al used only 2 DIs for view generation. Limiting the

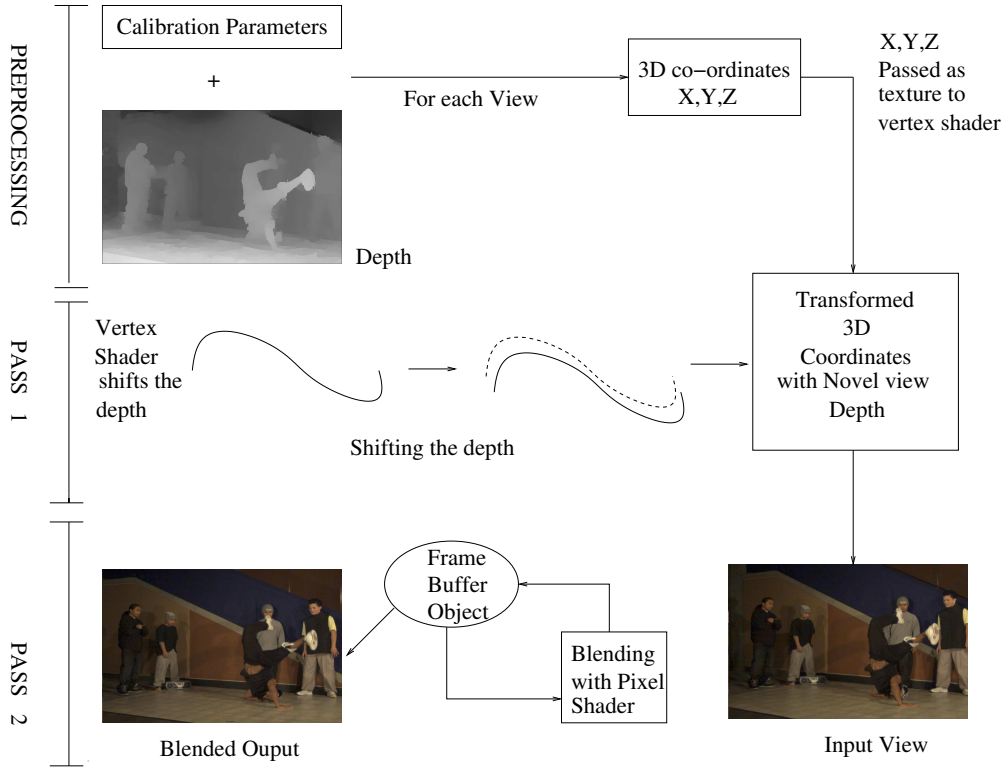

Figure 2. Block diagram of the GPU-based rendering algorithm

number of DIs used for rendering each view is essential for the scalability of the representation as a rendering linear in the number of views will seriously restrict the use of Depth Images.

The algorithm was modified to select only DIs that are within 40 degrees of the novel view. This results in 2-3 DIs being rendered for each view typically. The framerate increased to 50 fps at full resolution and to 100 fps at half the resolution for the 10 DI data set. The frame rates for the 20 DI data set were 20 and 55 for the two resolutions. We also tested the algorithm on the dancer data from Microsoft research. This dataset consists of 8 cameras arranged nearly linearly. Only 2 DIs can therefore be used for each new view generation. Our rendering algorithm was able to give a frame rate of 50 fps on it, whereas the algorithm used by them reported a frame rate of 15 fps [23].

3.2. Locality

A Depth Image is a local, $2\frac{1}{2}$D model of the scene. The view generated using it will be very good when the novel viewpoint is close to the view parameters of the Depth Image. If the novel view camera coincides with that of the DI_i, the rendered image will be perfect –

