Dynamic point cloud compression for free viewpoint video

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

cgl computer graphics lab
Institute of Computational Science

CS Technical Report #430, December 10, 2003

Dynamic Point Cloud Compression for Free Viewpoint Video

Edouard Lamboray, Michael Waschbüsch, Stephan Würmlin, Hanspeter Pfister*, Markus Gross

Computer Science Department
ETH Zurich, Switzerland
e-mail: {lamboray, waschbuesch, wuermlin,grossm}@inf.ethz.ch
http://graphics.ethz.ch/

*MERL - Mitsubishi Electric Research Laboratories
Cambridge, MA, USA
e-mail: pfister@merl.com
http://www.merl.com/

Dynamic Point Cloud Compression for Free Viewpoint Video

Edouard Lamboray Michael Waschbüsch Stephan Würmlin Hanspeter Pfister* Markus Gross
Computer Graphics Laboratory, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland
**MERL - Mitsubishi Electric Research Laboratories, Cambridge, MA, USA*
{lamboray, waschbuesch, wuermlin, grossm}@inf.ethz.ch pfister@merl.com

Abstract

In this paper, we present a coding framework addressing the compression of dynamic 3D point clouds which represent real world objects and which result from a video acquisition using multiple cameras. The encoding is performed as an off-line process and is not time-critical. The decoding however, must allow for real-time rendering of the dynamic 3D point cloud. We introduce a compression framework which encodes multiple attributes like depth and color of 3D video fragments into progressive streams. The reference data structure is aligned on the original camera input images and thus allows for easy view-dependent decoding. The separate encoding of the object's silhouette allows the use of shape-adaptive compression algorithms. A novel differential coding approach permits random access in constant time throughout the complete data set and thus enables true free viewpoint video.

1. Introduction

In the past, many 3D reconstruction methods have been investigation but no method was presented which is tailored to streaming and compression of free viewpoint video. In this paper, we present a compression framework for 3D video fragments which is a dynamic point based representation tailored for real-time streaming and display of free viewpoint videos [14]. Our representation generalizes 2D video pixels towards 3D irregular point samples and thus we combine the simplicity of conventional 2D video processing with the power of more complex polygonal representations for free viewpoint video. 3D video fragments are generic in the sense that they work with any real-time 3D reconstruction method which extracts depth from images. Thus the representation is quite complementary to model-based scene reconstruction methods using volumetric (e.g. space carving, voxel coloring), polygonal (e.g. polygonal visual hulls) or image-based (e.g. image-based visual hulls) scene reconstruction. Therefore, the framework can be used as a nice abstraction of the 3D video representation, its streaming and compression from 3D reconstruction.

2. Problem analysis

2.1. Prerequisites

The input data of free viewpoint video systems acquiring real-world objects typically consists of multiple concentric video sequences of the same scene. Before recording, the cameras have been calibrated and for each camera intrinsic and extrinsic calibration parameters are available to both encoder and decoder. The multiple 2D video sequences providing the input data are recorded with synchronized cameras. Additionally, we have at our disposal for every input frame an image mask, telling which pixels belong to the object of interest, i.e. are foreground pixels, and which pixels belong to the background.

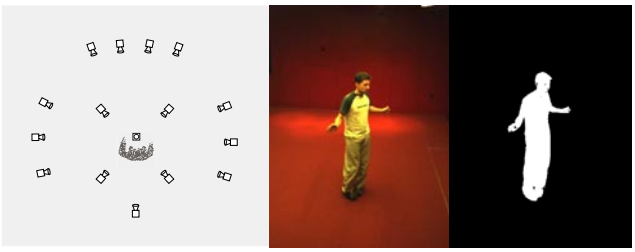


Figure 1: Input data: a) Concentric camera setup. b) Example image from one camera. c) Segmentation mask of image b).

After the input images have been processed by a 3D reconstruction algorithm, which is not specified in this document, each input frame provides for each foreground pixel a depth value describing, in combination with the camera calibration parameters, the geometry of the object, and a color value. For each foreground pixel, a surface normal vector and a splat size can be encoded as optional attributes. In general, it is possible to encode any attributes describing the visual appearance of an object. Given a specific rendering scheme or target application, any subset of the above pixel attributes might be sufficient.

