# DYNAMIC POINT SAMPLES FOR FREE-VIEWPOINT VIDEO

Stephan Würmlin Edouard Lamboray Michael Waschbüsch Markus Gross

Computer Graphics Laboratory, ETH Zurich

{wuermlin, lamboray, waschbuesch, grossm}@inf.ethz.ch


**Figure 1:** Examples of dynamic point samples for free-viewpoint video. a) real-time free-viewpoint video in the blue-c, b) 3D video recorder with object-space compression, c) 3D video recorder with image-space free-viewpoint video. These results were recorded in various acquisition prototype setups.

## ABSTRACT

Free-viewpoint video (FVV) uses multiple video streams to re-render a time-varying scene from arbitrary viewpoints. FVV enables free navigation with respect to time and space in streams of visual data and allows for virtual replays and for freeze-and-rotate effects, for instance. Moreover, FVV technology can improve communication between remote participants in high-end telepresence applications. In combination with spatial-immersive projection environments, 3D video conferencing allows for life-size, three-dimensional representations of the users instead of small, flat video images. In this paper we propose the application of dynamic point samples as primitives for FVV by generalizing 2D video pixels towards 3D irregular point samples. The different attributes of the point samples define appearance and geometry of surfaces in a unified manner. Furthermore, by storing the reference to a pixel in an input video camera efficient coding and compression schemes can be employed for FVV. We show two different systems for free-viewpoint video using this primitive, namely a real-time FVV system employed in a high-end telecollaboration system and a FVV recording system using two different representations and coding schemes. We evaluate performance and quality of the presented systems and algorithms using the blue-c system with its portals. Both presented 3D video systems are integrated into the telepresence software system of the blue-c, ready to use and demonstrable.

## 1 INTRODUCTION

Video conferencing has come quite a long way from the first commercial systems presented in the early sixties showing blurry images in large television-like boxes with small screens. By now, video conferencing advanced to easy-to-use desktop video conferencing systems with decent image quality. It can facilitate and enhance the collaboration in working groups with team members at remote locations. However, video conferencing today is like watching remote participants on television. Audio communication is augmented with video enabling visual cues and non-verbal gestures. But a real sense of presence is often not perceived. Moreover, it is impossible to make direct eye contact because the camera and the display cannot be at the same place.

Yet, conventional two-dimensional video is a mature technology in both professional environments and home entertainment. A multitude of analog and digital video formats is available today, tailored to the demands and requirements of different user groups. Efficient coding schemes have been developed for various target bit rates, ranging from less than 100 kilobits per second for video conferencing applications to several megabit per second for broadcast quality TV. All of these technologies, however, share the same shortcoming as they are only capable of capturing temporal changes of scenes and objects. Spatial variations, i.e. altering the viewpoint of the user, are not possible at playback. Spatio-temporal effects, e.g. freeze-and-rotate, have been demonstrated in numerous recent feature films, like *The Matrix*. However, these effects can only be realized by employing a large number of still cameras, and involving a considerable amount of manual editing. Typically, input data is processed off-line in various ways to create a plenitude of stunning effects. A *three-dimensional* video system can capture and process this data without manual intervention, in a shorter time and more cost-efficiently. 3D video acquires dynamics and motion of a scene during recording, while providing the user with the possibility to change the viewpoint during playback. Moreover, in a life-size spatially-immersive projection environment, 3D video technology allows to overcome the limitations of small and flat video images. 3D video conferencing features significantly augment the feeling of presence of the remote participants in a collaborative setting. This novel technology bears potential for previously unthinkable user experiences and interactions and thus motivates research towards representations and systems that enable capturing and processing of dynamic scenes and subsequent rendering from arbitrary viewpoints.

## 2 BACKGROUND AND RELATED WORK

A key feature of 3D video is interactivity in the sense that a user should have the possibility to chose an arbitrary viewpoint within a visual real-world scene. However, no formal definition of 3D video is available to date but one can define it as geometrically calibrated and temporally synchronized video data. The broad field of 3D video can be categorized according to spatial camera configurations and application domains. Figure 2 shows the different types of camera configurations for 3D video.

