Parametric Proxy-Based Compression of Multiple Depth Movies of Humans

Pooja Verlani, P. J. Narayanan
Centre for Visual Information Technology
IIIT, Hyderabad 500032 INDIA
{pooja@research., pjn@}iiit.ac.in

Abstract

Capturing dynamic scenes in 3D using a suitable depth or structure recovery mechanism has become popular today for image based modelling, 3D telelconferencing, etc. The $2\frac{1}{2}$D geometric structure and the aligned texture – together called a depth image – are captured by such setups. Time varying sequences of depth images are called *depth movies*. Depth movies of humans performing interesting actions are being captured using specialized setups today in different research labs. The data so captured is often streamed to another location for immersive viewing using standard depth-image rendering techniques. The depth maps are bulky and need innovative algorithms for efficient representation and compression. In this paper, we present a compression scheme that uses parametric proxy models for the underlying action. We use a generic articulated human model as the proxy and the various joint angles as its parameters for each time instant. The proxy model represents common prediction of the scene structure for that time instant. It can be projected to each view and the difference or residue between the depth due to the proxy and the original depth can be used to represent the scene. The residues compress better and can exploit temporal and spatial relationship between multiple depth movies. We show results on several synthetic actions to demonstrate the usefulness of the scheme.

1 Introduction

Capturing the geometric and photometric structure of real-life scenes has attracted a lot of attention in the past decade [9, 15]. The applications of such systems include virtual-space tele-conferencing, remote 3D immersion, etc. Cameras are the most popular way to capture dynamic scenes. They directly provide the appearance from several view points. Several computer vision techniques help compute the geometric structure as $2\frac{1}{2}$D representation of each time instant. A depth image is a representation of the aligned depth map and image. Location (i, j) of the depth map stores the depth or distance to the closest 3D point along the imaging ray corresponding to that pixel.

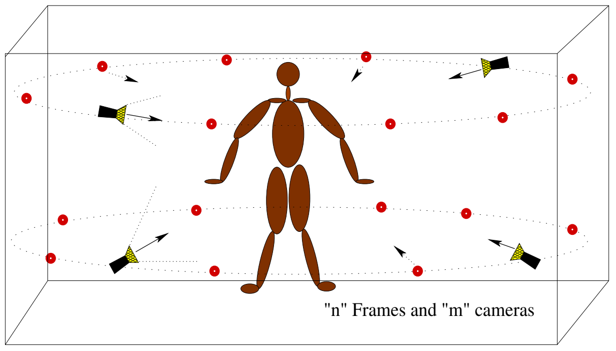

Figure 1: Setting of 20 cameras around the scene.

Multiple streams of depth images need to be captured for time-varying scenes. The appearance component from each camera is the traditional video. The time-varying depth map from each viewpoint also appears like a video but the content are distances. We call the combination of time varying appearance and depths as *Depth Movies*.

One common scenario involving depth movies is to transmit the captured event live to another location for 3D playback. Figure 1 outlines the typical setup with m cameras, each of which captures n frames of the event. This can be used for 3D teleconferencing or to provide an immersive display of an art event taking place remotely. For such a setting, the server at the capture site is linked over a network to a client at the rendering site. It is important to use efficient representation to reduce the network requirements, as depth images and depth movies are bulky objects. The problem of compressing individual video streams and multiple video streams have been studied well before. The handling of the time-varying depth maps from multiple sources has not been studied much.

Depth images have been used to interactively render the scene at the remote end [8]. Since a single depth image provides only a partial model, multiple depth images from multiple viewpoints are needed to provide a totally immersive experience. The frames of multiple depth movies provide just this information and can be used for rendering at the client's end. The representation, however, should support random access of the depth images of the depth movie for this purpose.

Depth movies contain sequences of depth maps varying with time. Data from the multiple viewpoints contains a lot of redundant geometric structure of the same scene. Effective compression needs to exploit the spatial redundancy in addition to the temporal redundancy in the depth maps. In this paper, we present a method to exploit the temporal and spatial redundancy present in the multiple depth movie representation. We focus on dynamic events involving a single human actor in this paper. The geometric structure is approximated using a light-weight *proxy* model for each time instant. We use a standard, articulated human model as a parametric proxy with the joint angles serving as its parameters. The time-varying parameters approximate the action adequately. The proxy model for each time-instant is projected to each viewpoint and the difference in depths between the proxy and the input is used as the residue. The residue can be encoded in multiple ways to provide different quality/size tradeoffs. We present related work in section 2. Section 3 presents our work on proxy-based compression of depth movies, followed by results in section 4.