In summary, our compression framework can be applied to every static or dynamic 3D data set which can be completely described by a set of concentric 2D views. In practice, this applies to all acquisition setups for 3D reconstruction of real world objects, e.g. a 3D scanner provides only one single image, but our coding framework still applies. Furthermore, the coding framework can smoothly be extended to an arbitrary camera setup where only a few cameras see the object at a time.

2.2. Application domains

Our streaming format for dynamic 3D point clouds should address the application domains depicted in Figure 2. A high compression ratio is efficient if the 3D point cloud is transmitted via a low bandwidth network. But a

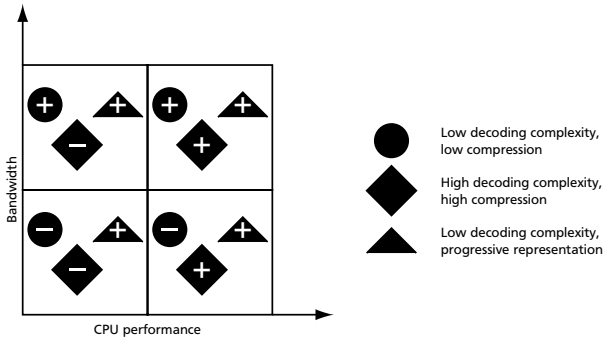


Figure 2: Application domains for streaming free viewpoint video.

high compression ratio is difficult to achieve at a low decoding complexity. In our opinion, a 3D compression framework should allow for decoding at a relatively low complexity, and thus support a wide range of target hardware devices. The problem of low bandwidth transmissions should be addressed by building upon a progressive representation of the data. In fact, bandwidth and CPU performance are often correlated, e.g. high-end computing nodes have, in general, access to a broadband network connection and nodes with a low bandwidth network access have also limited processing power.

2.3. Goals

The goal of our research is compression of free viewpoint video. In this paper, we distinguish between constrained and unconstrained free viewpoint video.

Definition 1: *Constrained free viewpoint video*

After decoding, the point cloud can be rendered from any possible direction, but either only small viewpoint changes are allowed during rendering, or discontinuities in rendering are tolerated in presence of large viewpoint changes.

Definition 2: *Unconstrained free viewpoint video*

After decoding, the point cloud can be rendered from any possible direction, the viewpoint being a function of the rendering time and the discontinuities during rendering are minimized.

Examples of spatio-temporal viewpoint trajectories are depicted in Figure 3. In the constrained case, the trajectory is constrained to a narrow band. In the unconstrained case, the trajectory can be arbitrary.

Taking into account that the number of cameras is potentially large and that real-time decoding is required, it is not realistic on current hardware to decode all the cameras first, and then to render the scene for a given viewpoint. Hence, it is necessary to reduce the set of processed cameras already during decoding. Thus, view dependent decoding must be supported by the proposed coding framework. In the remainder of this section, we define view dependent decoding.

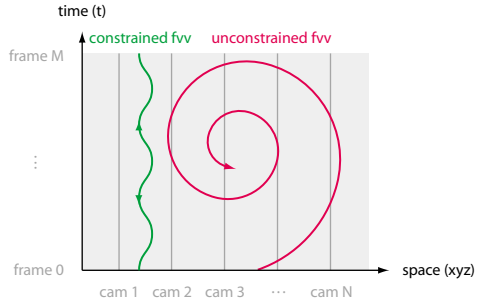


Figure 3: Possible viewpoint trajectories for constrained and unconstrained free viewpoint video.

Consider the following variables:

t	:	the recording time, which is a discrete time corresponding to the frame rate of the recording cameras.
θ	:	the rendering time which is completely asynchronous to the recording time t and has a potentially much higher frame rate.
$v(\theta)$	:	the viewpoint as a function of the rendering time
$D(v(\theta), t)$	:	the set of data decoded for a given viewpoint $v(\theta)$ and a timestamp t
$R(v(\theta), t)$	:	the set of data decoded for a given viewpoint $v(\theta)$ and a timestamp t which becomes visible after rendering

Note that $D(v(\theta), t)$ is the result after decoding and $R(v(\theta), t)$ is the result after rendering. In any case, we have $R \subseteq D$.

Definition 3: *Optimal view dependent decoding*

Optimal view dependent decoding is achieved if $R(v(\theta), t) = D(v(\theta), t)$.