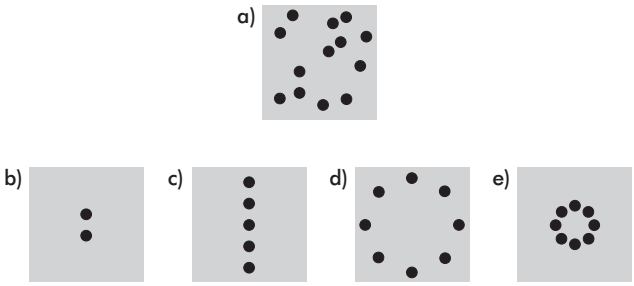


Figure 2: Spatial camera configurations for 3D video. a) arbitrary view, b) stereo view, c) parallel view, d) convergent view, e) divergent view.

While an arbitrary view configuration (Figure 2 a) would permit all application domains, physical and algorithmic constraints typically lead to a reduced complexity of spatial camera configurations. They are typically tailored to specific application domains for 3D video and can be summarized as follows:

3D-television [5, 13] marks a first line of recent research, aiming at view-independent video for dynamic scenes but in a very limited viewing range only. That is, users might experience changes in parallax but no fly-around effects are possible. It can provide stereoscopic display, one view for each eye, to produce a 3D impression for the viewer. 3D-TV is typically acquired using either stereo video captured with two cameras (Figure 2 b), or by a number of densely arranged cameras in parallel view (Figure 2 c). Dense means that the baseline between two cameras does not exceed 50 cm. Such configurations can also be used for spatio-temporal video effects with limited spatial scalability [22].

The concept of free-viewpoint video [2, 20], on the other hand, allows for truly free navigation in the spatial range of captured data, i.e. in the range covered by acquisition cameras. The scene is captured by a number of sparsely arranged cameras in a convergent setup (Figure 2 d). Sparse stands for cameras with a baseline that is in the range of 1 or 2 meters at an angle of 30 degrees. Additional information about the scene geometry, e.g. disparity data, enables interactive and free navigation through the scene.

Omni-directional video [17] is an extension of the conventional planar 2D video image plane towards other non-planar planes like in the static but well-known QuicktimeVR [3] application. User interactions are limited to zoom and rotation around a pre-defined viewpoint. Video is captured at a certain viewpoint into every direction (Figure 2 e).

## 3 DYNAMIC POINT SAMPLES

In recent years, points have experienced a renaissance as a graphics primitive. While there are various methods for fast and high-quality rendering of point sampled geometry at our disposal, e.g. [24, 16, 1], to date none of them can efficiently cope with dynamically changing objects or scenes. But, the input of almost all 3D video systems is generated by range sensing techniques, e.g. laser range scanners, vision-based methods, etc. These acquisition methods and devices inherently generate as raw data non-uniform point samples. The point samples constitute the discrete building blocks of 3D object geometry and appearance—like pixels are the digital elements for images. However, no connectivity information is available and thus, if triangle meshes or higher-order surfaces representations need to be generated and rendered, connectivity has to be reconstructed for the definition of the surface, which is, however, a difficult and time consuming task.

Moreover, a general approach that is able to handle dynamic objects must allow for topology changes—an operation that is costly and hard to perform in real-time on a 3D mesh. But in free-viewpoint video, topology changes in the geometric representation occur frequently due to its time-varying nature. With dynamic point samples, topology changes are implicitly performed while the primitives' attributes are updated. Furthermore, with point samples we can avoid the use of complicated and restricting models, e.g. avatars or articulated human-body models. Model-based approaches [2] need an adaptation of the initial model to the observed object and do not allow to represent objects which cannot be described by the model in the first place.

A point-sampled representation for free-viewpoint video features many advantages as compared to other descriptions. Firstly, it may be understood as a unified representation—quite contrary to approaches based on mesh and texture information that require handling of heterogeneous types of data, i.e. geometry and images. Dynamic point samples can be seen as a natural generalization of 2D video pixels towards 3D irregular point samples. Since the representation incorporates geometrical scene knowledge in terms of point sample attributes we have to deal with less acquisition cameras for even broader viewing ranges as compared to purely image-based approaches in the spirit of Light Fields [11] or Lumigraphs [6].

We propose the application of dynamic point samples as primitives for FVV. The term point sample denotes a sample of a surface in three-dimensional space. It includes information about the underlying surface geometry, such as its position, its normal, its orientation, and the surface appearance, i.e. its reflectance properties. Employed in free-viewpoint systems each point sample stores the reference to a pixel in an input video camera. This enables the use of efficient coding and compression schemes for free-viewpoint video by exploiting the correspondence between pixels and point samples. Figure 3 depicts the relationship between pixels in input images and 3D dynamic point samples.