2 Related Work

Geometric structure of real-life scenes can be captured using multicamera stereo, range scanners, etc. The Virtualized Reality system [9] captured multiple depth image sequences of subjects in a special, instrumented room. Multi-camera arrangements have been popular for 3D capture since then [13, 7, 1, 14, 2, 15]. Spatially immersive displays were also conceptualized for applications like virtual office [11]. The attempt is to capture dense or sparse 3D structure of the scene using cameras as time-varying depth and texture maps or depth movies.

While the compression of multi-camera images or lightfields has attracted a lot of attention, compression of depth maps hasn't been studied much. The standard image compression methods like JPEG emphasize perceived visual quality. They may not suit depth image compression, since a depth map represents a distance map for each pixel from the camera, and needs to be accurate for reconstruction. Compression of depth image of a static scene using JPEG produced unsatisfactory results [3]. They advocate the use of region of interest (ROI) and reshaping of the depths to preserve depth edges or occlusion boundaries when dealing with depth maps.

Disparity compensated compression of multi-view images attempts to exploit the spatial redundancy in a lightfield's representation. Computing disparity maps and using disparity compensation to predict the appearance in other views yielded higher compression for different types of textures or lightfields [5]. Block-based disparity compensation was also used for efficient encoding of multiple textures [6]. The disparity or depth maps were computed and used for encoding and never stored by these efforts. This idea was extended to dynamic scenes for isolating and compressing video objects [12]. They used an MPEG-like algorithm to compress multiple video data using temporal correlation within one video stream and spatial correlation using depth values across views. They, however, treated depth maps as uncompressible and encoded it in a separate layer without any loss. An attempt to compress multiple depth streams of a scene encoded color and depth streams using motion vectors [4]. They concluded that separate motion vectors to encode color and depth perform better than a common motion vector.

The notion of proxy-based compression of depth images of static scenes was introduced in [10]. They used a geometric proxy model as the common structure of the scene. The proxy was projected to each stream to obtain the proxy depth maps. These were subtracted from the input depth maps of the scene to obtain the residues, which were compressed and transmitted to the rendering client. Simple triangulated proxies and parametric proxies like ellipsoids provided good compression and quality on static 3D scenes.

We extend the proxy-based compression to dynamic scenes or depth movies. Figure 2 presents the overview of the scheme for human models. The input to the system is the