This implies that the decoder, for a given rendering frame, only decodes information of the corresponding recording time frame which becomes visible in the final rendering. This strong condition can be relaxed using a weaker formulation:

Definition 4: *View dependent decoding*

View dependent decoding minimizes the cardinal of the complement $D(v(\theta), t) \backslash R(v(\theta), t)$.

This implies that the decoder, for a given rendering frame, maximizes the ratio of decoded information of the corresponding recording time frame which is visible in the final rendering versus the total amount of decoded information for the given rendering instant.

2.4. Playback features

The 3D streaming data format should address the following features:

- Multi-resolution: scalability and progressivity with respect to resolution. This feature can be achieved using either a progressive encoding of the data, as proposed by the embedded zero-tree wavelet coding algorithm (EZW) [12], progressive JPEG or progressive sampling of the data, as proposed in [14]. The progressive encoding approach is more promising than the progressive sampling.
- Multi-rate: scalability with respect to time, i.e. playback of the sequence is possible at a different frame rate than the recording frame rate. Backward playback should also be possible.
- View-dependent decoding: In this paper, we address the problem of encoding data for view-dependent decoding. The algorithm for deciding which cameras are required for the view-dependent decoding of a given rendering frame is similar to the technique described in [14], i.e., given a viewpoint and the camera calibration data, we compute the contributing cameras and read the data accordingly before decoding.

3. Compression

Compression is achieved by exploiting coherence in the 3D point cloud data. In our compression framework, which uses the recorded camera images as input data structure, coherence can be possibly exploited in image space, in camera space and in the time dimension.

3.1. Coherence in image space

Similar to standard image compression algorithms, 2D transforms can be employed to our input data and, thus, the 2D coherence in the data is exploited.

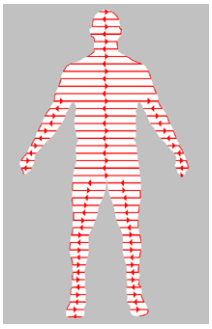


Figure 4: Scan line traversal of pixels inside a silhouette.

However, we are only interested in a part of the image which lies within the silhouette. We propose a shape-adaptive wavelet encoder, which puts the colors of the relevant pixels into a linear order by traversing the silhouette scan line by scan line in alternating directions, see Figure 4. We then apply a one-dimensional wavelet transform to this linear list of coefficients using the lifting scheme. The resulting wavelet coefficients are lossily encoded up to a desired bit rate by a one-dimensional zero-tree algorithm and finally compressed with an arithmetic coder. To allow for a correct decompression, the silhouette must be stored losslessly along with the compressed data. The compression performance of the 1D and 2D approaches is analyzed in Section 6.1.

3.2. Coherence in time

In the case of constrained free viewpoint video, we can reasonably admit that in most case the sequence is played back with increasing t and at normal playback speed. Hence, we can use the information from previous frames for building the current frame.

For each camera i, $c_i(t)$ is the decoding function which returns the contribution to the 3D point cloud of the respective camera and for the time t. If temporal coherence is exploited by using the information from previous frames, the decoding function $c_i(t)$ thus has the form

$$c_i(t) = c_i(t') + \Delta c_i(t) \qquad (1)$$

with $t' < t$ and where $\Delta c_i(t)$ describes the specific contribution of frame t.

Note that all popular 2D video compression algorithms belong to this class of codecs. This approach is also feasible for constrained free viewpoint video according to Definition 1.

In case of unconstrained free viewpoint video, however, it is more difficult to exploit temporal coherence. The decoder is supposed to implement a function f, which returns a 3D point cloud at any time instant θ during rendering. This implies a viewpoint $v(\theta)$ and a mapping function $t = m(\theta)$ which maps the rendering time to the recording time. A weight function $w(v)$ tells us, given the viewpoint v, which cameras contribute to the visible part of the 3D point cloud. In a first approximation, we can assume that $w(v)$ returns 1 if a camera has a visible contribution and 0 if not.

We get

$$f(v(\theta), t) = w(v(\theta)) \cdot f(t)$$

$$= \begin{bmatrix} w_0(\theta) & \dots & w_{N-1}(\theta) \end{bmatrix} \cdot \begin{bmatrix} c_0(t) \\ \dots \\ c_{N-1}(t) \end{bmatrix}$$

Assume $\theta' = m^{-1}(t')$. If $w_i(\theta') = 0$, the decoding of $c_i(t)$ requires the decoding of $c_i(t')$ with $t \neq t'$ and the condition of view-dependent decoding is violated. Hence, optimal view-dependent decoding can only be implemented using decoders which can be defined as

$$c_i(t) = C_i + \Delta c_i(t) \qquad (2)$$

with C_i independent of t.