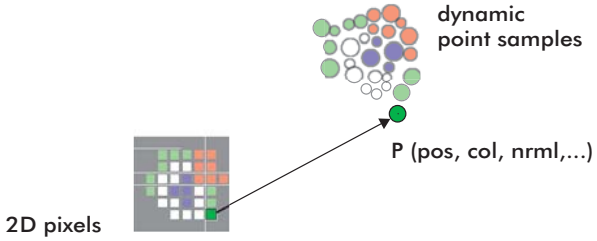


Figure 3: Relationship between 2D pixels and 3D dynamic point samples.

Thus, a dynamic point sample can be seen as an extension to a traditional, static point sample or *surfel* [14, 23]. Dynamic point samples are typically non-uniformly distributed in space, do not have a spatial extent, and do not store connectivity information. A dynamic point sample for FVV additionally stores the correspondence or reference to the camera and to the pixel in this camera

image it was generated from. Thus, a dynamic point sample is necessary generated from an input device, e.g. a digital video camera. Hence, as opposed to mesh based representations, dynamic point samples provide a one-to-one mapping between points and associated color and normal attributes avoiding interpolation and alignment artifacts.

## 4 REAL-TIME FREE-VIEWPOINT VIDEO

We propose a system for real-time free-viewpoint video based on dynamic point samples. A differential update stream inserts, deletes or updates point samples on-the-fly in real-time. It exploits the spatio-temporal coherence of individual 2D video streams by inter-frame prediction of input changes in image space. Our prediction does not require expensive calculations like texture motion fields or 3D scene flows [18]. While being conceptually lean and simple the presented approach effectively cuts down the number of expensive 3D shape computations. By using a feedback loop which confines the number of active cameras, we dynamically control the acquisition process and scale smoothly from view-dependence to view-independence. Moreover, virtual viewpoint- and resolution-driven sampling allows smooth transitions between a subset of the reference cameras and adapts to bandwidth or processing bottlenecks. The method features efficient rendering from arbitrary spatio-temporal positions and supports multiple viewers. Our real-time free-viewpoint video pipeline is designed and optimized for real-time applications and, hence, at multiple stages we trade-off between performance and quality. Figure 4 depicts a conceptual overview of the real-time processing pipeline.

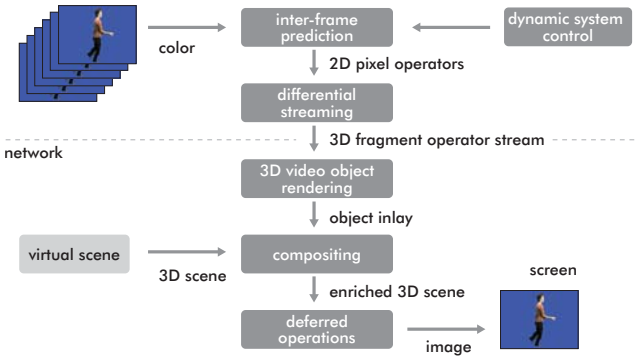


Figure 4: Conceptual components of the real-time free-viewpoint video processing pipeline.

### 4.1 Differential coding

The correspondence between the point samples and the pixel in the input video camera allows to detect changes in the input image and to propagate them to the point samples. Consequently, a dynamic point sample can be generated, updated and deleted based on the changes in the camera image. Inspired by Reeve's work on dynamic particle modeling [15] we describe the dynamic behavior of point samples with three basic operators:

- NEW generates new point samples after they have become visible in one of the input cameras.
- KILL removes point samples from the representation once they vanish from the view of the input camera.
- UPDATE corrects appearance and geometry attributes of point samples that are already generated, but whose attributes have changed with respect to prior frames of the input camera.

The time sequence of these operators creates a differential operator stream that updates a 3D video data structure on a remote side. An INSERT operator results from the reprojection of a pixel with color attributes from image space into three-dimensional object space. Any real-time 3D reconstruction method which extracts depth and normals from images can be employed for this purpose, e.g. the image-based visual hull algorithm [12]. DELETE operators perform a lookup of the reference point sample and eliminate it. UPDATE operators are generated by all pixels which have been inserted in previous frames and which are still foreground pixels. They can be divided into three categories: The detection of *color changes* is performed during inter-frame prediction and leads to an UPDATECOL operator. UPDATEPOS operators take care of *geometry changes* and are analyzed on spatially coherent clusters of pixels in image space. If the differences to the previous depths exceed a threshold, we recompute 3D information for entire blocks of points. Thus, our scheme proposes an efficient solution to the problem of uncorrelated texture and depth motion fields. Note that position and color updates can be combined to an UPDATEPOSCOL operator. All other candidate pixels for updates remain *unchanged* and no further processing is necessary.