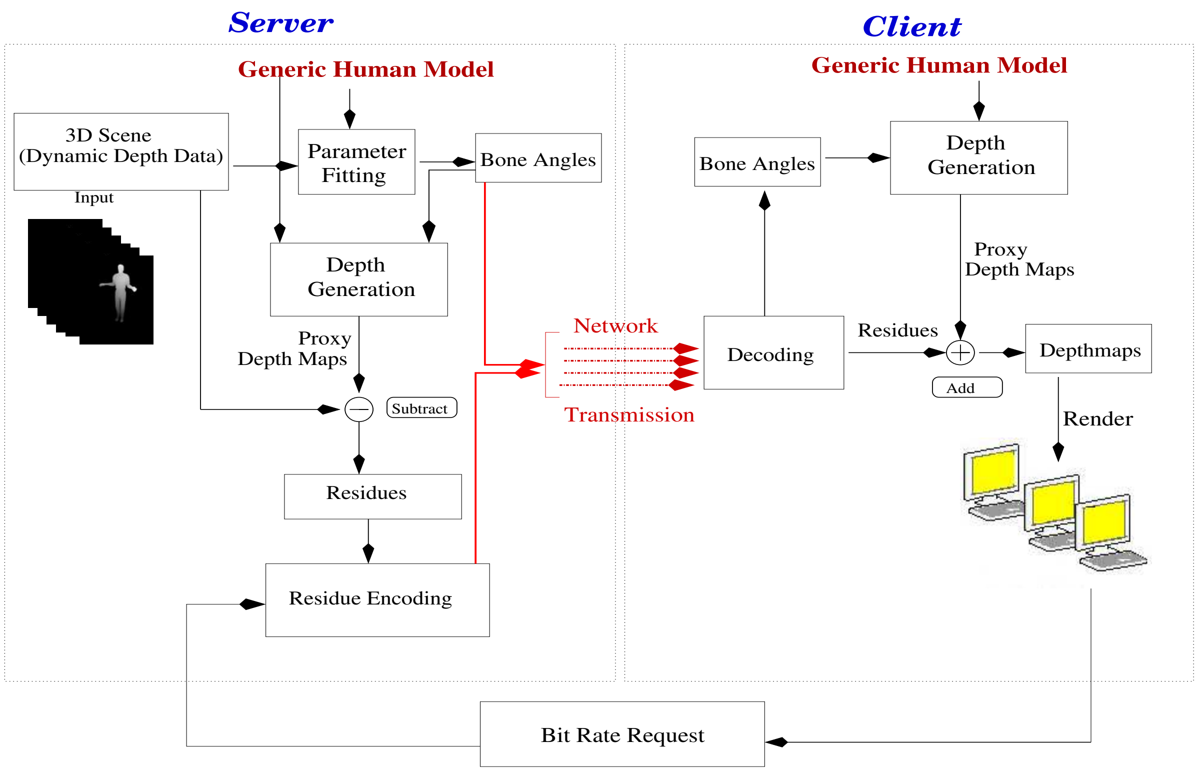

Figure 2: Overview of 3D Depth Movie compression and transmission

multiple depth movies of a scene with a human performing some action. We first fit an articulated proxy model to the point cloud for each frame. The output of this process is the bone angles of the model, which are the parameters of the proxy. The fitted proxy for each frame is projected to each view and the resulting proxy depth maps are used as a prediction of the input depth maps. The residue or prediction error between the original and the fitted proxy model is calculated by subtracting the predicted proxy from the input depth maps. The residues are encoded and sent to the client along with bone-angle parameters. At the client's side, the bone parameters are applied on the articulated model to get the proxy depth maps. Residues are added to these depth maps to get the real depth for rendering. Figure 3 shows some sample depth maps, point clouds using all views, and the fitted articulated human model.

3 Proxy-Based Compression of Depth Movies

We consider dynamic scenes involving a human performer and use an articulated, parametric human model as the proxy. The joint-angles are the free variables of the model and serve as the parameters of the model. Any position of the human performer can be approximated using appropriate set of parameters and serve as a common "prediction" for all views for each time instant.

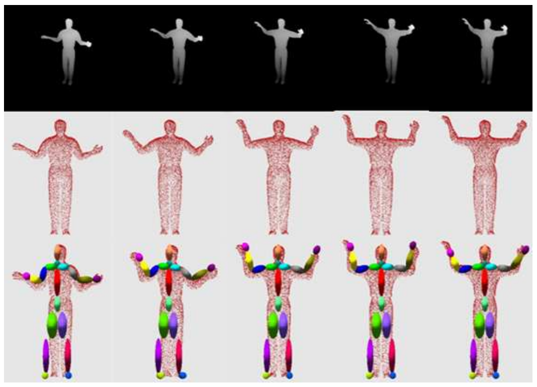

Figure 3: Articulated model fitting: Top row shows the depth maps for 5 frames, middle row shows the corresponding point clouds for each frame and the bottom row shows the corresponding fitted articulated model.

Algorithm 1 Encoding depth maps for n frames from m views/cameras

```
Input: $n$ depth maps from each of $m$ cameras, $k$ for encoding.
Output: $mn$ residues, $n$ parameter values (one per frame)
1: Fit the proxy model to the point cloud for each frame and recover the bone-angle
   parameters.
2: Compute the mask image for each frame of each view.
3: Project the fitted proxy to each view and compute the proxy depth maps.
4: Compute the residue $R$ by subtracting proxy depth from input depth.
5: Encode $R$ for the given bit rate as $R'$.
6: Run-length encode on mask images.
```

3.1 Parametric Proxy for Human Models

We generate a proxy representation of the depth movies using a common articulated parameterized model. The parameters are joint angles that change from one frame to another through the movie, but are same for all views of each frame. The articulated model we use has 18 vertices as joints. Each bone position needs 3 parameters for the angles as heading (h), pitch (p) and roll (r). Thus, for each frame, a total of $18 \times 3 = 54$ parameters define the model completely. A triangulated skin-model is used to generate the realistic human model from the skeletal structure. The skeletal and skinned model are available to both the encoder and the decoder. Thus, only a few bytes to fix the 54 parameters have to be sent for each unique pose or frame of the scene. The same model can be reconstructed at the client using these angles.