Thus, a decoder supporting unconstrained free viewpoint video needs to implement decoding in constant time for frames addressed in a random order.

3.3. Coherence in camera space

First we discuss the coherence in camera space for the constrained free viewpoint case where $v(\theta) = v$.

If camera coherence is exploited, the decoding function has the form

$$c_i(t) = c_j(t) + \Delta c_i(t) \text{ with } i \neq j. \qquad (3)$$

In general, we need the data from two or three cameras for rendering a 3D point cloud from any arbitrary viewpoint. However, unlike in the time dimension, we do not have a preferential direction in space, and it is not possible

to make a reasonable prediction during encoding, from which viewpoints the point cloud will preferentially be rendered. Hence, the combination of cameras for exploiting camera space coherence can only be done in an arbitrary fashion, or, at best, combining the cameras such that an optimal compression ratio is achieved. However, this combination of cameras can again lead to unfortunate configurations, in which several cameras, which are not directly visible for $v(\theta)$ need to be decoded.

As in the time dimension, the problem becomes even more difficult for unconstrained free viewpoint decoding, if we need to calculate (3) and we additionally have $w_j(\theta') = 0$.

From the precedent analysis, we conclude that traditional 2D video coding algorithms do not fulfill the requirements of unconstrained free viewpoint video. In Section 4, we propose a coding scheme for constrained free viewpoint video which uses conventional video coding algorithms and introduce a new coding scheme for unconstrained free viewpoint video which follows the guidelines of Equation (2), i.e. allows spatio-temporal random access in constant time.

4. Compression framework

The underlying data representation of our free viewpoint video format is a dynamic point cloud, in which each point has a set of attributes. Since the point attributes are separately stored and compressed, a referencing scheme, allowing for the unique identification between points and their attributes is mandatory. Using the camera images as building elements of the data structure, each point is uniquely identified by its position in image space and its camera identifier. Furthermore, looking separately at each camera image, we are only interested in foreground pixels, which contribute to the point cloud describing the 3D object.

Thus, we use the segmentation mask from the camera images as reference for all subsequent coding schemes. In order to avoid shifts and wrong associations of attributes and points, a lossless encoding of the segmentation mask is required. This lossless segmentation mask must be at the disposal of all encoder and decoders. However, all pixel attributes can be encoded by a lossy scheme. Nevertheless, a lossless or almost lossless decoding should be possible if all data is available. The stream finally consists of key frames and delta frames, which rely upon a prediction based on the closest key frame. The key frame coding and the prediction is different in case we want to encode constrained or unconstrained free viewpoint video.

The overall compression framework is depicted in Figure 5. From the segmentation masks and the camera calibration data, a geometric reconstruction of the object of interest is computed. The output of the geometric reconstruction are 3D positions, surface normal vectors, and splat sizes. Note that any geometric reconstruction scheme which is able to deliver these streams from the provided input data can be used. The data streams are compressed and, along with the texture information and the segmentation masks, multiplexed into an embedded, progressive free viewpoint video stream. The camera calibration data is encoded as side information. In the following sections, the specific coders are described in detail.

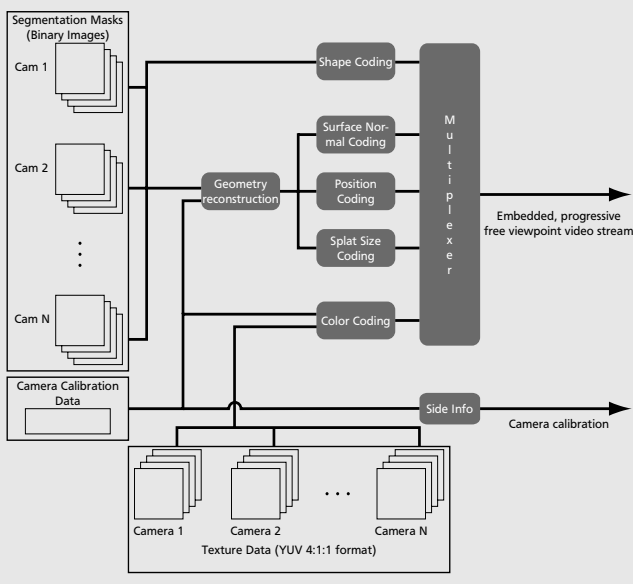


Figure 5: Compression framework.