We employ a simple image space inter-frame prediction mechanism which derives the operators from the original video images by only using two functions for pixel classification, namely foreground/background segmentation and color differencing.

### 4.2 Implementation

Based on the differential coding scheme a system is implemented for real-time free-viewpoint video acquisition, processing and rendering. A detailed structural overview of the system architecture of the real-time system is depicted in Figure 5.

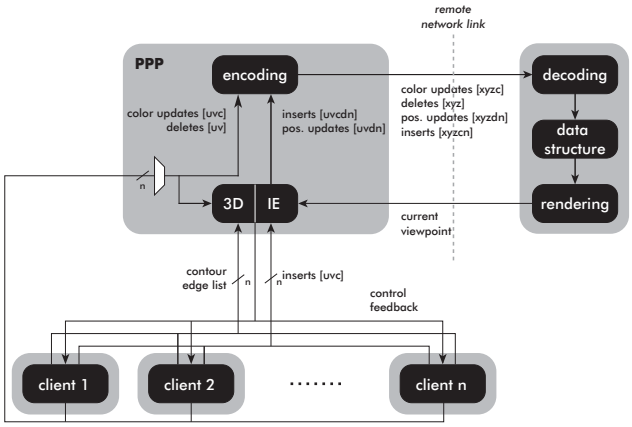


Figure 5: Detailed structure of the real-time free-viewpoint video system architecture.

The central component is the PPP—for Progressive Point Processor—which computes 3D point samples and performs encoding of the 3D video stream. A distributed system architecture is implemented with multi-threading capabilities to employ multiple processors. Figure 5 also shows data flows in a real-time free-viewpoint video system with differential coding using the operator framework. Gray boxes indicate node boundaries. The left side of the dotted line is called acquisition side which typically runs on an acquisition cluster, and the part to the right is the rendering side, which renders the free-viewpoint video object and

embeds it into a virtual scene. Black boxes are explained in more detail below. After acquiring the images the basic entities that are dealt with are point sample operations.

- **Clients.** The clients acquire the images, perform background subtraction, and extract the contours, which are send directly to the PPP. They also perform the image differencing, resulting in three output streams, containing INSERTs, UPDATEs, or DELETEs, respectively. Positional updates are completely processed on the 3D/IE unit. Since the calculation of 3D coordinates is done in a later step, these operations are still in image coordinates. A detailed structure can be found in [19].
- **Insert engine (IE).** The inserts are sent to the so–called insert engine which schedules blocks of INSERTs according to the current viewpoint. It contains the 3D unit to compute the depth values of the corresponding points and forwards them to the PPP. Furthermore, this unit takes care of the positional updates.
- **3D.** The 3D unit implements a 3D reconstruction method, currently a variant of the IBVH algorithm [12]. For each pixel we calculate a depth value which is then projected to a point sample in 3D space with associated normal during encoding.
- **PPP encoding.** The UPDATECOLs and DELETEs are sent directly to the PPP encoding unit. It converts all points from image coordinates with eventual depth information to real 3D points, and attaches referencing information needed for updates and deletions. Furthermore, it stores all active point samples in a caching structure called the *point-cache*. This cache is needed for faster processing of UPDATE and DELETE operators. Eventually, all operators are encoded, compressed and sent to the remote side [9].

The result of the PPP is a stream of operations, i.e. point sample INSERTs, UPDATECOL and UPDATEPOS and DELETEs in 3D–space, which is sent to the rendering side over the main transmission link.

- **Rendering.** The rendering unit is composed of decoding the stream of operations, and the rendering itself. Rendering can be done asynchronously to the acquisition frame rate. Currently, a variant of EWA surface splatting [24] is used.

### 4.3 Results

We evaluated different configurations of the differential coding framework. To this end, we simulated real-time processing of a 3D video sequence of 100 frames using the recording system (Section 5) performing the differential scheme on all cameras. Camera control was disabled and, hence, no progressive sampling is performed. This leads to significantly higher bit rates than in a real-time session [20] since all pixels are processed in each frame.

