Design, Implementation, and Evaluation of a Point Cloud Codec for Tele-Immersive Video

Rufael Mekuria, *Student Member, IEEE*, Kees Blom, and Pablo Cesar, *Member, IEEE*

***Abstract*— We present a generic and real-time time-varying point cloud codec for 3D immersive video. This codec is suitable for mixed reality applications in which 3D point clouds are acquired at a fast rate. In this codec, intra frames are coded progressively in an octree subdivision. To further exploit inter-frame dependencies, we present an inter-prediction algorithm that partitions the octree voxel space in $N \times N \times N$ macroblocks ($N = 8, 16, 32$). The algorithm codes points in these blocks in the predictive frame as a rigid transform applied to the points in the intra-coded frame. The rigid transform is computed using the iterative closest point algorithm and compactly represented in a quaternion quantization scheme. To encode the color attributes, we defined a mapping of color per vertex attributes in the traversed octree to an image grid and use legacy image coding method based on JPEG. As a result, a generic compression framework suitable for real-time 3D tele-immersion is developed. This framework has been optimized to run in real time on commodity hardware for both the encoder and decoder. Objective evaluation shows that a higher rate-distortion performance is achieved compared with available point cloud codecs. A subjective study in a state-of-the-art mixed reality system shows that introduced prediction distortions are negligible compared with the original reconstructed point clouds. In addition, it shows the benefit of reconstructed point cloud video as a representation in the 3D virtual world. The codec is available as open source for integration in immersive and augmented communication applications and serves as a base reference software platform in JTC1/SC29/WG11 (MPEG) for the further development of standardized point-cloud compression solutions.**

***Index Terms*— Data compression, point clouds, teleconferencing, video codecs, virtual reality.**

I. Introduction

With increasing capability of 3D data acquisition devices and computational power, it is becoming easier to reconstruct highly detailed photo-realistic point clouds (i.e., point-sampled data) representing naturalistic content, such as persons or moving objects/scenes [1], [2]. 3D point clouds are a useful representation for 3D video streams in mixed reality systems. Not only do they allow free viewpoint rendering (for example, based on splat rendering), but they can also be compositely rendered in a synthetic 3D scene as they provide full information on 3D geometry coordinates. Therefore, this type of video representation is preferable in mixed reality systems, such as *augmented reality*, in which a natural scene is combined with synthetic (authored, e.g., computer graphics) objects, or, vice versa, in *immersive virtual rooms* in which a synthetic scene is augmented with a live captured natural 3D video stream representing a user.

Traditionally, 3D polygon meshes have often been used to represent 3D object-based visual data. However, point clouds are simpler to acquire than 3D polygon meshes, as no triangulation needs to be computed, and they are more compact as they do not require the topology/connectivity information to be stored. Therefore, 3D point clouds are more suitable for real-time acquisition and communication at a fast rate. However, realistic reconstructed 3D point clouds may contain hundreds of thousands up to millions of points, and compression is critical to achieve efficient and real-time communication in bandwidth-limited networks.

Compression of 3D point clouds has received significant attention in recent years [3]–[6]. Much work has been done to efficiently compress single point clouds *progressively*, such that lower-quality clouds can be obtained from partial bit streams (i.e., a subset of the original stream). To compare different solutions, often the *compression rate* and the *geometric distortion* have been evaluated. Sometimes the *algorithmic complexity* was analyzed, and in addition schemes for *attribute coding* (i.e., colors and normals) have been proposed. While these approaches are a good starting point, in the context of immersive, augmented, and mixed reality communication systems, several other additional factors are of importance.

One important aspect of these systems is *real-time performance* for encoding and decoding. Often, in modern systems, parallel computing that exploits multicore architectures available in current computing infrastructures is utilized. Therefore, *parallelizability* becomes important. Second, as in these systems point cloud sequences are captured at a fast rate, inter-frame redundancy can be exploited to achieve a better compression performance via *inter prediction*, which is usually not considered in existing static point cloud coders.

Furthermore, as tele-immersive codecs are intended for systems with real users, *subjective quality assessment* is needed to assess the performance of the proposed codec in addition to the more common *objective quality assessment*. Further, a codec should be *generic*, in the sense that it should be able to

Manuscript received October 2, 2015; revised December 18, 2015; accepted February 1, 2016. Date of publication March 16, 2016; date of current version April 3, 2017. This work was supported by the European Community's Seventh Framework Programme (FP7/2007-2013) through the REVERIE Project under Grant ICT-2011-7-287723. This paper was recommended by Associate Editor P. Eisert.

R. Mekuria is with Vrije Universiteit Amsterdam, Amsterdam 1081 HV, The Netherlands.

K. Blom is with Centrum Wiskunde & Informatica, Amsterdam 1098 XG, The Netherlands.

P. Cesar is with the Distributed and Interactive Systems Group, Centrum Wiskunde & Informatica, Amsterdam 1098 XG, The Netherlands.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2016.2543039

compress point cloud sequences coming from different setups with different geometric properties (i.e., sampling density, manifoldness of the surface, etc.).

With these requirements in mind, we introduce a codec for time-varying 3D point clouds for augmented and immersive 3D video. The codec is parallelizable and generic and operates in real time on commodity hardware. In addition, it exploits inter-frame redundancies. For evaluation, we propose an objective quality metric that corresponds to common practice in video and mesh coding. In addition to objective evaluation using this metric, subjective evaluation in a realistic mixed reality system is performed in a user study with 20 users.

The codec is available as open source and currently serves as the reference software framework for the development of a point-cloud compression technology standard in JTC1/SC29/WG11 (MPEG).[1] To facilitate benchmarking, the objective quality metrics and file loaders used in this paper have all been included in the package. In addition, the point cloud test data are publicly available.

The rest of this paper is structured as follows. In Section III, we detail the lossy attribute/color coding and progressive decoding scheme. In Section IV, the inter-predictive coding algorithm is detailed. Section V details the experimental results, including subjective test results in a mixed reality system, objective rate distortion (R-D), and real-time performance assessment. In Section II, we provide the overview of the codec, and in this section, we provide the context and related work.

A. Contributions

In this paper, we propose a novel compression framework for progressive coding of time-varying point clouds for 3D immersive and augmented video. The major contributions are as follows.

1) *Generic Compression Framework:* A framework that can be used to compress point clouds with arbitrary topology (i.e., different capturing setups or file formats).
2) *Inter-Predictive Point Cloud Coding:* Correlation between subsequent point clouds in time is exploited to achieve better compression performance.
3) *Efficient Lossy Color Attribute Coding:* The framework includes a method for lossy coding of color attributes, which takes advantage of naturalistic source of the data using existing image coding standards.
4) *Progressive Decoding:* The codec allows a lower-quality point cloud to be reconstructed by a partial bit stream.
5) *Real-Time Implementation:* The codec runs in (near) real time on commodity hardware, benefiting from multicore architectures and a parallel implementation.

B. Augmented and Immersive 3D Video Versus FVV and 3DTV

There exist quite a few technologies for coding 3D video. In this section, we further motivate the additional need for point cloud compression for immersive and augmented 3D video. 3D video often refers to 3D television (3DTV) or free viewpoint video (FVV). 3DTV creates a depth perception, while FVV enables arbitrary viewpoint rendering. Existing video coding standards, such as Advanced Video Coding (AVC) Multi View Video (MVV) [7] and MVV-D [8], can support these functionalities via techniques from (depth) image-based rendering (DIBR). Arbitrary views can be interpolated from decompressed view data using spherical interpolation and original camera parameter information. This enables free viewpoint rendering without having explicit geometry information available. However, in mixed reality systems, explicit object geometry information is needed to facilitate composite rendering and object navigation.

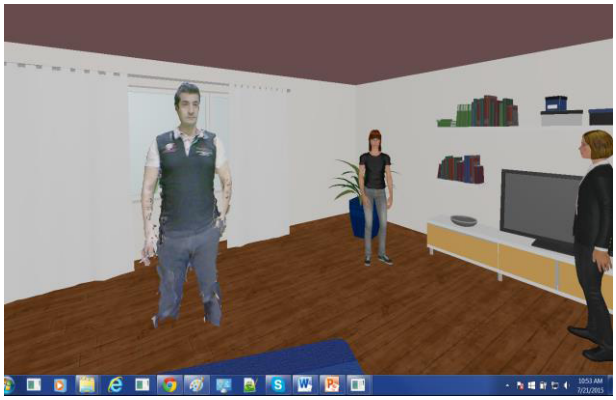

Fig. 1. Screenshot of composite rendering in virtual room based on the Reverie system. The point cloud's naturalistic content is rendered compositely with synthetic content. Navigation and user support enable interaction between naturalistic point cloud and synthetic avatar users.