3.2 Computing the Proxy Model

The input consists of mn depth maps for n frames from m views. We unproject the depth maps from each camera to get a point cloud for each frame. The articulated model is fit then to the point cloud. The fitting can be performed by optimizing the error between the skinned model and the point cloud. This is a computationally intensive process but can give good results. For this paper, we use a semi-automatic tool to fit a skinned model to the point cloud interactively. Our tool enables fitting of one frame in a few seconds using simultaneous display of the model and the points. In the end, the parameters of the articulated model for each frame represent the scene parametrically as a proxy.

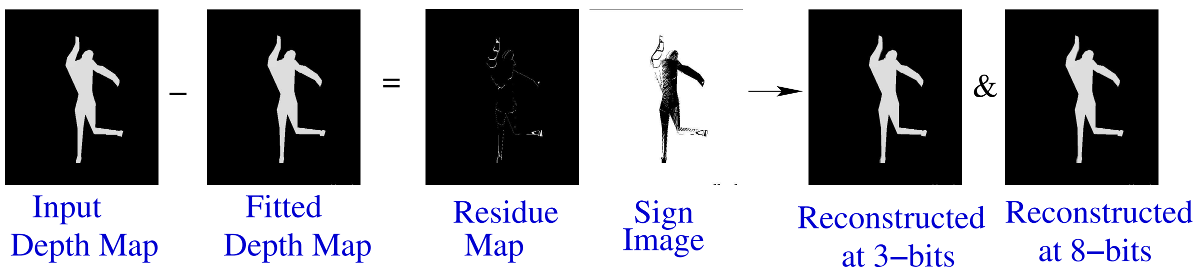

Figure 4: The input depth map, the fitted depth map, the generated residue and sign image, the reconstructed depth maps at 3-bits and 8-bits

3.3 Residue Computation

The fitted proxy model – consisting of the generic skinned model and the bone-angle parameters – is projected to each view to get its depth map and the mask bit. The mask helps identify the true projection of the subject so that the computation can be restricted to it. The difference between the actual input depth map and the proxy depth map is stored as the residue or the prediction error for each frame. The sign bit and the magnitude bits of the residue are stored separately. Thus, we have mn residues and sign images for each frame and view. Figure 4 shows the Input depth map, the fitted depth map and the difference between the two as the residue map and the sign image.

3.4 Residue Encoding

The residues are small in value if the proxy model fits the original model well. Residues are also more correlated since the difference of proxy and original depths vary smoothly from one frame to another. Hence, residues compress better than the depth maps. We use two compression schemes for the residues. The first one encodes the successive residue-maps of each depth stream using standard MPEG compression scheme, giving m residue movies. This can suffer in quality and can incur high reconstruction costs. The other method performs bit-plane encoding of the residues, using as many bits as client demands. Thus, the most significant k-bits represent each residue value. Mask and sign bits of each frame are run-length encoded.

3.5 Compressed Representation

The representation to be sent to the rendering client includes the following. (a) The bone-angle parameters for each frame, (b) the mask bits for each view and each frame, (c) m MPEG streams for the residue values when using MPEG or (c) the bit-plane encoded residues for each frame and each view when using bit-plane encoding. All data is zipped together at the end as a simple entropy-encoding method. This information is sent to the rendering client, which decodes the depths back.

3.6 Decoding

Decoding at the client is done as given in Algorithm 2. First, the encoded residues are extracted along with the bone parameters. These parameters are used to generate the proxy depth maps of the articulated model. The residues are then added to them to generate the reconstructed depth maps of the scene at the client side. This can be used for immersive rendering of the scene along with its appearance. Fig 4 shows the decoded depth maps after adding 3-bits and 8-bits of depth maps generated using bone parameters at the client side.