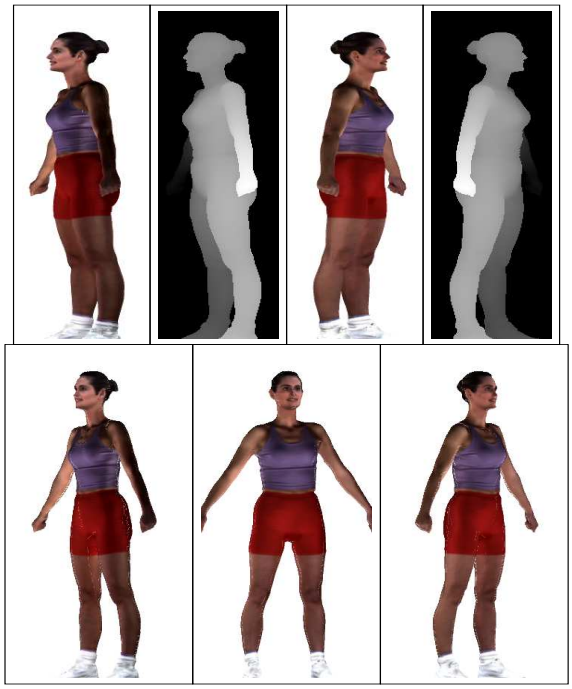

Figure 4. Top rows Input Depth Images. Bottom row: new views for the female model.

and identical to I_i – even if the depth map is totally random, when rendered using splatting. The dynamic selection of DIs for a novel view based on proximity can result in very high quality images throughout the space. The blending also ensures that the influence of a DI diminishes as we move away from it.

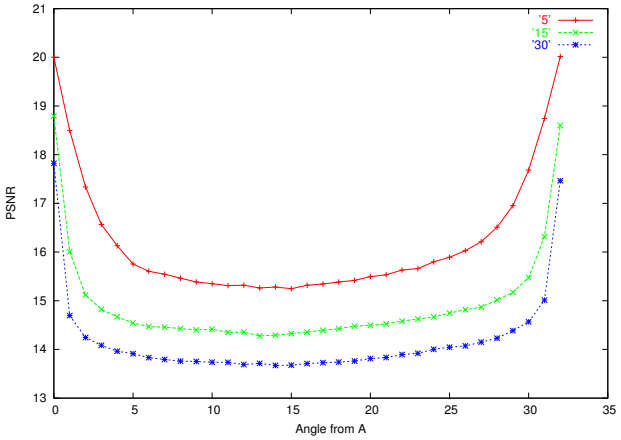

Figure 5. Locality: The PSNR is good when near a Depth Image but can fall when moving away. Results using 2 Depth Images with varying angle of the novel view for different noise levels

.

Figure 6. Locality: novel views from one Depth Image to another Depth Image

.

We explore the locality property quantitatively. For this, we use two DIs separated by about 70 degrees in Figure 6. New views were generated by moving the viewpoint from the first to the second in steps of 2 degrees. Since the data is synthetic, we compared the view generated using the DIs with a view generated using the original model itself and the PSNR values were calculated from the differences. The experiment is repeated with different levels of noise added to the DIs and the graph is shown in Figure 5. It can be seen that the quality of rendered view is high near the given DIs even with high noise.

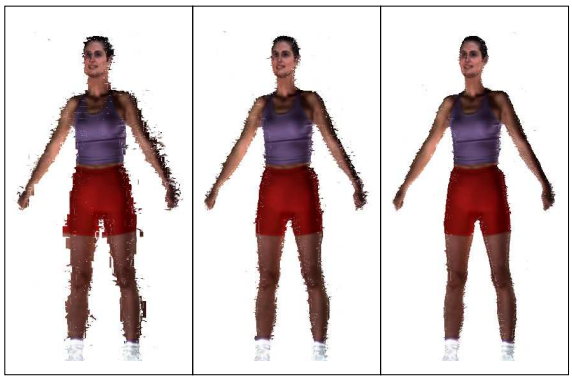

Figure 7. Random novel views of a particular view point with JPEG quality factors being 10, 50 and 90 respectively.

4. Compact Representations of Depth Images

The collection of Depth Images of a scene contains considerable redundant information. The images are of the same scene and the depth maps are of the same geometric objects. There is great scope for compaction possible.

JPEG Quality	Compression Ratio	PSNR (dB)
10	6.38	16.65
20	4.63	17.46
30	3.81	18.12
40	3.33	13.76
50	2.98	18.56
60	2.69	18.68
70	2.36	13.89
80	1.98	19.27
90	1.51	20.68
100	0.81	31.95

Table 1. Performance of JPEG on compressing depth-maps of the female model.