In such systems, rendering is usually done based on object geometry, such as meshes or 3D point clouds, using generic computer graphics Application Programming Interface (e.g., OpenGL and Direct3D). This is in line with common practice in computer games and virtual worlds. Therefore, to enable true convergence between naturalistic and synthetic contents in immersive and augmented reality, object-based video compression of point clouds and 3D meshes remains a challenge. To illustrate this need further, we show some examples from a practical tele-immersive system that we have been working on in the last four years, the Reverie system [9]. The Reverie system is an immersive communication system that enables online interaction in 3D virtual rooms, represented by either a 3D avatar or 3D object-based video stream based on 3D point clouds or meshes. As can be seen in Fig. 1, each representation can be rendered compositely in a common 3D space. This 3D video stream is a segmentation of the real 3D user. Fig. 2 illustrates the low-cost 3D point cloud acquisition system deployed in this system. It uses multiple calibrated Kinect sensors. Original color plus depth streams Red, Green, Black (RGB) + D are fused and the foreground moving object is segmented as a point cloud. The basic steps of such an algorithm are shown in Fig. 3. The output is a point cloud ready for remote rendering. This raises the need for efficient 3D point cloud video compression and transmission.

[1] http://wg11.sc29.org/svn/repos/MPEG-04/Part16-Animation_Framework_eXtension_(AFX)/trunk/3Dgraphics/

Fig. 2. Low-cost point-cloud capturing setup. Point clouds are reconstructed from multiple 3D input streams of calibrated Microsoft Kinect devices.

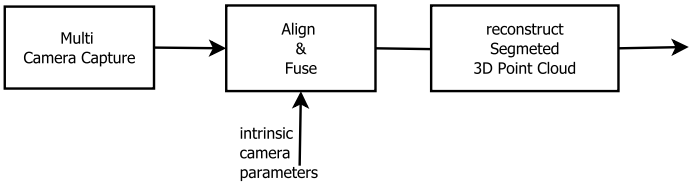

Fig. 3. Schematic of 3D point cloud reconstruction for immersive video.

Alternatively, by running the algorithm in Fig. 3 at a remote site on decompressed RGB + D data, one could use RGB + D video coders directly on sensor data. We have done some experimentation comparing RGB + D coding using MPEG-4 AVC simulcast (QP8–QP48, zero latency, x264 encoder) and point cloud compression (8–10 b per direction component) using the point clouds in [2] reconstructed from 5 (RGB + D) streams (data available in [10], captured with Microsoft Kinect). Byte sizes of 132–800 kB per frame were achieved using RGB-D coding, while bitrates of 40–265 kB were obtained using the octree-based point cloud compression. In addition, the distortion introduced in the point cloud, which could be unpredictable for low-bitrate RGB-D coding, is bound by the octree resolution in an octree-based point cloud codec. Last, the experiments showed lower encoding/decoding latencies when using point cloud compression. As in telepresence and immersive reality, low latency, low bitrate, and bound distortion are critical; we develop compression for time-varying point clouds.

C. Related Work

1) Point Cloud Compression: There has been some work on point cloud compression in the past, but most works aimed at the compression of only static point clouds, instead of time-varying point clouds. Such a codec was introduced in [11] based on octree composition. This codec includes bit reordering to reduce the entropy of the occupancy codes that represent octree subdivisions. This method also includes color coding based on frequency of occurrence (colorization) and normal coding based on efficient spherical quantization. A similar work in [12] used surface approximations to predict occupancy codes and an octree structure to encode color information. The work in [3] introduced a real-time octree-based codec that can also exploit temporal redundancies by XOR operations on the octree byte stream. This method can operate in real time, as the XOR prediction is extremely simple and fast. A disadvantage of this approach is that by using XOR, only geometry and not colors can be predicted, and that the effectiveness is significant only for scenes with limited movement (which is not the case with envisioned application with moving humans). Last, Thanou *et al.* [5] introduced a time-varying point cloud codec that can predict graph-encoded octree structures between adjacent frames. The method uses spectral wavelet-based features to achieve this and an encoding of differences to achieve a lossless encoding. This method also includes the color coding method from [6], which defines small subgraphs based on the octree of the point cloud. These subgraphs are then used to efficiently code the colors by composing them on the eigenvectors of the graph Laplacian.

2) Mesh Compression: 3D objects are often coded as 3D meshes, for which a significant number of compression methods have been developed. Mesh codecs can be categorized as progressive, i.e., allowing a lower resolution rendering from partial bit streams, and single rate (only decoding at full resolution is available) [13]. For networked transmission, progressive methods have generally been preferred, but for 3D immersive video, single rate can also be useful, as they introduce less encoder computation and bitrate overhead [14]. [15]–[17] have aimed at compression of object-based immersive 3D video using single-rate coding. While these methods are promising, it seems that methods based on 3D point clouds can result in coding with even less overhead and more flexible progressive rendering capabilities, as the format is simpler to acquire and process. Last, there have been methods defined in the international standards for mesh compression [18], [19] which are greatly beneficial for interoperability between devices and services. These methods have been mostly designed for remote rendering and have low decoder complexity and a slightly higher encoder complexity; in 3D immersive and augmented 3D video coding, having both low encoder and decoder complexity is important (analogous to video coding in video conferencing systems compared with video on demand).

3) Multiview Plus Depth Compression: Multiview plus depth representation was considered for storing video and depth maps from multiple cameras [7], [8]. Arbitrary viewpoints are then rendered by interpolation between different camera views using techniques from DIBR to enable free viewpoint. While these formats can be used to represent the visual 3D scene, they do not explicitly store 3D object geometries, which is useful for composite rendering in immersive communications and augmented reality. Therefore, these formats are not directly applicable to immersive and augmented 3D object-based video.

4) Compression of Immersive 3D Video: The specific design requirements posed by 3D video for immersive communications have been addressed before in [20] and [21]. The first work introduces a compression scheme of polygon-based 3D video (in this case, a segmentation and meshing

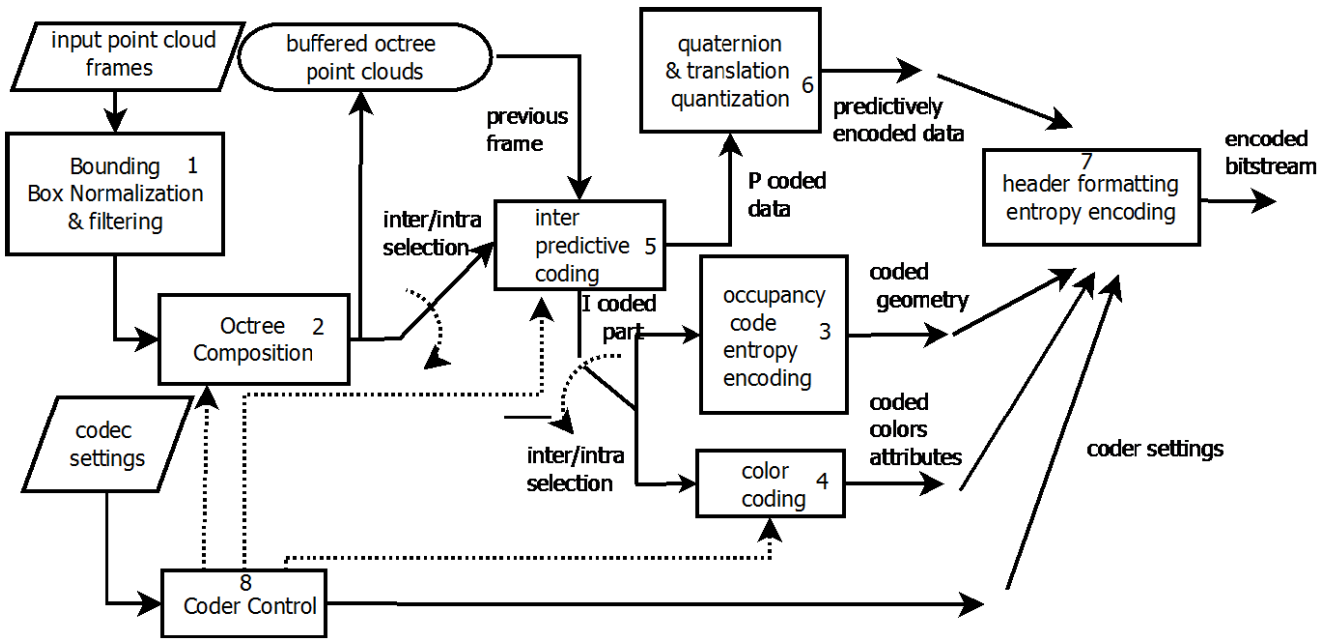

Fig. 4. Schematic of a time-varying point-cloud compression codec.

in a color + depth video) and a purely perception-based compression mechanism combined with entropy encoding. In this paper, the level of detail (LoD) (polygon size) is adapted to match user/application needs. These needs were derived offline in subjective studies with prerecorded stimuli. In [21], the MPEG-4 video codec was used in a smart way together with camera parameters. Both methods have been developed in a context that combines the capturing, compression, and networking components toward the specific tele-immersive configuration. This lack of generality makes it harder to assess the relevance and performance of video compression compared with other methods. In this paper, we aim to provide a generic 3D point cloud codec that can be compared with other codecs for point cloud compression and applied to any capturing setup that produces 3D point cloud data.