Note that all color coding is performed in the 4:1:1 YUV format. The compression performance results as suggested by the MPEG standardization committee should be generated in YUV color space [5]. However, our framework is also capable of handling texture data in other formats. Furthermore, instead of computing and encoding the surface normal vectors and the splat sizes during the reconstruction and encoding process, these attributes can also be evaluated in real-time during rendering. In that case, no explicit encoding for surface normals and splat sizes is necessary.

5. Free viewpoint video coding

5.1. Constrained free viewpoint video

Our compression framework can use existing MPEG coders in the case of constrained free viewpoint video coding. The silhouettes are encoded using lossless binary shape coding [3]. The depth values are quantized as luminance values and a conventional MPEG-4 video object codec can be utilized. Surface normal vectors can be progressively encoded using an octahedron subdivision of the unit sphere [1]. The splat sizes can simply be quantized to one byte and the codewords are represented in the gray scale MPEG video object.

The complete decoding of one constrained free viewpoint video frame requires for each reconstruction view three gray scale MPEG video objects (depth, surface normal, splat size) and one color video object, i.e. if two reconstruction views are used a total of six gray scale video objects and two color video objects.

5.2. Unconstrained free viewpoint video

For unconstrained free viewpoint video we propose a coding scheme which uses averaged information as key frames. The principle of average coding is depicted in Figure 6. In each foreground pixel of a time window, an average value for each attribute is computed. This information becomes the key frame, which is, within the respective

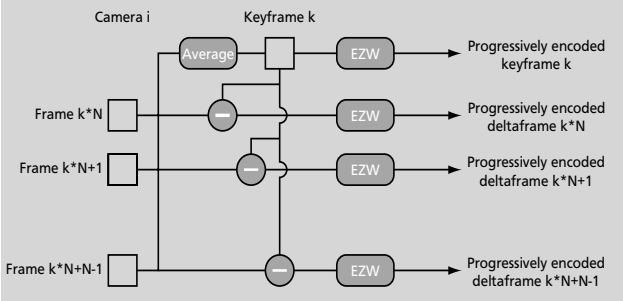


Figure 6: Principle of average coding

time window, independent from the recording time. The delta frame is simply the difference information of the original frame and the respective key frame, It is true that for every window of N frames which use the same key frame, we need to encode N+1 image frames. However, we can use pretty high values for N and thus, the additional cost of coding the key frame is distributed over a large number of delta frames.

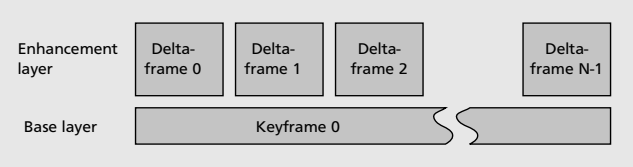


Figure 7: Decomposition of a stream into base and enhancement layers

The streams can thus be decomposed into a base layer which contains the averaged key frames, and an enhancement layer composed by the delta frames which allow to decode the complete information for every frame, see Figure 7. It appears that the key frames are only independent of the rendering time inside each window. But the windows can cover a large number of frames without limiting the navigability of the stream, as it is the case with coders that follow the scheme of Equation (1).

Figure 10 illustrates the principle of average coding using a window of texture data of one single camera.

The silhouettes are again encoded using lossless MPEG-4 binary shape coding [3]. All point attributes are encoded using the average coding scheme: the key frame averages the attributes of the window; the delta frame contains the difference image with respect to the key frame.

The depth values are encoded using the shape-adaptive 1D wavelet approach described in Section 3.1 and the colors are encoded using the embedded zero-tree wavelet algorithm for image compression [12].

If we assume large windows, the decoding of the key frames can be neglected and the complexity is about the same than in the constrained free viewpoint case. If surface normals and splats are determined during rendering, we need for each reconstruction camera one binary shape image, one gray scale image and one color image.