We evaluated two parameters of the update operators. The *positional update threshold* denotes the maximum absolute depth difference between clusters in subsequent frames. Since we have a metric calibration at our disposal we can define this threshold in meters. Hence, a threshold of 0.04 denotes a maximum absolute difference in the depth value of ±4 cm. The *color update threshold* denotes the maximum absolute difference between the RGB color channels of a pixel in subsequent frames. Hence, a threshold of 16:8:16 denotes that the maximum color intensity difference for the R- and B-channels is ±16 , and ±8 for the G-channel.

**Positional Updates (UPDATEPOS).** Figure 6 shows rendered images from these sessions with varying positional update threshold after 100 frames of differential coding. The color update threshold is held constant in the configuration 16:8:16. As can be seen clearly in the difference images in Figure 7, differential free-viewpoint video leads to block artifacts if the threshold is set too

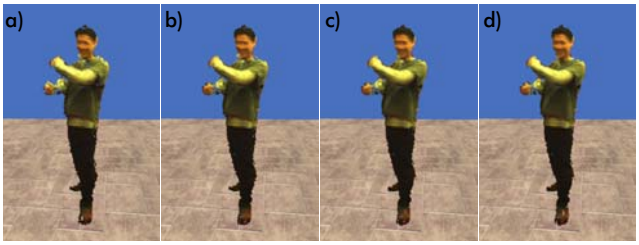


**Figure 6:** Rendered images in differential FVV evaluating positional updates. a) threshold 0.04, b) threshold 0.02, c) threshold 0.01, d) threshold 0.005.

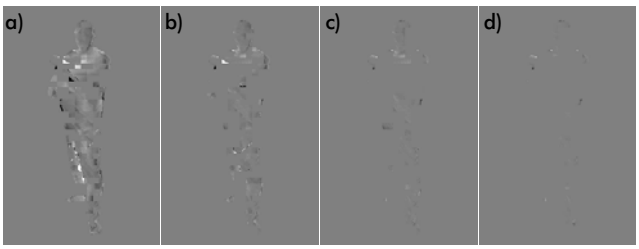


**Figure 7:** Difference images for positional updates evaluation comparing correct and differentially updated depth images, from a camera contributing to the view in Figure 6, magnified by a factor of 5. a) threshold 0.04, b) threshold 0.02, c) threshold 0.01, d) threshold 0.005.

different positional updates configurations. We compared the correct and differentially updated depth images and averaged the result over all frames of the sequence. We see that the quality of the differential depth images is very high for all evaluated thresholds. But for thresholds over 0.02 on average more than half of the fragments get updated in each frame which leads to a high bit rate in the real-time stream. Nevertheless, even with a threshold of 0.04 decent image quality can be achieved.

**Table 1:** Results from positional updates in differential FVV, PSNR values and bit rates for different positional update thresholds comparing correct and differentially updated depth images at 15 fps, from a camera contributing to the view in Figure 6.

| UPDATEPOS threshold | PSNR [dB] | total bit rate [Mbps] | UPDATEPOS only [Mbps] |
|---|---|---|---|
| 0.04 | 35.4 | 4.81 | 2.10 |
| 0.02 | 37.8 | 6.18 | 3.46 |
| 0.01 | 40.6 | 8.06 | 5.34 |
| 0.005 | 44.1 | 10.10 | 7.38 |

**Color Updates (UPDATECOL).** Figure 8 shows rendered images from the simulated real-time sessions with varying color update threshold after 100 frames of differential coding. The positional update threshold is held constant at 0.02. Figure 9 shows difference images for a camera contributing to the rendered result in Figure 8. We see that the error is smoothly distributed over the whole image.

As can be seen in Table 2 we can cut down the bit rate significantly by setting a high color update threshold. Although the PSNR values for high thresholds forebode low color quality the rendered results still look satisfactory. For a threshold of 8:16:8 on average 2'700 fragments get color updated out of a total number of 12'200 per frame. By increasing the threshold to 48:24:48 we can lower the number of color updates to an average of 1'200

fragments per frame. We refer the reader to [19, 20] for an in-depth technical discussion of the real-time free-viewpoint video system.