II. Overview of Point Cloud Coding Scheme

We first outline the requirements for point cloud compression, after which we detail the point cloud coding schematic (Fig. 4).

A. Requirements and Use Cases

3D video based on point clouds is relevant for augmented and mixed reality applications, as shown in Fig. 1. In addition, it has various applications (e.g., to store data used in geographic information systems and 3D printing applications). We focus on time-varying point cloud compression for 3D immersive and augmented video and follow the requirements for point cloud compression as defined in the MPEG-4 media standard [22].

1) *Partial Bit Stream Decoding:* It is possible to decode a coarse point cloud and refine it.
2) *Lossless Compression:* The reconstructed data are mathematically identical to the original data.
3) *Lossy Compression:* Compression with parameter control of the bitrate.
4) *Time Variations and Animations:* Temporal variations, i.e., coding of point cloud sequences, should be supported.
5) *Low Encoder and Decoder Complexity:* This is not a strict requirement but desirable for building real-time systems.

B. Schematic Overview

The architectural design of the proposed 3D video based on point clouds combines features from common 3D point cloud (octree-based) codecs [11], [12] and common hybrid video codecs, such as MPEG-4 p.10 AVC and HEVC (which include block-based motion compensation). Based on the numbering in the diagram in Fig. 4, we detail the most important components in the codec, which also correspond to our main contributions.

1) Bounding Box Alignment and Filter (1): A specific algorithm for bounding box computation and alignment has been developed. This algorithm is applied to the point cloud before the octree composition (2) is performed. This algorithm aligns subsequent frames by computing a common expanded bounding box. This allows common axis and range representation between frames, which facilitates the inter-predictive coding and a consistent assignment of the octree bits. In addition, it includes an outlier filter, as our experiments have shown that outlier points can reduce the effectiveness of both the bounding box alignment and inter-predictive coding, and should be filtered out from the input clouds (see Section III-A for details).

2) Constructing the Progressive Octree (2): The encoder recursively subdivides the point cloud-aligned bounding box into eight children. Only nonempty children voxels continue to be subdivided. This results in an octree data structure, in which the position of each voxel is represented by its cell center. Its attribute (color) is set to the average of enclosed points and needs to be coded separately by the

attribute encoder. Each level in the octree structure can represent a LoD. The final LoD is specified by the octree bit settings. This is a common scheme for both regularizing the unorganized point cloud and compressing it (see Section III-B for details).

3) Coding of the Occupancy Codes (3) (LoD): To code the octree structure efficiently, the first step is the encoding of LoDs as subdivisions of nonempty cells. Contrary to [11] and [12], we develop a position coder that uses an entropy coder in the corresponding LoDs. In addition, by avoiding surface approximations, as in [12], and occupancy reordering, as in [11], we keep the occupancy code encoder as simple and fast as possible (see Section III-B for details). In particular, we set the level of decodable LoDs to two, thus reducing overhead.

4) Coding of Color Attributes (4): In order to code the color attributes in the point cloud in a highly efficient manner, we integrated methods based on mapping the octree traversal graph to a JPEG image grid, exploiting correlation between color attributes in the final LoDs. The rationale of the JPEG mapping is that as the color attributes result from natural inputs, comparable correlation between adjacent pixels/points exists. By mapping the octree traversal graph to a JPEG grid, we aim to exploit this correlation in an easy and fast way that is suitable for real-time 3D tele-immersion.

5) Inter-Predictive Frame Coding (5): We introduce a method for inter-predictive encoding of frames based on previous inputs, which includes both rigid transform estimation and rigid transform compensation. We first compute a common bounding box between the octree structures of consecutive frames, i.e., block (1). Then, we find shared larger macroblocks on $K(=4)$ levels above the final LoD voxel size. For blocks that are not empty in both frames, color variance and the difference in point count are used to decide if we have to compute a rigid transform based on the iterative closest point (ICP) algorithm. If this is the case and the ICP algorithm is successful (converges), the computed rigid transformation can be used as a predictor. The rigid transform is stored compactly in a quaternion and translation quantization schema (6). This prediction scheme can typically save up to 30% bitrate. This is important to reduce the data volume at high capture rates.

6) Coder Control (8) and Header Formatting (7): The coder uses prespecified codec settings (via a configuration file), which include the octree bit allocation for (2), macroblock size and prediction method for (5), and color bit allocation and mapping mode for (4). In addition, it includes the settings for the filter and the color coding modes. We use a common header format and entropy coding based on a static range coder to further compress these fields.

III. Intra-Frame Coder

The intra-frame coder consists of three stages (1, 2, and 3 in Fig. 4). It first filters outliers and computes a bounding box. Second, it performs an octree composition of space. Third, entropy encoding of the resulting occupancy codes is performed.

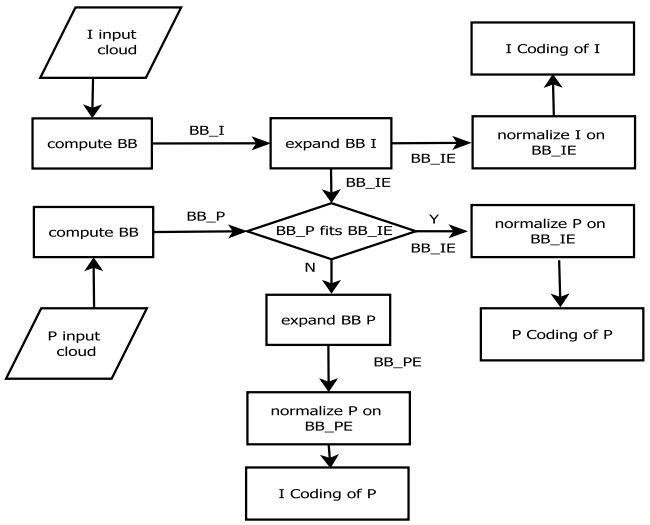

Fig. 5. Bounding box alignment scheme.

A. Bounding Box Normalization and Outlier Filter

The bounding box of a mesh or point cloud frame is typically computed as a box with a lower corner *(x_min, y_min, z_min)* and an upper corner *(x_max, y_max, z_max)*. This bounding box is used as the root level of the octree. The bounding box can change from frame to frame, as defined by these extrema. As a consequence, the correspondence of octree voxel coordinates between subsequent frames is lost, which makes inter prediction much harder. To mitigate this problem, Fig. 5 illustrates a scheme that aims to reduce bounding box changes between adjacent frames. The scheme enlarges (expands) the bounding box by a certain percentage δ, and then, if the bounding box of the subsequent frame fits this bounding box, it can be used instead of the original bounding box. As shown in Fig. 5, the bounding box of an intra frame is expanded from the BB_IE ($x\_\text{min} - \text{bb\_exp}$, $y\_\text{min} - \text{bb\_exp}$, $z\_\text{min} - \text{bb\_exp}$) to upper corner ($x\_\text{max} + \text{bb\_exp}$, $y\_\text{max} + \text{bb\_exp}$, $z\_\text{max} + \text{bb\_exp}$), where bb_exp was computed from δ and the ranges of the original bounding box. Then, the subsequent P cloud is loaded and if a bounding box computed for this frame, BB_P, fits the expanded bounding box BB_IE, P is normalized on BB_IE. BB_IE is subsequently used as the octree root and frame P can be used by the predictive coding algorithm presented in Section IV. Otherwise, the expanded bounding box is computed as BB_PE and cloud P is normalized on BB_PE. In this case, cloud P is intra coded instead, as our predictive coding algorithm works only when the bounding boxes are aligned.

The bounding box operation is an important preprocessing step to enable efficient intra- and inter-coding schemes. We assume that point clouds represent segmented objects reconstructed from multicamera recordings, and in our experiments, we also work with such data. We discovered that in some cases, the segmentation of the main object (human or object) is not perfect and several erroneous background/foreground points exist in the original cloud. As these points can degrade the bounding box computation and

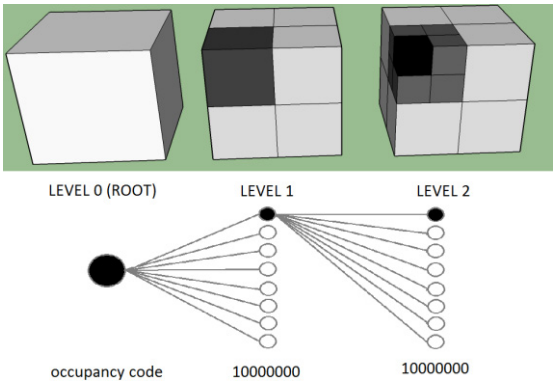

Fig. 6. Octree composition of space.