Both **D** and **I** components of the representation are like images. Image compression algorithms such as JPEG can be used to compress them. These algorithms are psycho-visually motivated and hence may not be the best for Depth Images, especially for the depth-maps which carry the geometric information. Earlier work by Krishnamoorthy et al confirm the bad match of JPEG as a compression scheme for depth images [10]. The results of our experiments of using JPEG for depth map compression are given in Table 1. Compression Ratios are computed as the ratio of total size of Zipped

8 bit depth maps and the total size of JPEG compressed depth maps. The results are not very good at all as shown in Figure 7. Lossless image compression schemes can be used to compress images and depth-maps, naturally, but yield only moderate compression ratios.

The above methods deal with each image and depth map as independent entities. Commonality across the DIs is not exploited by them. The topic of compressing multiple images of a scene – called lightfield usually – has attracted a lot of attention. Early light field compression techniques used vector quantization [11], disparity compensation techniques [21, 7] and wavelet transform based techniques [9, 7] to compress the images alone without using any geometry. Enhancement in prediction of accuracy is shown in [12] by using inferred geometry such as depth-maps and 3D models. These lightfield compression techniques can be used effectively to compress the image part of the collection of DIs.

Compression of multiple depth-maps of a scene has not been worked on a lot. Depth-maps differ from images qualitatively and new methods are needed to compress them. We introduced proxy-based compression for depth-maps [18]. The proxy geometry represents common, approximate information in the form of a parametric or geometric model. Each depth map is encoded with respect to a proxy geometry as a residue. The residues are highly correlated and compress well using techniques like JPEG or LZW.

When a proxy-based compression scheme applied to the female model, we achieved 4.81 compression ratio with PSNR being 29.54 dB using 1030 triangles proxy with 8 bit residue maps. Here compression ratio is computed as size of original ZIPed 16bit depth-maps to the size of ZIPed bit planes of residue maps and ZIPed triangle mesh of proxy. Progressive improvement of a random novel view is shown in the table 2 with the addition of bit planes from most significant bit to least significant bit.

# Bits	CR	PSNR (dB)
0	36.18	22.95
1	33.04	22.96
2	30.37	23.06
3	27.20	23.18
4	20.71	23.90
5	13.59	24.92
6	8.85	26.30
7	6.27	27.95
8	4.81	29.54
9	3.90	30.96
10	3.29	32.61
11	2.84	33.76
12	2.50	33.68
13	2.23	33.82
14	2.01	35.07
15	1.84	35.76
16	1.69	38.45
17	1.56	40.63
18	1.45	40.64

Table 2. Compression ratio and PSNR for progressive addition of bits from most significant bit to least significant bit of residues to the base depth map computed from the 1030 **triangles proxy for the female model.**

5. Quality Levels in IBR

Levels of detail (LoD) are employed in graphics for models to control the rendering time through the number of primitives or size of the model and texture. Discrete LoDs are generated using mesh decimation and texture sub-sampling. Progressive compression techniques allow for smoother levels of detail. IBR techniques have traditionally focussed on obtaining the highest rendering quality and haven't been studied for their trade-off between rendered image quality and the size or rendering time.

Depth Image representation admits smooth variation in the quality levels for IBR. Having several options of quality is important if the image-based model is used by different users, spanning PDAs, low-end graphics, and high-end graphics. There are several sources for a quality vs model size trade-off for the DI-based IBR representations.

1. The number of Depth Images can be varied to get different overall quality of rendering. A scene can be covered using a small number of DIs. Eight or ten DIs placed in orthogonal directions can provide adequate coverage. The quality of rendered output could suffer as some of the holes may not be filled optimally.

2. The resolution of the Depth Images can be varied by sub-sampling the depth map and/or the texture image. The view generation can be performed with low resolution DIs with a drop in quality. The 2D-grid organization of the DIs allow for smooth variation of the resolution using sub-sampling.

3. The proxy-based compression affords another

source of model reduction as shown in the progressive compression in the previous section. The polygonal proxy in combination with a number of images of the scene may be sufficient in some situations. This essentially reduces to view dependent texture mapping scheme [6] which used geometry with textures. Higher quality images can be generated by introducing the residues that give depth values. A spectrum of simplified models can be generated progressively by controlling the quantization of the residues.