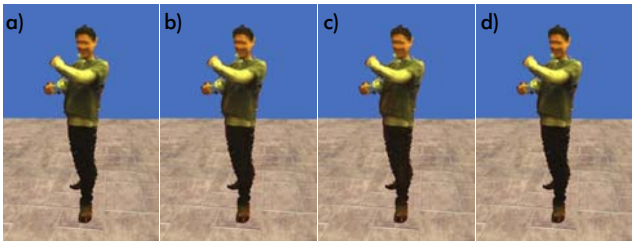


Figure 8: Rendered images in differential FVV evaluating color updates. a) RGB thresholds 16:8:16, b) RGB thresholds 32:16:32, c) RGB thresholds 48:24:48, d) RGB thresholds 24:8:24.

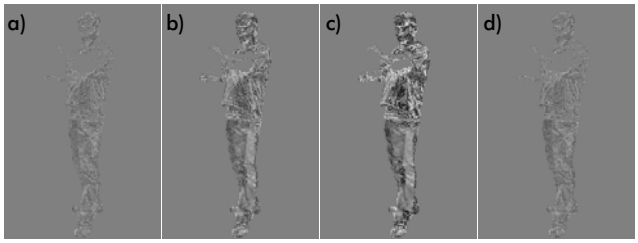


Figure 9: Difference images for color updates evaluation comparing correct and differentially updated color images, from a camera contributing to the view in Figure 8, magnified by a factor of 5. a) RGB thresholds 16:8:16, b) RGB thresholds 32:16:32, c) RGB thresholds 48:24:48, d) RGB thresholds 24:8:24.

Table 2: Results from color updates in differential FVV, PSNR values and bit rates for different color update thresholds comparing correct and differentially updated color images, from a camera contributing to the view in Figure 8.

| UpdateCol thresholds | PSNR [dB] | total bit rate [Mbps] | UpdateCol only [Mbps] |
|---|---|---|---|
| 16:8:16 | 38.3 | 6.17 | 1.95 |
| 32:16:32 | 32.8 | 5.44 | 1.21 |
| 48:24:48 | 29.7 | 5.10 | 0.87 |
| 24:8:24 | 37.7 | 6.09 | 1.86 |

# 5 FREE-VIEWPOINT VIDEO RECORDING

A 3D video recorder is a system capable of recording, processing, and playing three-dimensional video from multiple points of view. First, 2D video streams are recorded from several synchronized digital video cameras and pre-processed images are stored to disk. An off-line processing stage converts these images into time-varying three-dimensional data and stores this 3D video to disk. Dynamic point samples are a suitable primitive for such a system. We propose two encoding techniques, one building a hierarchical point-based data structure and one that encodes the data in image-space. The latter guides the way to the usage of conventional video coding techniques for FVV. A 3D video player decodes and renders 3D videos from hard-disk in real-time, providing interaction features known from common video cassette recorders, like variable-speed forward and reverse, and slow motion. 3D video playback can be enhanced with novel 3D video effects such as freeze-and-rotate and arbitrary scaling.

We refer the reader to [19] for an in-depth technical discussion of the free-viewpoint video recording system and its coding techniques.

## 5.1 Object-space Compression

For object-space compression of free-viewpoint video data we use the PR*k*-tree [21] as data representation for a dynamic point cloud. These trees, which represent the reconstructed 3D video frames, can be stored using a space efficient and progressive representation. In order to achieve a progressive encoding, we traverse a tree in a breadth-first manner. Hence, we first encode the upper-level nodes, which represent an averaged representation of the corresponding subtree. As suggested in [4], a succinct storage of the 3D representation can be achieved by considering separately the different data types it contains. We distinguish between the connectivity of the PR*k*-tree, which needs to be encoded without loss, and the position of the points, color and normal information, where the number of allocated bits can be traded against visual quality and lossy encoding is acceptable.

We follow the algorithm from [7] to encode the connectivity of the tree. Position information is encoded using a two step process consisting of an approximation and a refinement step by encoding the error using a Laplace quantizer. The color data is encoded in YUV color space with a quantization scheme using twice as much bits for the Y component than for the U and V components, respectively. The normal vectors are encoded using quantized spherical coordinates. We normalize the vectors before encoding and then allocate a certain amount of bits for each of the two angles. Table 3 summarizes the storage requirements for the different data types per node and compares them to the initial data size. For lossless encoding of the connectivity of the PR*k*-tree, we use a scheme coming close to the information theoretic bound. The indicated values for the remaining data types are those which provided us with visually appealing results.