5.3. Multiplexing

Since all attributes are encoded independently and progressively, a stream satisfying a given target bit rate can be obtained by multiplexing the individual attribute bit streams into one bit stream for transfer. Appropriate contributions of the single attribute bit streams are determined according to desired rate-distortion characteristics.

For example, a bit stream of 300 kbps may contain 30 kbps of shape information, 60 kbps of position information, 120 kbps of color information, 45 kbps of surface normal information and 45 kbps of splat size information.

5.4. Extension to full dynamic 3D scenes

The encoders, described so far for video objects, can also be used to encode entire dynamic scenes. Distinct objects in the scene can be encoded in different layers. Static objects are described by a single key frame. A scene graph, which is stored as side information, describes the spatial relations between the different layers. View dependent decoding is again enabled by decoding only those layers which are visible from the current arbitrary viewpoint.

6. Preliminary results

6.1. Wavelet filters for depth image compression

We investigated the rate-distortion function of the EZW image compression algorithm for different wavelet filters applied to a depth image. The results are presented in Figure 8 and Figure 9. As input we used a 512x512 depth image generated from a synthetic 3D human body. The bit rate is given in bits per surfel, where we consider a foreground pixel in the depth image as surfel. Our example depth image contained 40k surfels.

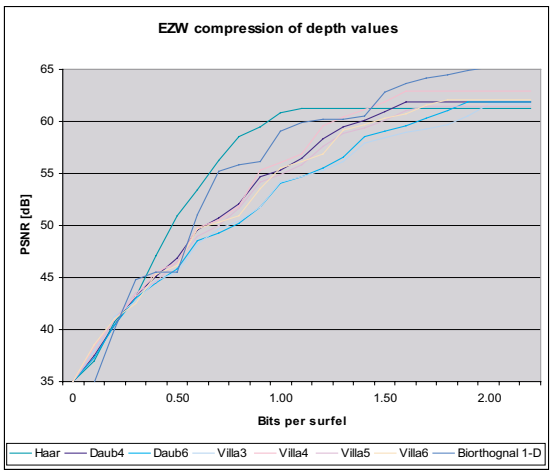


Figure 8: Rate-distortion diagram for different EZW encoded wavelet transform coefficients of a synthetic depth image.

6.2. Bit rates for unconstrained free viewpoint video

We compressed an MPEG test data sequence with our unconstrained free viewpoint video codec [5]. The test sequence consists of input data from 25 different camera positions, each with a resolution of 320x240 pixels. We assume a frame rate of 25 frames per second. The average key frame covers a window of 200 frames which corresponds to a time of 8 seconds. Only disparity and colors are coded, normals and splat radii are estimated on the fly during rendering. For view-dependent decoding and ren-

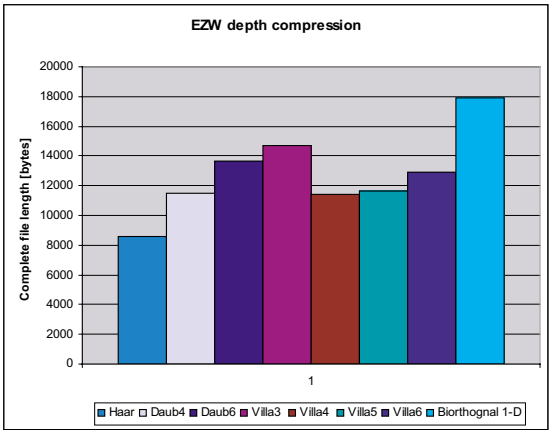


Figure 9: .File length for the different EZW encoded wavelet transform coefficients of the same synthetic depth image.

dering the three cameras nearest to the virtual viewpoint are chosen. The resulting data stream consists of approximately 8000 surfels per frame. We compressed it with several example bit rates specified in Table 1. To further improve the compression performance we use downscaled depth images which are re-expanded to full resolution during reconstruction. The silhouette images have to be compressed lossless which produces an additional amount of data of 0.3 bits per surfel. Snapshots from the respective sequences are shown in Figure 11.

Table 1: Example bit rates for unconstrained free viewpoint video with three reconstruction cameras.