0	1	2	3	4	5	6	7	64
15	14	13	12	11	10	9	8	79
16	17	18	19	20	21	22	23	80
31	30	29	28	27	26	25	24	…
32	33	34	35	36	37	38	39	…
47	46	45	44	43	42	41	40	…
48	49	50	51	52	53	54	55	…
63	62	61	60	59	58	57	56	…

Fig. 7. Scan line pattern for writing octree values to an image grid.

alignment scheme, we have implemented filters to preprocess the point cloud. Our method is based on a radius removal filter. It removes points with less than K neighbors in radius R from the point cloud. This filter removes erroneously segmented points from the point cloud and improves the performance of the subsequent codec operations.

B. Octree Subdivision and Occupancy Coding

Starting from the root node, each octree subdivision results in an 8-b occupancy code, as shown in Fig. 6. Only nonempty voxels are subdivided further and the decoder only needs the occupancy codes to reconstruct each LoD of the point cloud. Therefore, efficient coding of occupancy codes is central to point cloud codecs [12], [13]. The work in [12] aims at reducing the entropy by using surface approximations at each level to predict likely occupancy codes. The method in [11] introduces a method based on reordering the order of the bits in the occupancy code by traversing the eight child cells in different orders. This traversal order is computed by identifying the point neighborhood in each subdivision. These approaches are not so practical for 3D immersive video based on point clouds, as such continuous surface approximations and statistics are computationally expensive and hinder parallelization. Instead, we follow a modified approach, taken from [3] (based on a carry-less-byte-based range coder), which we applied to the different decodable LoDs. In our experiments, this gave better results for compression of all LoDs at once.

While previous work on progressive static point cloud compression enabled many decodable LoDs for interactive rendering, we limited the number of decodable LoDs in our implementation to two to avoid introducing a large overhead in the coding of color attributes that need to be coded for each level.

C. Color Encoder

Apart from coding the object geometry position, coding of color attributes is essential to achieve a high visual quality rendering. In the past, methods based on Digital Pulse Code Modulation (DPCM) [12], colorization (keeping a frequency table of colors) [13], and octree structures have been proposed [12]. However, these methods are all characterized by low compression efficiency as they cannot exploit correlation between multiple colocated points. An attempt to improve this was made in [6] by modeling subsets of points as a weighted graph that can be directly derived from the octree. The color attributes are then modeled as a signal on this graph. The signal values are then projected on each of the eigenvectors, resulting in spectral representation. This method imposes a large computational computing overhead the eigenvalues and eigenvectors. Therefore, instead, we introduce a method based on existing legacy JPEG codec that exploits correlation present in the point cloud reconstructed from natural inputs. Instead of mapping the octree directly to a graph and treating the color attributes as a graph signal, we map the color attributes directly to a structured JPEG image grid from a depth-first tree traversal. We have defined a mapping that enables efficient coding of colors based on a zigzag pattern. As shown in Fig. 7, the colors are written to 8×8 blocks based on the depth-first octree traversal. The grid entries in Fig. 7 correspond to the order of octree traversals, while the grids themselves correspond to pixels in the preallocated image grid. So, based on the depth-first octree traversal pixels, are written to an image grid. This grid is reallocated based on the original leaf count of the octree. The method takes advantage of the fact that in the depth-first traversal, subsequent pixels are often colocated and therefore correlated. The Discrete Cosine Transform transform in the JPEG codec later exploits this from the mapping to 8×8 blocks. As for some octree traversal steps (big jumps) the assumption of correlation does not hold, some distortion could be introduced. This can be combatted by storing residuals. However, subjective assessment of the decoded data in Section V-E did not assert that distortion introduced in this method was perceptually significant.

IV. Inter-Frame Coder

To perform inter-frame encoding between subsequent point clouds, we present an inter-frame prediction algorithm that reuses the intra-frame coder presented in the previous section in combination with a novel lossy prediction scheme based on ICP algorithm. All abbreviations used to describe the algorithms are given in Table I.

TABLE I
SYMBOLS USED IN INTER-PREDICTIVE POINT CLOUD COMPRESSION ALGORITHM

Symbol	Description
ICP	Iterative Closest Points
I	Preceding Reference Point Cloud intra frame
P	Point Cloud that will be predictively encoded
C_p	Predictively coded compressed part of *P*
C_i	Intra coded compressed part of *P*
M_i	*I* frame Macroblocks organized in an octree
M_p	*P* frame Macroblocks organized in an octree
M_s	Macroblocks non empty in both I and P
M_px	Macroblocks non-empty in *P* only
M_p_i	Macroblocks in P that could not be predicted
pc	Point count range percentage
C_var_thresh	Threshold for color variance between blocks
icp_fit_thresh	Threshold for ICP convergence
k	Key in the octree structure (x,y,z)
T	Rigid Transformation Matrix (4 by 4)
R	Rotation Matrix (3by 3)
t	Translation vector
q_o(s,t,u,v)	Quaternion vector
q_d(s,t,u,v)	Decoded Quaternion Vector

A. Predictive Algorithm

Fig. 8 outlines the inter-predictive coding algorithm. The algorithm codes the data in the P frame in two parts. An I-coded part (***C_i***) of data is the one that contains the vertices that could not be predictively coded and a P-coded part (***C_p***) with data that could be predicted well from the previous frame. The algorithm starts with the normalized and aligned I and P clouds (based on the bounding box alignment algorithm presented in Section III-A). The macroblocks ***M_i*** are (1) generated at level K above the final LoD of octree O, which was generated coding the intra frame in the previous iteration. We chose $K = 4$ resulting in macroblocks of $16 \times 16 \times 16$. While $K = 3$ resulting in $8 \times 8 \times 8$ or $K = 5$ resulting in $32 \times 32 \times 32$ blocks are also possible, $K = 4$ gave the best results in terms of block overlap and final convergence. A similar macroblock octree at $K = 4$ is also computed for P resulting in ***M_p*** (1).

In the next step, (2) each of the macroblocks ***M_p*** is traversed to find if a corresponding macroblock exists in ***M_i***. For each of these blocks ***M_s*** (which are nonempty in I and P), the inter-predictive coding algorithm continues. Blocks occupied only in ***M_p*** and not in ***M_i*** are stored in ***M_px*** and written to the *intra-coded part (C_i)* of the compressed data, which will be coded with the intra coder presented in Section III. In the current implementation, all data coded to the intra-coded part is written to a point cloud data structure, which will be later coded with the intra-coding algorithm from Section III.

Each block in ***M_s*** will be a candidate for the prediction; two extra conditional stages are traversed before the predictive coding of the block is started. First, the number of points is asserted to be in the range of the two corresponding

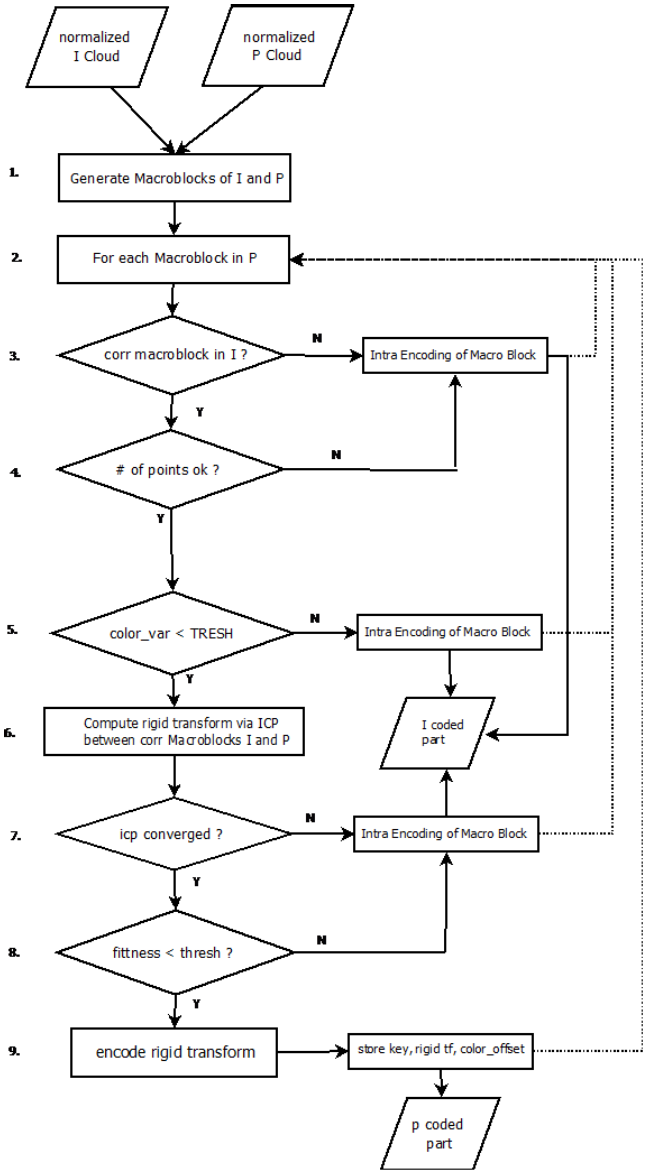

Fig. 8. Inter-predictive point cloud coding algorithm.