Table 3: Memory requirements for one PR27 tree.

| name | data type | raw [bits] | compressed [bits] |
|---|---|---|---|
| position | float[3] | $3 \cdot 32$ | $3+3+3$ |
| color | char[3] | $3 \cdot 8$ | $6+3+3$ |
| normal | float[3] | $3 \cdot 32$ | 8 |
| noOfChildren | unsigned char | 8 | $2+1+\lceil \lg 27 \rceil$ |
| children | *PRkNode | $27 \cdot 32$ | |
| total | | 1088 | 37 |

Consecutive frames in a 3D video sequence contain a lot of redundant information, i.e. regions of the object remaining almost static, or, changes which can be efficiently encoded using temporal prediction and motion compensation algorithms. However, the efficient computation of 3D scene flows is non-trivial. Detailed discussion of object-space FVV compression can be found in [8].

## 5.2 Image-space Compression

Alternatively, we can organize and compress the free-viewpoint video data in image-space. To this end, we use an image-space representation and data format adopted by MPEG as an extension of the MPEG-4 AFX standard. In the standard, this representation is termed depth image-based representations (DIBR) Version 2 used for high-quality rendering of point-sampled objects. Combined with suitable coding methods it is capable of streaming and displaying sparse multi-view video data from arbitrary view-

points. It is based on the fundamental concept of storing all information describing a scenes visual appearance in multi-channel video images. Each pixel's channels define different attributes of discrete point samples of observed surfaces. These include color, position, and optional data needed for high-quality rendering. Multi-channel multi-view video compression can be implemented with standard MPEG video coding tools and ready-available video coding methods can thus be reused. Consequently, a complete free-viewpoint video system can be built using only MPEG-standardized tools—to our knowledge the first of its kind. Figure 10 illustrates this image-space free-viewpoint video framework.

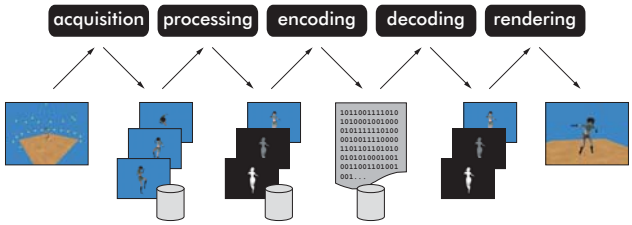


Figure 10: The image-space FVV video recording pipeline.

The image-space FVV framework is discussed in detail in [19] and a novel progressive video coding method for this representation is proposed in [10]. Figure 11 shows rendered image-space FVV results using MPEG-4 as coding method at different bit rates.

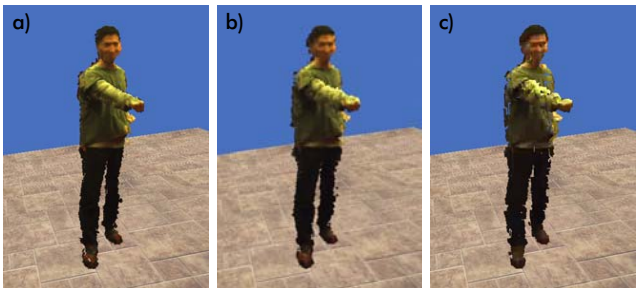


Figure 11: Rendered FVV images with different coding methods. a) uncompressed data, b) MPEG-4 at total bit rate of 512 kbps per camera, c) MPEG-4 at total bit rate of 128 kbps per camera.

## 6 CONCLUSIONS AND FUTURE WORK

In this paper we investigated representations, coding methods and frameworks for free-viewpoint video using sparsely arranged video cameras. We presented a novel primitive which allows to represent real-world objects by dynamic point samples which handle geometry and appearance of three-dimensional surfaces in a unified way. Albeit the presented frameworks and results, free-viewpoint video is still a challenging and emerging visual media and there is plenty of room for future work, e.g. improving the quality of free-viewpoint video, handling dynamic scenes, and interacting with or editing this spatio-temporal multi-view video data.

## ACKNOWLEDGEMENTS

We thank Stefan Hösli, Peter Kaufmann, Nicky Kern, and Christoph Niederberger for implementing parts of the system. Special thanks to all members of the blue-c team for many fruitful discussions, Wojciech Matusik for sharing the IBVH source code with us, Hanspeter Pfister and Aljoscha Smolic for interesting discussions, and Reto Lütolf for many proof-reading sessions. This work is carried out in the context of the blue-c project, funded by ETH grant No. 0-23803-00 as an internal poly-project.