In practice, only a small subset of proximate DIs are rendered for each novel view. The number DIs and the quality of the rendered image are not correlated in a straightforward way, as a result. Different quality levels for IBR can thus be generated by varying the DI resolutions, size of the proxy, and the detail of the residues using this scheme. Figures 8 and 9 shows the plots of CRs of triangle proxies with 134 and 948 triangles respectively and figure 10 shows the plot of PSNR when 134 triangle proxy is used. Table 3 shows the total size of a few possibilities and Figure 11 shows some results of rendering them using bunny model. Here compression ratio is computed with respect to the original ZIPed 8bit depth-maps.

Changing the resolution changes the number of primitives to be rendered for each view. If the model is being sent out for rendering, the size also reduces with the resolution. Reduction in resolution is thus essential to generate novel views on low-end systems, PDAs, etc. Changing the proxy model and/or the residue values change the model size as can be seen in Table 2. The number of primitives rendered remain the same, but the PSNR improves with the model size. Thus, changing the proxy and residue settings is better when the model needs to be sent over a low-speed network for rendering using a high-end system.

6. Discussion and Conclusions

In this paper, we presented the Depth Images as effective and practical representations for IBR. We gave real-time rendering algorithms for the representation and also presented results from a proxy-based compression scheme for depth-maps. Depth Images are promising representations as they can be captured in a non-obtrusive way from the world. The area of compression of Depth Images is still quite an open area and need new ideas. Of particular interest is the compression of depth and image values together in a single codec to take advantage of the intra-image and inter-image correlations.

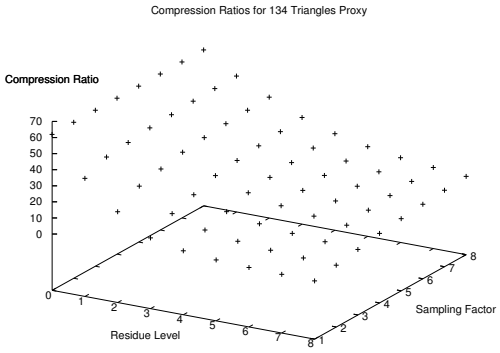

Figure 8. Shows the 3D plot of compression ratio when residue level and sampling factor for 134 Δs and all Depth Images are used for rendering

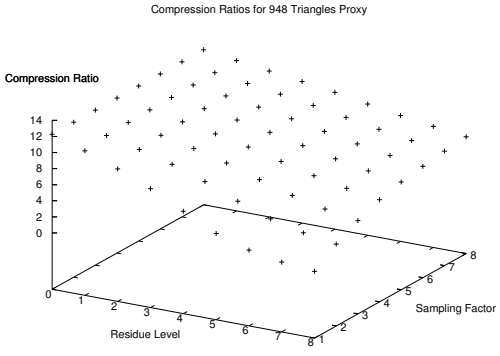

Figure 9. Shows the 3D plot of compression ratio when residue level and sampling factor for 948 Δs and all Depth Images are used for rendering

The issue of varying the quality levels smoothly will be important for an IBR representation to be practically usable. Our studies show that the DI-based representation afford a wide-variety of varying quality levels and image sizes.

References

[1] http://graphics.stanford.edu/projects/mich/.

[2] http://www.cyberware.com.

[3] H. Baker, D. Tanguay, I. Sobel, M. E. G. Dan Gelb, W. B. Culbertson, and T. Malzbender. The Coliseum Immersive Teleconferencing System. In *International Workshop on Immersive Telepresence (ITP2002)*, 2002.

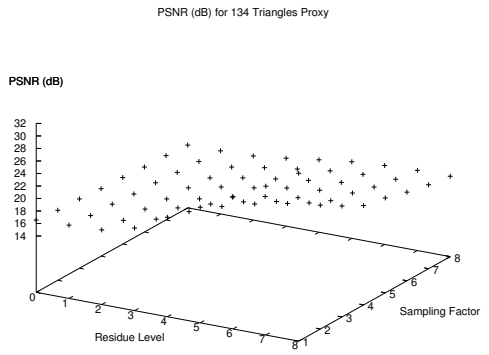

Figure 10. Shows the 3D plot of PSNR (dB) when residue level and sampling factor for 134 Δs and all Depth Images are used for rendering

#DIs	Sub sampling	Proxy Size (Δs)	#bits of residue	CR	PSNR (dB)
10	1	None	-	01.00	∞
05	1	None	-	02.00	21.46
10	4	None	-	07.08	20.90
10	8	134	0	61.95	15.04
10	4	134	4	18.33	18.35
10	8	134	8	13.66	17.86
10	4	134	8	08.25	20.89
10	1	134	8	01.22	31.83
10	8	948	0	12.27	17.73
10	1	948	8	01.30	32.47

Table 3. The variation is model size and rendering quality for the bunny model.