IMAGE RESOLUTION	DEPTH BITS / SURFEL	COLOR BITS / SURFEL	TOTAL BIT RATE BITS / SECOND
107x80	0.166	0.7	350k
160x120	0.5	1.5	560k
320x240	1.5	1.5	740k
320x240	2.0	2.0	920k
320x240 (uncompressed)	8.0	8.0	10200k

7. Related work

In [14], we propose a point-based system for real-time 3D reconstruction, streaming and rendering which does not make any assumptions about the shape of the reconstructed object. Our approach uses point samples as a straightforward generalization of 2D video pixels into 3D space. Thus a point sample holds, additionally to its color, a number of geometrical attributes and a one-to-one relation between 3D points and foreground pixels in the respective 2D video images is guaranteed.

For shape coding, we rely on lossless coding techniques for binary images [3, 7]. The compression is based on a context-sensitive adaptive binary arithmetic coder.

In our free viewpoint video framework, images are compressed using a wavelet decomposition followed by zero-tree coding of the respective coefficients [11, 12]. Finally, an arithmetic entropy coding is applied [8]. Alternatively, MPEG-4 video object coding, based on shape adaptive discrete cosine transform coding, can be used [6].

Several coding techniques for large but static point representations have been proposed in the literature. Rusinkiewicz and Levoy presented Streaming QSplat [10], a view-dependent progressive transmission technique for a multi-resolution rendering system, which is based on a hierarchical bounding sphere data structure and splat rendering [9]. In [1], Botsch et al. use an octree data structure for storing point sampled geometry and show that typical data sets can be encoded with less than 5 bits per point for coding tree connectivity and geometry information. Including surface normal and color attributes, the authors report memory requirements between 8 and 13 bit per point. A similar compression performance is achieved by a progressive encoding scheme for isosurfaces using an adaptive octree and fine level placement of surface samples [4].

Briceno et al. propose to reorganize the data from dynamic 3D objects into 2D images [2]. This representation allows to deploy video compression techniques for coding animated meshes. This approach, however, fails for the case of unconstrained free viewpoint video.

Vedula et al. developed a free viewpoint video system based on the computation of a 3D scene flow and spatio-temporal view interpolation [13]. However, the coding of the 3D scene flow representation is not addressed.

8. Conclusions

This paper introduces a compression framework for free viewpoint video using a point-based data representation. The deployment of a novel average coding scheme enables for an unconstrained spatio-temporal navigation throughout the 3D video stream. All data is progressively encoded and hence a 3D video stream can be generated for different target bit rates.

In the future, additional components and investigations are required. The framework is independent of the 3D reconstruction method and we currently use a variant of the image-based visual hulls method, but other reconstruction algorithms should be investigated. Furthermore, the framework is independent of the actual attribute codecs, the only requirements on the codecs is the support for progressive decoding capabilities. The specific codecs also need to be investigated in detail. Finally, an algorithm optimizing the window length for average coding should be developed.

References

[1] M. Botsch, A. Wiratanaya, and L. Kobbelt. Efficient high quality rendering of point sampled geometry. In *Proceedings of the 13th Eurographics Workshop on Rendering*, pages 53–64, 2002.

[2] H. Briceno, P. Sander, L. McMillan, S. Gortler, and H. Hoppe. Geometry videos. In *Proceedings of ACM Symposium on Computer Animation 2003*, July 2003.

[3] A. K. Katsaggelos, L. P. Kondi, F. W. Meier, J. Ostermann, and G. M. Schuster. Mpeg-4 and rate-distortion-based shape-coding techniques. *Proceedings of the IEEE*, 86(6):1126–1154, June 1998.

[4] H. Lee, M. Desbrun, and P. Schroeder. Progressive encoding of complex isosurfaces. In *Proceedings of SIGGRAPH 03*, pages 471–475. ACM SIGGRAPH, July 2003.

[5] MPEG-3DAV. Description of exploration experiments in 3DAV. ISO/IEC JTC1/SC29/WG11 N5700, July 2003.

[6] J. Ostermann, E. S. Jang, J.-S. Shin, and T. Chen. Coding of arbitrarily shaped video objects in mpeg-4. In *Proceedings of the International Conference on Image Processing*, pages 496–499, 1997.

[7] W. B. Pennebaker, J. L. Mitchell, G. G. Langdon, and R. Arps. An overview of the basic principles of the q-coder adaptive binary arithmetic coder. *IBM Journal of Research and Development*, 32(6):717–726, 1988.

[8] J. Rissanen and G. G. Langdon. Arithmetic coding. *IBM Journal of Research and Development*, 23(2):149–162, 1979.