Algorithm 2 Decoding at the client side for $\mathbf{i} = \mathbf{1}, \mathbf{2}, \cdots, \mathbf{n}$ frames

Input: mn residues, mn masks and n bone-angle parameters
Output: mn depth maps for each view/frame for rendering

```
1: Get the bone-angle parameters for the frame from the input stream.
2: Capture the proxy depth maps $D$ using the parametric proxy model for each frame
   and view.
3: Decode the sign bit and mask bit.
4: Compute the residue $R$ for each frame and view.
5: $D' \leftarrow D + R$. Use $D'$ to render the 3D scene.
```

4 Experiments and Results

In this section, we present the results of compressing multiple depth movies of a dynamic human action using proxies. Since real data is not available in plenty, we use synthetic data to develop and test our algorithm. We use freely available MOCAP (Motion Capture) data to generate synthetic data by giving real motion to synthetic models. The input depth maps are generated for the MOCAP data by articulating a skinned proxy model with MOCAP parameters. These depth maps have a 8-bit representation. To simulate real-life situations, we add two kinds of noise to the process. First, some noise is added to the bone angles to simulate bad fitting. Second, noise is added to the depth map to simulate errors in the capture process. The depth noise has a small random component at each pixel which rides on top of a larger component that varies slowly over the whole depth map.

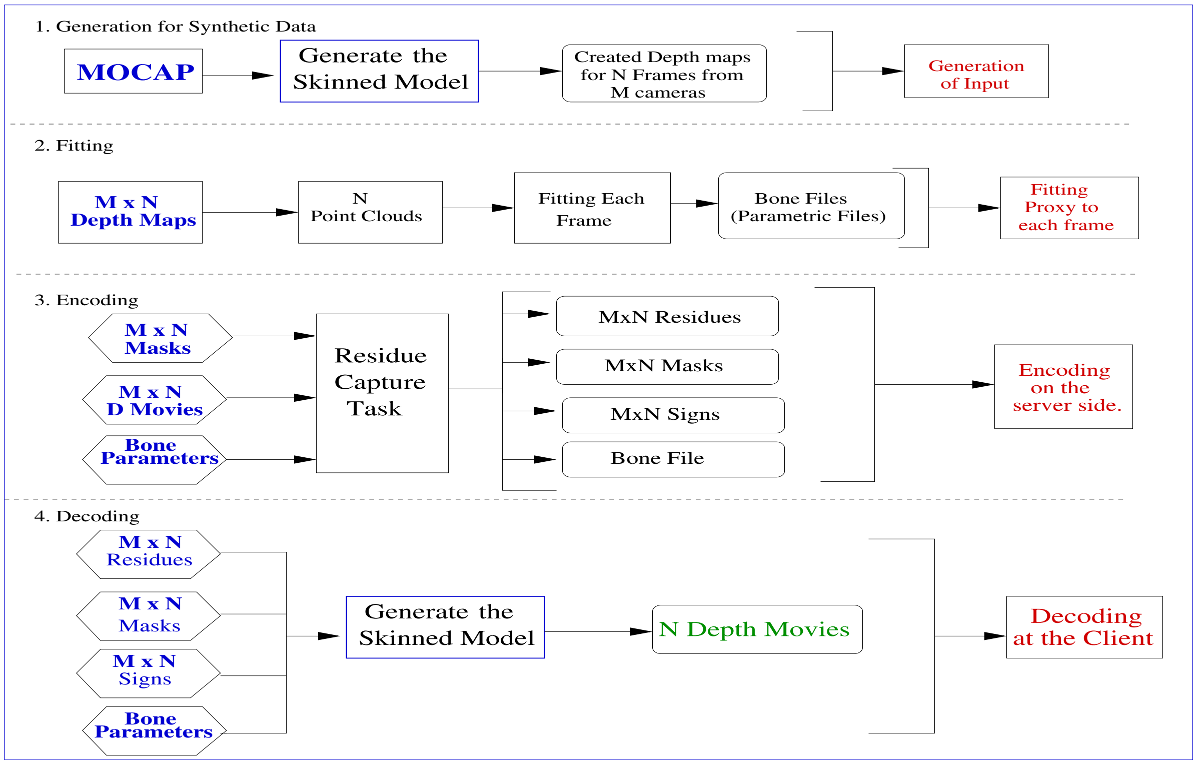

Figure 5: Steps for Encoding and Decoding at Server and Client

The MOCAP data and the bone angle parameters are the same, except for the bone noise. The proxy model is articulated using the noisy bone angles to generate the depth maps. Residues are generated by subtracting the fitted proxy depth maps from the input depth maps, which uses the MOCAP data without noise. These residues are compressed using both schemes and with varying number of bits.