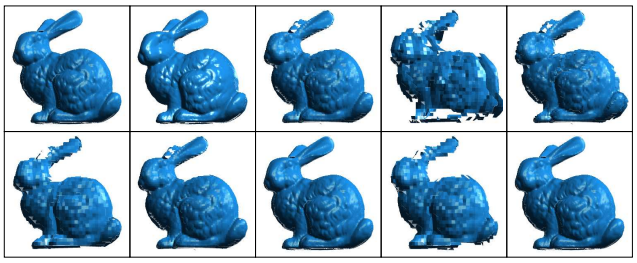

Figure 11. [left - right, top - bottom] Rendered images corresponding to rows of the Table 3

[4] C. Buehler, M. Bosse, L. McMillan, S. J. Gortler, and M. F. Cohen. Unstructured Lumigraph Rendering. In *SIGGRAPH*, 2001.

[5] S. Chen and L. Williams. View Interpolation for Image Synthesis. In *SIGGRAPH*, 1993.

[6] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and Rendering Architecture from Photographs: A Hybrid Geometry and Image-Based Approach. In *SIGGRAPH*, 1996.

[7] B. Girod, C.-L. Chang, P. Ramanathan, and X. Zhu. Light Field Compresion Using Disparity-Compensated Lifting. In *ICASSP*, 2003.

[8] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The Lumigraph. In *SIGGRAPH*, 1996.

[9] I. Ihm, S. Park, and R. K. Lee. Rendering of spherical light fields. In *Pacific Graphics*, 1997.

[10] R. Krishnamurthy, B.-B. Chai, H. Tao, and S. Sethuraman. Compression and Transmission of Depth Maps for Image-Based Rendering. In *International Conference on Image Processing*, 2001.

[11] M. Levoy and P. Hanrahan. Light Field Rendering. In *SIGGRAPH*, 1996.

[12] M. Magnor, P. Eisert, and B. Girod. Multi-view image coding with depth maps and 3-d geometry for prediction. *Proc. SPIE Visual Communication and Image Processing (VCIP-2001),* San Jose, USA, pages 263–271, Jan. 2001.

[13] W. R. Mark. *Post-Rendering 3D Image Warping: Visibility, Reconstruction, and Performance for Depth-Image Warping.* PhD thesis, University of North Carolina, 1999.

[14] L. McMillan. *An Image-Based Approach to Three-Dimensional Computer Graphics.* PhD thesis, University of North Carolina, 1997.

[15] L. McMillan and G. Bishop. Plenoptic Modelling: An Image-Based Rendering Algorithm. In *SIGGRAPH*, 1995.

[16] P. J. Narayanan, P. W. Rander, and T. Kanade. Constructing Virtual Worlds Using Dense Stereo. In *Proc of the International Conference on Computer Vision*, Jan 1998.

[17] P. J. Narayanan, Sashi Kumar P, and Sireesh Reddy K. Depth+Texture Representation for Image Based Rendering. In *ICVGIP*, 2004.

[18] Sashi Kumar Penta and P. J. Narayanan. Compression of Multiple Depth-Maps for IBR. In *Pacific Graphics*, 2005.

[19] S. M. Seitz and C. R. Dyer. View Morphing. In *SIGGRAPH*, 1996.

[20] H.-Y. Shum and L.-W. He. Rendering with concentric mosaics. In *SIGGRAPH*, 1999.

[21] X. Tong and R. M. Gray. Coding of multi-view images for immersive viewing. In *ICASSP*, 2000.

[22] H. Towles, W.-C. Chen, R. Yang, S.-U. Kam, and H. Fuchs. 3D Tele-Collaboration Over Internet2. In *International Workshop on Immersive Telepresence (ITP2002)*, 2002.

[23] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. In *SIGGRAPH*, 2004.

[24] Michael Waschbusch, S. Wurmlin, D. Cotting, F. Sadlo, and M. Gross. Scalable 3D Video of Dynamic Scenes. In *The Visual Computer*, 2005.