[9] S. Rusinkiewicz and M. Levoy. QSplat: A multiresolution point rendering system for large meshes. In *SIGGRAPH 2000 Conference Proceedings*, ACM Siggraph Annual Conference Series, pages 343–352, 2000.

[10] S. Rusinkiewicz and M. Levoy. Streaming QSplat: A viewer for networked visualization of large, dense models. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, pages 63–68. ACM, 2001.

[11] A. Said and W. A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6:243–250, June 1996.

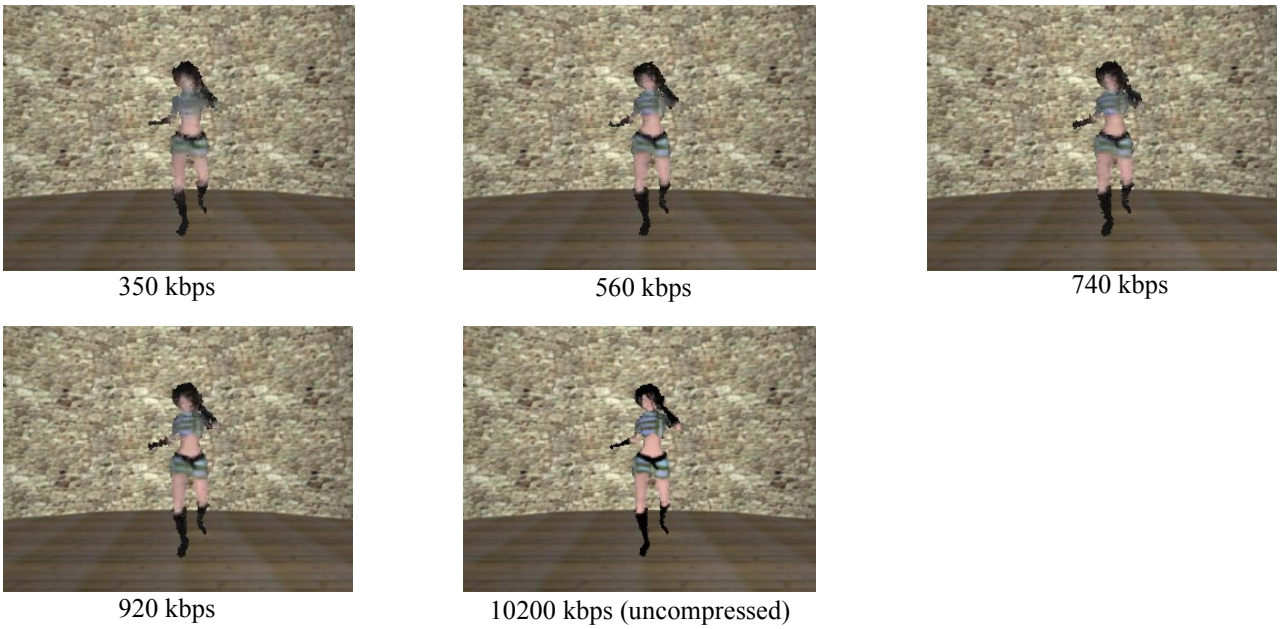
[12] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41:3445–3462, December 1993.

[13] S. Vedula, S. Baker, and T. Kanade. Spatio-temporal view interpolation. In *Proceedings of the 13th ACM Eurographics Workshop on Rendering*, June 2002.

[14] S. Wuermlin, E. Lamboray, and M. Gross. 3D video fragments: Dynamic point samples for real-time free-viewpoint video. *Computers & Graphics, Special Issue on Coding, Compression and Streaming Techniques for 3D and Multimedia Data*, 28(1), 2004

(a) (b) (c) (d) (e) (f)

Figure 10: Average coding illustrated with a texture example: a) Key frame of 12kB. b) Masked key frame (PSNR=29.3dB). c) Key frame plus 100B of delta frame (PSNR=31.4dB). d) Key frame plus 300B of delta frame (PSNR=33.7dB). e) Key frame plus 4.3kB of delta frame (PSNR=42.8dB) f) Original frame.



350 kbps 560 kbps 740 kbps

920 kbps 10200 kbps (uncompressed)

Figure 11: Example images at various bit rates from an MPEG test sequence. The virtual viewpoint lies in between three reconstruction cameras.