macroblocks in ***M_i*** and ***M_p*** (4). We set the threshold to do this to plus or minus *pc*%. Only when the number of points is in range between the two blocks, prediction is possible. We set the value of *pc* to 50%. This parameter *pc* can be tuned by comparing the percentage of macroblocks that are shared with the percentage of blocks suitable for prediction and the resulting quality. In the second stage, a check is done on the color variance of the points in the macroblock. We perform inter prediction only in areas of the point cloud that have low color/texture variance. The reason is that inter-predictive coding may result in visible artifacts in high-variance-texture image regions. The prediction is performed based on computing a rigid transform between the blocks mapping the points M_i to M_p. This computation is based on the ICP algorithm, which takes only the geometric positions into account and not the colors. The threshold of the total variance in the blocks *C_var_thresh* was set to 100 in our experiments. In these corresponding low-variance macroblocks, we can then finally

TABLE II
DATA STRUCTURE OF AN INTER-CODED BLOCK OF DATA

Key_x		Key_y		Key_z
Quat1		Quat2		Quat3
T1		T2		T3
C_off1	C_off2	C_off3		

compute a motion vector as a rigid transform via *ICP*. In case the *ICP* converges and the achieved fitness level is below a certain threshold (*icp_fit_thresh*) (in our case this was chosen to be four times the target resolution of the point), we code the predictor [that is, the rigid transform and the (x, y, z) key k in the macroblock grid $\boldsymbol{M\_p}$]. Otherwise the points are written to the intra-coded part. This is a point cloud data structure that is coded after the inter prediction terminates. The encoding of rigid transformation $\mathbf{T}$ is as follows. It is first composed as a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$. The rotation matrix $\mathbf{R}$ is converted into a quaternion **q_or (s, t, u, v)** and quantized using a quaternion quantization scheme of only three numbers Quat1, Quat2, and Quat3 (16 b per number). The translation vector $\mathbf{t}$ is quantized using 16 b per component (T1, T2, T3). Further details on the scheme for rigid transform coding resulting from ICP are presented in Section IV-B.

Last, a color offset is optionally coded to compensate for color difference offsets between the corresponding macroblocks. This color offset can optionally be used to compensate for fixed color offsets between blocks due to changes in lighting that have resulted in a brightness difference. The position in $\boldsymbol{M\_p}$ is stored as a key (x, y, z) k using 16-b integer values for each component and corresponds to a macroblock in $\boldsymbol{M\_i}$ (k).

This key can be used to decode the predicted blocks directly from the previously decoded octree frame $\boldsymbol{M\_i}$ *in any order*. This random access indexing method also enables parallel encoding/decoding of the data, which is important for achieving real-time performance. The memory outline of the data field is presented in Table II. Each row represents 6 B, and the data structure consists of 18–21 B (as coding the color offset is optional). The final *predictively coded part* **C_p** of the byte stream consists of a concatenation of all predicted blocks resulting from the prediction. All blocks that could not be predicted $\boldsymbol{M\_p\_i}$ are stored in a single point cloud that is coded using the method developed in Section III.

B. Rigid Transform Coding

The 4×4 rigid transform matrix $\mathbf{T}$ resulting from ICP-based prediction is composed into a (unitary) 3×3 rotation matrix $\mathbf{R}$ and a 3×1 translation vector $\mathbf{t}$

$$\begin{matrix} \boldsymbol{R} & \boldsymbol{t} \\ \mathbf{0} & 1 \end{matrix}.$$

Matrix rotation $\boldsymbol{R}$ is more compactly represented as a four-element quaternion (see the Appendix for conversion formulas)

$$\boldsymbol{q}(s, t, u, v).$$

The quaternion is not only more compact, but also makes the rotation more susceptible to round-off errors that are introduced during the quantization of its elements. The resulting quaternion is a unit quaternion that satisfies the relationship

$$s^2+t^2+u^2+v^2=1.$$

This allows one component to be coded implicitly based on this relationship. On the basis of [23], we chose the largest component of $\boldsymbol{q}$ to be coded implicitly. In addition, the sign of this element need not be stored, as we can make sure it is always positive (by negating the quaternion, this does not change the represented rotation). We scale the other values in the range $[0, 1/\sqrt{2}]$, as $1/\sqrt{2}$ is the maximum possible value of the second largest element of the quaternion. This scheme allows the rotation to be stored by three elements that we quantize using 16 b (including one sign bit). The decoded quaternion **q_d** may still suffer from round-off errors and numerical instability. In our experiments, we have observed that this is in less than 4% of the cases. We test for these cases in the encoder, and when they occur, we quantize the two linearly independent rows $\boldsymbol{R}$ and a sign vector for the third dependent vector to code the rotation matrix directly. This alternative quantization scheme guarantees correct recovery of rotation in all cases. Vector $\boldsymbol{t}$ is quantized using 16 b per component.

C. Parallelization

The proposed algorithm and bit streams have been designed to enable parallel execution. The algorithm in Fig. 8 can be parallelized as follows. First, the generation of macroblocks in I and P (1) can happen in parallel. Next, in the traversal of macroblocks in $\boldsymbol{M\_p}$ (2), operations (5) and especially (6) are most computationally intensive but can happen using parallel computation. The computation of color variance (5) in a point cloud subset can easily be offloaded to a Graphics Processing Unit. Most importantly, the ICP prediction can be parallelized, as the ICP prediction on each macroblock in $\boldsymbol{M\_s}$ can happen independently. In addition, due to the data structure outlined in Table II, the results can be written to C_p in any order, as the key index also enables coding blocks independently. We have implemented this using OpenMP for multicore processor Intel architectures.

The I-coded ($\boldsymbol{C\_i}$) part is a continuously updated point cloud with points that could not be predicted. This point cloud is later compressed (after the algorithm in Fig. 8 terminates) resulting in $\boldsymbol{C\_i}$ based on the octree-based scheme in Section III. As in our experiments, we observed that this algorithm spends a large fraction of the time in organizing the octree structure; parallelizing the octree composition is a good possibility to speed up this part of the algorithm. Octree data structures are common, and this has already been intensively studied, for example, in [24]. While such techniques can be used to speed up the algorithm, in this paper, we focus on parallelization of ICP prediction, which distinguishes our codec and uses a large fraction of the computation time as well.

V. Experimental Results

A. Data Sets and Experimental Setup

A Dell Precision M6800 PC with Intel core i7-4810MQ 2, 8-MHz CPU, and 16.0 GB of RAM running 64 win7 Operating system, a Dell Precision T3210 (Xeon 3.7 GHz), and a custom-built system with i7 3.2 GHz running Ubuntu Linux are the hardware used in the experiments. The data sets used for the evaluation (http://vcl.iti.gr/reconstruction/) are realistic tele-immersive reconstructed data based on [1] and [2]. Note that we made comparisons with the available real-time point cloud codec in [3] with octree composition and DPCM coding.

B. Objective Quality Evaluation Metric

To evaluate the point cloud quality, we deploy a full reference quality metric that combines common practices from 3D mesh and video compression: a Peak Signal to Noise Ratio (PSNR) metric based on point-to-point symmetric root-mean-square distances.

The original point cloud V_{or} is a set of K points without a strict ordering (where v contains information on both position and color)

$$V_{\text{or}} = \{(v_i) : i = 0 \ldots K-1\}. \tag{1}$$

The decoded cloud does not necessarily have the same number of points

$$V_{\text{deg}} = \{(v_i) : i = 0 \ldots N-1\}. \tag{2}$$

The full reference quality metric $Q_{\text{point\_cloud}}$ is computed by comparing V_{or} and V_{deg} as follows. Instead of the distance to surface, we take the distance to the most nearby point in V_{deg}, $v_{\text{nn\_deg}}$. We defined geometric PSNR in (5) as the peak signal of the geometry over the symmetric rms distance defined in (4). The color difference between the original cloud and the most nearby point in the degraded cloud $v_{\text{nn\_deg}}$ is used to compute the PSNR per YUV component (psnr_y, psnr_u, etc.) [see (7)]. The $v_{\text{nn\_deg}}$ is efficiently computed via a K-d tree in L2 distance norm based on algorithms available in [4]

$$d_{\text{rms}}(V_{\text{or}}, V_{\text{deg}}) = \frac{1}{\sqrt{K}} \sum_{v_l \in V_{\text{or}}}, \quad ||v_l - v_{\text{nn\_deg}}||_2 \tag{3}$$

$$d_{\text{sym\_rms}}(V_{\text{or}}, V_{\text{deg}}) = \max(d_{\text{rms}}(V_{\text{or}}, V_{\text{deg}}), d_{\text{rms}}(V_{\text{deg}}, V_{\text{or}})) \tag{4}$$

$$\text{psnr}_{\text{geom}} = 10\log_{10}\left(||\max_{x,y,z}(V_{\text{deg}})||_2^2 / (d_{\text{sym rms}}(V_{\text{or}}, V_{\text{deg}}))^2\right) \tag{5}$$

$$d_y(V_{\text{or}}, V_{\text{deg}}) = \frac{1}{\sqrt{K}} \sum_{v_l \in V_{\text{or}}}, ||y(v_l) - y(v_{\text{nn\_deg}})||_2 \tag{6}$$

$$\text{psnr}_y = 10\log_{10}\left(||\max_y(V_{\text{deg}})||_2^2 / (d_y(V_{\text{or}}, V_{\text{deg}}))^2\right). \tag{7}$$

Additional metrics for the quality assessment include partial bitrates for color, attribute and geometry data, and encoder/decoder time. These quality evaluation metrics are well aligned with existing practice in mesh and video compression and are recommended for evaluation of point cloud codecs [25]. For the evaluation of time-varying point cloud compression in 3D tele-immersive virtual room, we will also perform subjective tests with real users.