For decoding, bone parameters are given to the proxy model available to the client to get the proxy depth maps. The decoded residues are added to approximate the original depth maps. The quality of the reconstruction is governed by the quality of the residues. If residues are encoded losslessly, the original depth maps can be reconstructed. Lossy encoding using MPEG on the residues or fewer number of bits will distort the model. For comparison, we also encode the original depth maps directly using MPEG and compare the reconstruction results for all three options.

Results on a few synthetic datasets are shown in Table 1. We experimented on 5 datasets with 100 to 300 frames. The compression ratio is with respect to the original, uncompressed depth maps. The PSNR is calculated by comparing the reconstructed depth maps with the input depth maps. The residue MPEG compression exploits the spatial and temporal redundancy in the data so sometimes it improves over the proxy based compression scheme. The bit-plane scheme provides high compression and moderate quality at low number of bits and good compression and excellent quality at higher number of bits. It provides totally random access of the depths of individual frames. Most interestingly, the option of using 0 bits of residue provides a very low bit-rate ap-

#bits used	Throw (100 frames)	Spin Kick (100 fr.)	Break dance (120 fr.)	Dance (230 fr.)	FootBall (300 fr.)
0	10584/10.11	19859/10.59	12563/9.35	11309/10.33	14391/12.34
1	4927/24.56	9354/25.56	5397/23.31	6371/22.60	6927/26.07
2	4839/25.03	9247/26.85	5036/24.56	6005/23.91	6543/28.11
3	4737/25.41	9218/28.31	4821/25.91	5941/24.35	6307/29.67
4	4650/25.52	9137/29.92	4598/26.74	5803/25.69	6251/30.93
5	4542/26.82	9116/31.53	4512/28.34	5749/26.78	6201/31.16
6	4458/28.93	9069/32.63	4439/29.16	5710/28.54	6149/33.32
7	4386/30.91	9032/34.17	4387/29.55	5673/29.01	6111/34.51
8	4257/32.57	8959/35.09	4297/29.97	5592/29.36	6021/36.93
MPEG-R	4239/26.49	9129/29.43	4353/26.84	5549/25.81	5945/29.91
MPEG-D	4154/24.58	8938/28.23	4155/25.35	5302/25.96	5713/27.82

Table 1: Compression ratios and PSNR for different datasets with varying bit-wise compression, MPEG encoding of the residues (MPEG-R) and MPEG encoding of the input depth maps (MPEG-D). The first number is the compression ratio and the second the PSNR.

proximation of the input scene. The error of such approximation is somewhat high as the reconstructed shape at the client will be that of the articulated model. Figure 6 plots the PSNR and the compression ratio against the number of bits used to encode the residues for one dataset. It can be seen that the PSNR varies slowly with the number of bits, but the compression ratio of bit-plane encoding is very good. The MPEG compression of depth and residues (MPEG-R and MPEG-D in Table 1) provides decent compression and quality, but the bit-plane encoding scheme provides more size to quality tradeoffs to suit any situation.