## REFERENCES

[1] M. Botsch and L. Kobbelt. High-quality point-based rendering on modern gpus. In *Proceedings of Pacific Graphics 2003*, pages 335 – 442. IEEE Computer Society Press, 2003.

[2] J. Carranza, C. Theobalt, M. Magnor, and H.-P. Seidel. Free-viewpoint video of human actors. In *Proceedings of SIGGRAPH 2003*, pages 569–577. ACM Press / ACM SIGGRAPH, 2003.

[3] S. E. Chen. Quicktime VR - an image-based approach to virtual environment navigation. In *Proceedings of SIGGRAPH 95*, pages 29–38. ACM Press, Addison Wesley, 1995.

[4] M. Deering. Geometry compression. In *Proceedings of SIGGRAPH 95*, pages 13–20. ACM SIGGRAPH, Addison Wesley, 1995.

[5] C. Fehn. Depth Image-Based Rendering (DIBR), compression and transmission for a new approach on 3D-TV. In *Proceedings of Stereoscopic Displays and Applications*, 2004.

[6] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *Proceedings of SIGGRAPH 96*, pages 43–54. ACM SIGGRAPH, Addison Wesley, 1996.

[7] G. Jacobson. Space-efficient static trees and graphs. In *30th Annual Symposium on Foundations of Computer Science*, pages 549–554. IEEE Computer Society Press, 1989.

[8] E. Lamboray. *A Communication Infrastructure for Highly-Immersive Collaborative Virtual Environments*. PhD thesis, ETH Zurich, No. 15618, 2004.

[9] E. Lamboray, S. Wuermlin, and M. Gross. Real-time streaming of point-based 3D video. In *Proceedings of the IEEE Virtual Reality 2004 conference*, pages 91–98. IEEE Computer Society Press, March 2004.

[10] E. Lamboray, S. Würmlin, M. Waschbüsch, M. Gross, and H. Pfister. Unconstrained free-viewpoint video coding. In *Proceedings of ICIP 2004*, 2004.

[11] M. Levoy and P. Hanrahan. Light field rendering. In *Proceedings of SIGGRAPH 96*, pages 31–42. ACM SIGGRAPH, Addison Wesley, 1996.

[12] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based visual hulls. In *Proceedings of SIGGRAPH 2000*, pages 369–374. ACM Press / ACM SIGGRAPH, 2000.

[13] W. Matusik and H. Pfister. 3D TV: A scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. In *Proceedings of SIGGRAPH 2004*. ACM Press / ACM SIGGRAPH, 2004.

[14] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of SIGGRAPH 2000*, pages 335–342. ACM SIGGRAPH, Addison Wesley, 2000.

[15] W. T. Reeves. Particle systems - a technique for modeling a class of fuzzy objects. In *Proceedings of SIGGRAPH 83*, pages 91–108. ACM Transactions on Graphics, 1983.

[16] L. Ren, H. Pfister, and M. Zwicker. Object space EWA surface splatting: A hardware accelerated approach to high quality point rendering. In *Proceedings of Eurographics 2002*, COMPUTER GRAPHICS Forum, Conference Issue, pages 461–470, 2002.

[17] A. Smolic and D. McCutchen. 3DAV exploration of video-based rendering technology in MPEG. *IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Immersive Communications*, 14(9):348–356, 2004.

[18] S. Vedula, S. Baker, and T. Kanade. Spatio-temporal view interpolation. In *Proceedings of the 13th Eurographics Workshop on Rendering*, pages 65–76, 2002.

[19] S. Würmlin. *Dynamic Point Samples as Primitives for Free-viewpoint Video*. PhD thesis, ETH Zurich, No. 15643, 2004.

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

[21] S. Würmlin, E. Lamboray, O. G. Staadt, and M. H. Gross. 3D video recorder. In *Proceedings of Pacific Graphics 2002*, pages 325–334. IEEE Computer Society Press, 2002.

[22] L. Zitnick, S. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. In *Proceedings of SIGGRAPH 2004*. ACM Press / ACM SIGGRAPH, 2004.

[23] M. Zwicker. *Continuous Reconstruction, Rendering, and Editing of Point-Sampled Surfaces*. PhD thesis, ETH Zurich, No. 15135, 2003.

[24] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Surface splatting. In *Proceedings of SIGGRAPH 2001*, pages 371–378. ACM Press / ACM SIGGRAPH, New York, 2001.