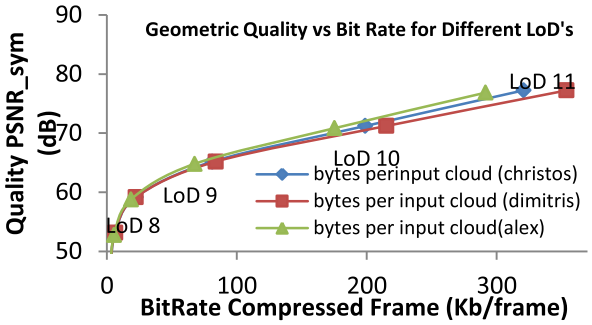

Fig. 9. LoD: bitrate versus quality.

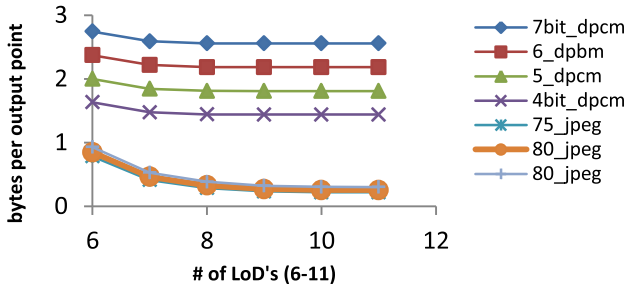

Fig. 10. Bytes per point for color coding per LoD.

C. Intra-Coding Performance

To illustrate the effect of LoD decoding the bit stream, we show the LoD, bitrate and quality, in Fig. 9. From our experience with the given data sets with the realistic 3D immersive system, LoD 11 gives a realistic representation when in close range of the point cloud. For point clouds at a distance in the room, lower-quality LoD would be sufficient. The results in Fig. 9 plot the quality metric defined in (3) based on PSNR versus bitrate in kilobytes per frame. In Fig. 10, we show the results of bitrate of the colors per output point for different LoDs. For the DPCM, quantization of 4–7 b per color component was set. For the JPEG color coding scheme, 75 and 80 have been used as the JPEG quality parameter. For higher LoDs, both the DPCM and JPEG color coding scheme are more efficient, as correlation between neighboring points increases. Overall, the color coding scheme based on JPEG provides much lower bitrates and lower quality. However, at approximately the same quality, bitrate is up to ten times lower compared with DPCM-based coding. Such a configuration is very useful for the targeted application of 3D tele-immersion, in which low bitrates are needed. In addition, the difference in objective quality does not necessarily correspond well to the subjective quality. In Section V-E, we will show in the subjective studies that the color quality degradation introduced by the color coding method is negligible even compared with 8-b DPCM-based coding.

In Figs. 11–13, we show the number of bytes per output point versus the objective color quality based on the metric defined in (7) for the Y, U, and V components for each of the different LoDs (7–11). As for higher LoDs, the qual-

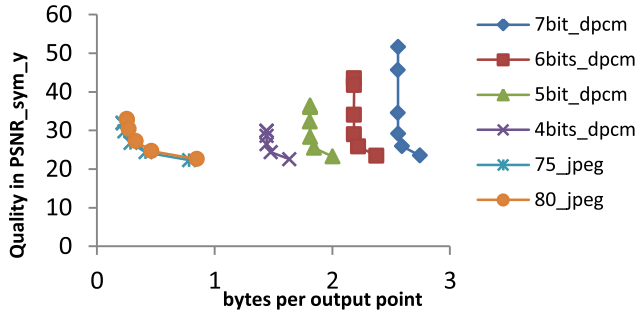

Fig. 11. R-D for color coding (Y component).

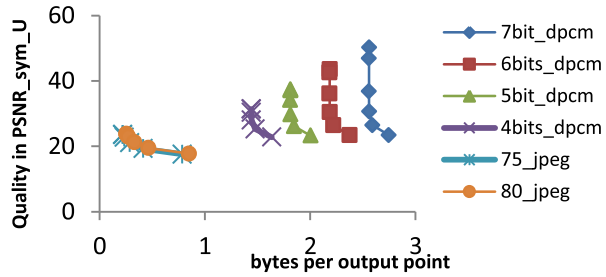

Fig. 12. R-D for color coding (U component).

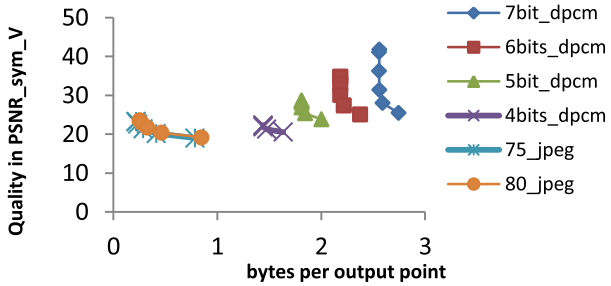

Fig. 13. R-D for color coding (V component).

ity increases (more points) and the bytes per output point decrease; therefore, curves show higher quality for lower bitrates. The JPEG-based scheme gives qualities compared with 4-b, 5-b, and sometimes even 6-b DPCM at up to ten times less bytes per output point. This is the main advantage gained by the color coding methods, i.e., low bitrate coding of the color attributes. As the method reuses a fast optimized JPEG implementation, we do not introduce any extra computation time compared with that for the DPCM-based methods (i.e., the encoding speed and decoding speed have been similar for both methods), which is an advantage compared with the methods in [6]. For the U and V components, the achieved quality with JPEG is slightly lower compared with that achieved with the Y component.

D. Inter-Predictive Coding Performance

Inter-predictive coding can be applied when two frames are aligned using the bounding box alignment scheme from Section III-A. We show in Fig. 14 the results of bounding box alignment on 3D tele-immersive point cloud data sets that we use in the evaluation. For a bounding box expansion factor of 20%, we achieve a good alignment percentage (>75%), and we use this setting in each of our following experiments.

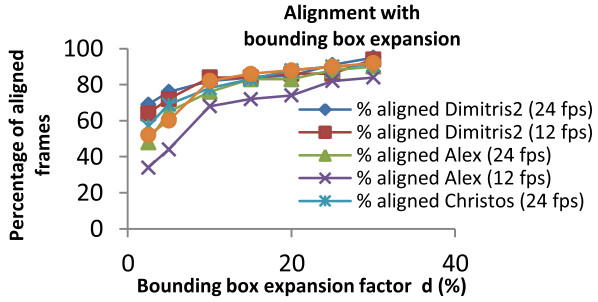

Fig. 14. Bounding box alignment.

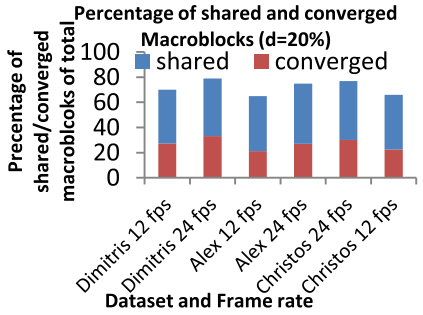

Fig. 15. Shared macroblock and convergence percentage.

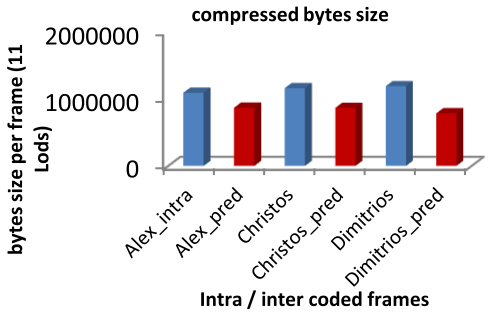

Fig. 16. Predicted versus intra-coded frame size.

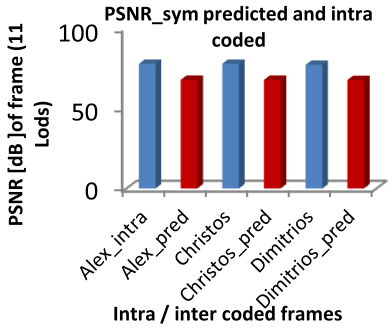

Fig. 17. Quality of predicted frames versus intra-coded frames.