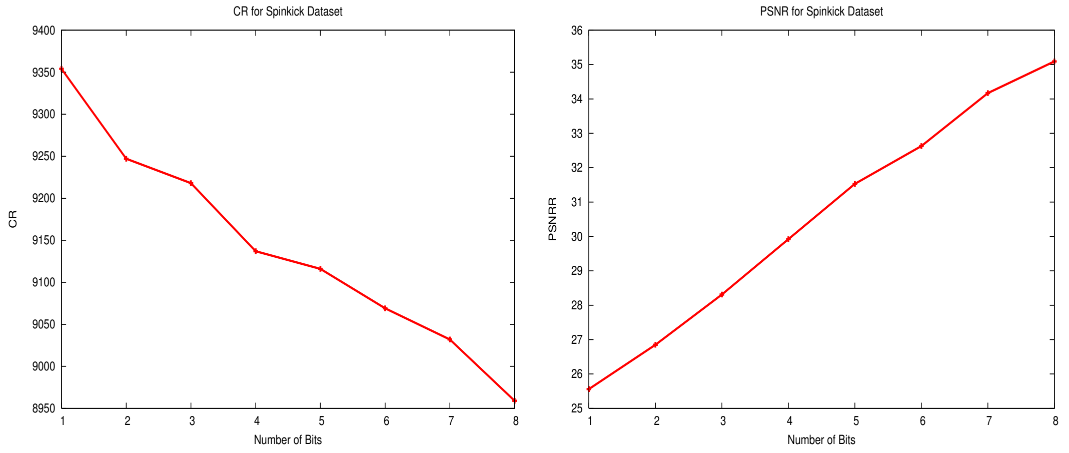

Figure 6: Results for 1-8 bit-wise encoding on Spinkick dataset with 100 frames

5 Conclusion

We presented a proxy-based compression scheme for multiple depth movies of a scene involving dynamic human action in this paper. We used a simple articulated model as the proxy and joint angles as the parameters that approximate the model for a particular frame. The scheme provides good compression ratios and at acceptable quality levels. The proxy-based scheme provides several controls on the amount of data to be sent. This makes it ideal for sending the captured data for applications like 3D teleconferencing.

References

[1] J. Carranza, C. Theobalt, M. A. Magnor, and H.-P. Seidel. Free-viewpoint video of human actors. *ACM Trans. Graph.*, 22(3):569–577, 2003.

[2] M. H. Gross, S. Würmlin, M. Näf, E. Lamboray, C. P. Spagno, A. M. Kunz, E. Koller-Meier, T. Svoboda, L. J. V. Gool, S. Lang, K. Strehlke, A. V. Moere, and O. G. Staadt. blue-c: a spatially immersive display and 3d video portal for telepresence. *ACM Trans. Graph.*, 22(3):819–827, 2003.

[3] R. Krishnamurthy, B. Chai, H. Tao, and S. Sethuraman. Compression and transmission of depth maps for image-based rendering. In *ICIP01*, pages III: 828–831, 2001.

[4] S. Kum and K. MayerPatel. Intra-stream encoding for multiple depth streams. *NOSSDAV*, pages 62–67, 2006.

[5] L. Levkovich-Maslyuk, A. Ignatenko, A. Zhirkov, A. Konushin, I. K. Park, M. Han, and Y. Bayakovski. Depth image-based representation and compression for static and animated 3-d objects. *IEEE Trans. Circuits Syst. Video Techn.*, 14(7):1032–1045, 2004.

[6] 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.

[7] W. Matusik and H. Pfister. 3d tv: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. *ACM Transactions on Graphics*, 23(3):814–824, Aug. 2004.

[8] P. J. Narayanan, S. K. Penta, and S. R. K. Depth+texture representation for image based rendering. In *ICVGIP*, pages 113–118, 2004.

[9] P. J. Narayanan, P. Rander, and T. Kanade. Constructing virtual worlds using dense stereo. In *ICCV*, pages 3–10, 1998.

[10] S. K. Penta and P. J. Narayanan. Compression of multiple depth maps for ibr. *The Visual Computer*, 21(8-10):611–618, 2005.

[11] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs. The office of the future: A unified approach to image-based modeling and spatially immersive displays. *Computer Graphics*, 32:179–188, 1998.

[12] Q. Wu, K. T. Ng, S.-C. Chan, and H.-Y. Shum. On object-based compression for a class of dynamic image-based representations. In *ICIP (3)*, pages 405–408, 2005.

[13] S. Würmlin, E. Lamboray, O. G. Staadt, and M. H. Gross. 3d video recorder: a system for recording and playing free-viewpoint video. *Comput. Graph. Forum*, 22(2):181–194, 2003.

[14] Z. Yang, K. Nahrstedt, Y. Cui, B. Yu, J. Liang, S.-H. Jung, and R. Bajcsy. Teeve: The next generation architecture for tele-immersive environment. In *ISM*, pages 112–119, 2005.

[15] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. A. J. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. *ACM Trans. Graph.*, 23(3):600–608, 2004.