In Fig. 15, we show the percentage of shared macroblocks in the coarse octree between aligned frames for LoD 11 (which is the preferred LoD for encoding and decoding) at $K = 4$ ($16 \times 16 \times 16$ blocks), $d = 20\%$, and the ICP convergence percentage. For all tested data sets (based on capturing at 12 or 24 frames/s), over 60% of macroblocks are shared in aligned frames and approximately 30% are both shared and converging at the ICP stage (these are used for prediction).

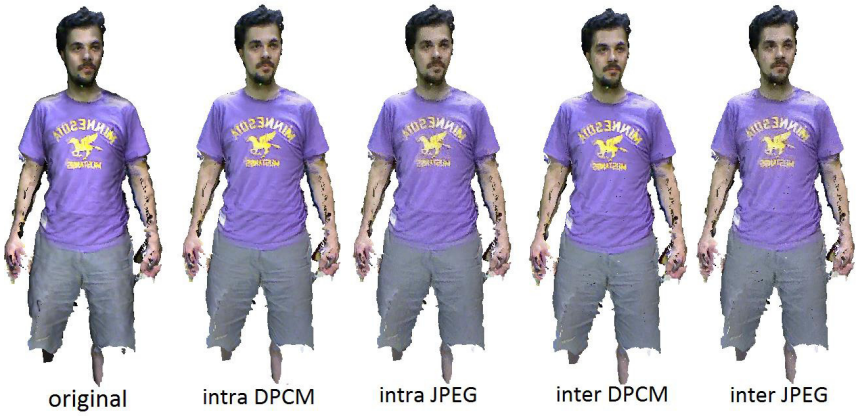

Fig. 18. Results at LoD Christos 11 (11-b octree).

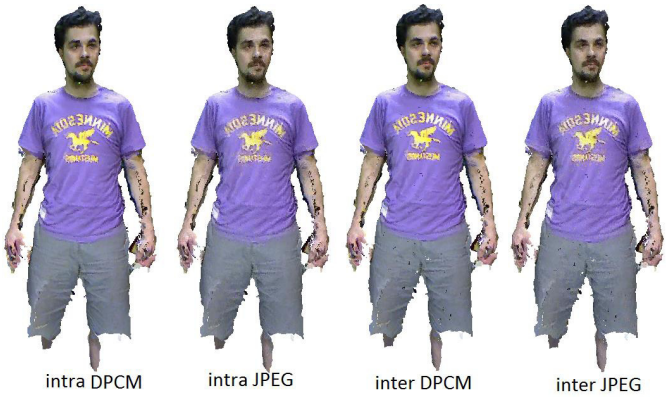

Fig. 19. Results at LoD Christos 10 (10-b octree).

Fig. 20. Results at LoD Alex 11 (11-b octree).

The percentage that is shared and converged is shown in red and relates largely to bitrate savings (which is over 30% for the Dimitrios data set that has the largest percentage of shared blocks). In Fig. 16, we compare the compressed size of intra-coded frames with that of predictive frames; for the Alex data set, the size was reduced by 20.5% compared with intra coding only; for Christos, this was 25.5%, and for Dimitris, 34.8%. Again, the bit saving relates largely to the percentage of shared blocks between the octree-based point clouds.

The comparison of the objective quality of predictively coded and intra-coded frames is shown in Fig. 17 based on metric (5). The loss in geometric PSNR in predictively encoded frames is about 10 dB in each of the tested data sets. We will explore the subjective effect of this degradation in the next section. In Figs. 18–21, we show the results for both predictive- and JPEG-encoded/decoded frames and the original for the Christos and Alex data sets. All point clouds have been rendered under the exact same conditions (lighting and camera position) using MeshLab Software.[2] For the LoD 11 and LoD 10 prediction, artifacts are not visible. For the data under test, a good convergence and shared block percentage were achieved for LoD 11 (11-b octree setting, i.e., 11 quantization bits per direction). For different data sets, some tuning could be necessary to find the best setting. The software implementation of codec can be configured via a parameter_config text file.

[2]http://meshlab.sourceforge.net/

Fig. 21. Results at LoD 10 (10-b octree).

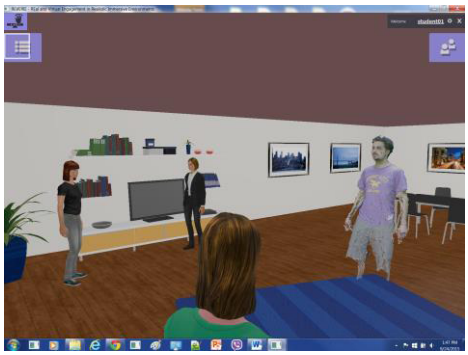

Fig. 22. Screenshot of point cloud rendering in virtual world.

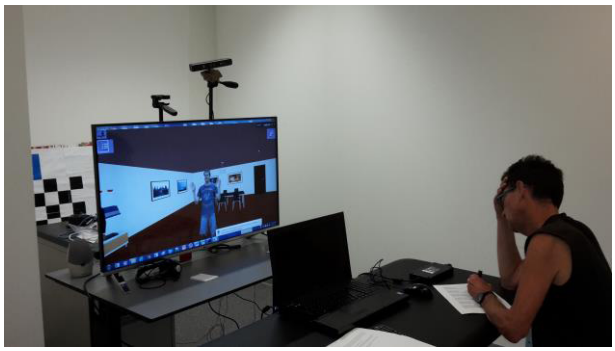

Fig. 23. Test setup with a user filling in the questionnaire.

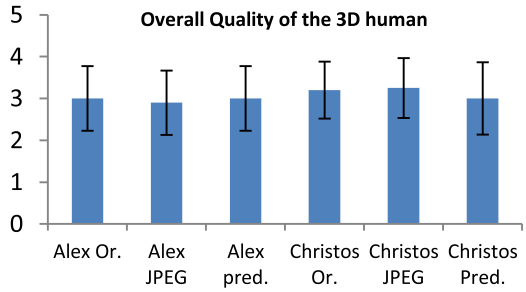

Fig. 24. Subjective results: perceived quality of the 3D human.

E. Subjective Quality Assessment

We evaluate the subjective quality of the codec performance in a realistic 3D tele-immersive system in a virtual 3D room scenario, in which users are represented and interact as 3D avatars and/or 3D point clouds. We show the system in Fig. 22. The user is represented as an avatar and can navigate in the room (forward, backward, left, right, and rotate) and look around with the use of the mouse (free viewpoint). The point cloud video represents the remote user that is rendered compositely in the room using OpenGL. The Reverie rendering system allows different modules to render compositely, which enables mixed reality of both synthetic (avatar) and naturalistic (point cloud) content in the scene. The system has implemented basic collision detection and navigation so that users cannot navigate through any of the occupied 3D space, point cloud, avatar, or other world objects.

Twenty test users have been recruited to work with the system and assess the performance. The users have been introduced to the goal and context of the experiment, and they signed the consent form for using this information for scientific purposes. The group consisted of four women and 16 men, in the age range from 24 to 62 years. All were working in the field of information technology (electronic engineering, computer science, or mathematics). Nationalities were mixed (ten Dutch, three Spanish, two Italian, two Chinese, two German, and one English), education levels were above B.Sc., and the participants have been exposed to different versions of point clouds at LoD 11, as shown in Figs. 18 and 20 (based on the point cloud data in Alex-Zippering and Christos Zippering available at http://vcl.iti.gr/reconstruction/). The inter-predicted frames are presented in an I-P-I-P interleaved pattern to simulate a realistic presentation. All data sets are rendered at 23 frames/s when presented to the user. (This corresponds to the capture rate.) Each of the users has been exposed to the three codec conditions (DPCM, JPEG, and inter predicted) on two different data sets (Alex and Christos) in a random order, resulting in a total of six test conditions. The controlled setup consisted of a 47-in LG TV (model num. 47LB670V) screen positioned at 1.5 m from the users running the Reverie system, as shown in Fig. 23, and a laptop + mouse to control the avatar. After each test condition, the users have been requested to fill a questionnaire on eight different quality aspects on a 1–5 scale ranging from bad to excellent based mean opinion score (MOS) (bad = 1, poor = 2, fair = 3, good = 4, excellent = 5). These aspects include the

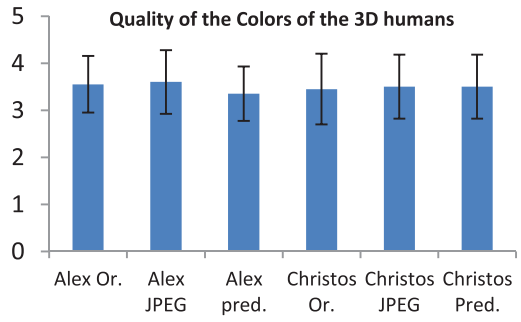

Fig. 25. Subjective results: colors of the 3D human.

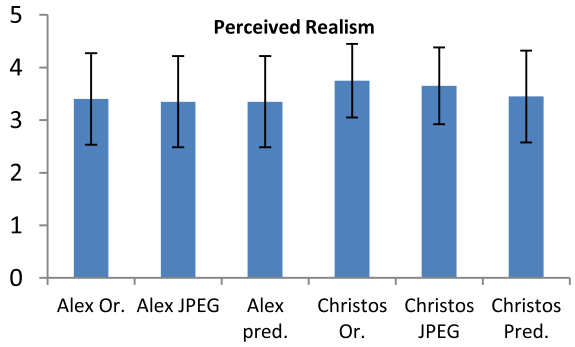

Fig. 26. Subjective results: quality of the motion.

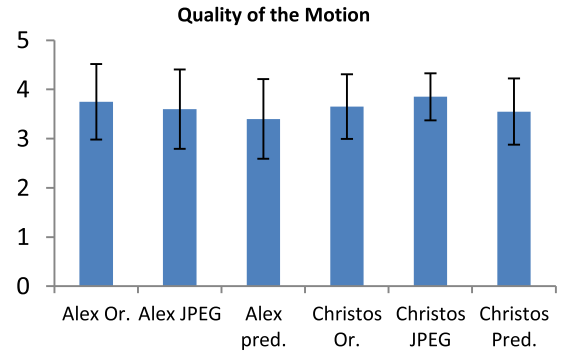

Fig. 27. Subjective results: perceived realism.

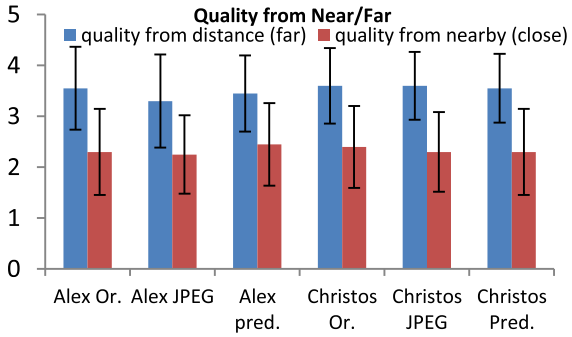

Fig. 28. Subjective results: quality from near versus far.

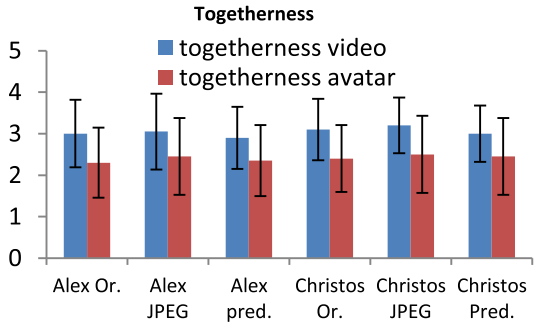

Fig. 29. Feeling of togetherness: avatar user versus 3D video user.

overall quality of the 3D human, the quality of the colors, the perceived realism of the point cloud, the motion quality, the quality from near/far, and how much the user felt together in a room with a real user comparing the avatar and the point cloud representations. The results in Figs. 24–29 can be interpreted as follows. The values 1–5 correspond to the MOSs of the results for each quality aspect. The error bars represent the standard deviation around the MOS. Due to the use of a low-cost capturing setup with multiple Microsoft Kinect 1, which is based on structured infrared light which results in interference, the original quality is already not free of artifacts. Nevertheless, many parts of the 3D reconstruction and the motion are photorealistic and natural, which was also indicated by most users. The main aim of our experiments is to check that the developed codec does not introduce significant extra subjective distortions. All point clouds are stored and decoded from an 11-LoD octree, with DPCM (Or. in Figs. 24–29), the proposed color coding (JPEG), and prediction (Pred. in Figs. 24–29). As can be seen in Fig. 24, for the Alex data set, the color coding and prediction do not introduce a significant degradation in quality, while for Christos some difference is observed (but not significant). The assessment of color quality in Fig. 25 compares the 8-b DPCM coding with the JPEG coding and inter prediction. Surprisingly, the JPEG color coding method does not introduce significant subjective distortion, as this method codes at up to ten times lower bitrates compared with the 8-b DPCM color data. The results on perceived realism and the quality of the motion are shown in Figs. 26 and 27. All versions represent the user well and are judged between fair and good at 3.45 and 3.75 without significant differences. Fig. 28 compares the perceived quality of the point cloud from near and far away; this shows that from nearby point, the point cloud quality is seen as low, while from a distance, it is between fair and good. This is mainly due to the fact that the rendering of the point clouds from the nearby point allows you to see through the points, while from a distance this is not the case. Alternative rendering methods or converting the point cloud to a mesh when nearby might solve this issue. Last, we investigated how the users valued the presence of the 3D point cloud video in the room compared with a simple computer avatar in terms of feeling together, which is the ultimate aim of telepresence technologies. In this respect, the point cloud representation scored significantly better than the computer avatar (Fig. 29).

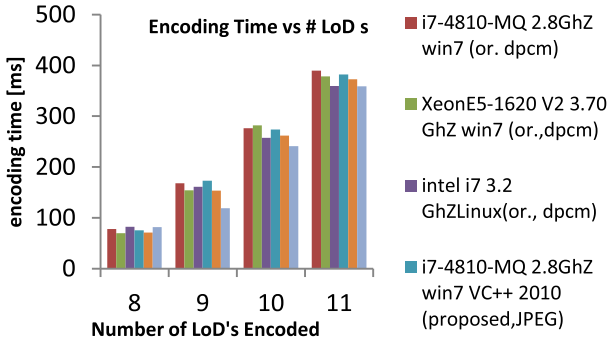

Fig. 30. Real-time performance of intra coding for different configurations.

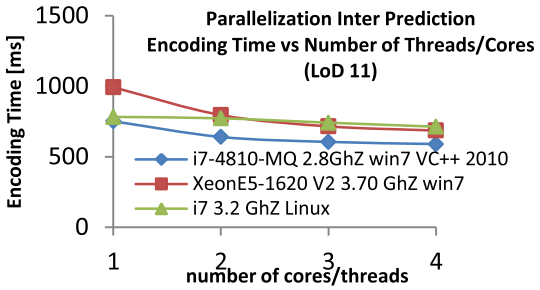

Fig. 31. Real-time performance of inter-predictive coding for different configurations.

F. Real-Time Performance and Parallelization

We have assessed the real-time performance of the codec on various computer hardware, showing that our codecs run in real time (<400 ms) at 11 LoDs for the intra coder and near real time (<1 s) for the inter-predictive encoder.

All tests have used the Dimitrios2-Zippering data set and ran on three Intel-based computers: the first based on i7 2.8 GHz and win7, the second based on Xeon 3.7 GHz running win7 (both compiled using VC++ 2010), and the third based on i7 3.20 GHz running Ubuntu Linux (gcc compiler suite). The real-time results for intra encoding for different LoDs and both proposed (JPEG) and original DPCM-based color coding are shown in Fig. 30. The decoding times are also lower (similar pattern) and are omitted due to space constraints.

In addition, we have been able to speed up the inter-predictive encoding significantly by parallelizing its execution based on OpenMP for multicore Intel architectures. (We measured up to 20% improvement on the windows platforms.) This hints in the direction that our design requirement for parallelization is met.

The real-time performance of the inter-predictive coding algorithm and its parallelization are shown in Fig. 31 for 11 LoDs. It plots the LoD's against the number of cores used to compute the ICP.

VI. Discussion and Conclusion

In this paper, we introduced a point cloud compression method that is suitable for 3D tele-immersive and mixed reality systems. We presented a hybrid architecture for point cloud compression, which combines typical octree-based point cloud compression schemes with hybrid schemes common in video coding. The architecture improves the state of the art by providing a mode for inter prediction on the unorganized point cloud and a lossy real-time color encoding method based on legacy JPEG methods. This enables easy implementation and real-time performance. We have implemented this architecture in an implementation that is both available as open source (https://github.com/RufaelDev/pcc-mp3dg, based on the point cloud library) and available in the important media standardization bodies (MPEG and JPEG) as the first reference platform for development of point cloud compression in MPEG-4 [22] and perhaps later for JPEG PLENO.[3] We rigorously evaluate the quality of our solution objectively using a novel PSNR-based quality metric that combines metrics from 3D meshes and video for point clouds. In addition, we performed subjective evaluation in a real mixed reality system that combines natural point cloud data and computer graphics-based 3D content with 20 users. The results show that the degradation introduced by the codecs is negligible. In addition, the point cloud has an added value in terms of feeling together compared with simple avatar representations, highlighting the importance of point clouds in these applications. These results further assert the importance of work on compression of point clouds and its standardization for immersive and augmented reality systems.

Appendix

A 3×3 rotation matrix R consisting of elements r_{ij} can be converted into a quaternion $\mathbf{q}(s,t,u,v)$ with s as the real part and t,u,v as the imaginary part as follows:

$$s = \frac{1}{2}\sqrt{1 + r_{11} + r_{22} + r_{33}}, \quad t = \frac{1}{4s}(r_{32} - r_{23})$$
$$u = \frac{1}{4s}(r_{13} - r_{31}), \quad v = \frac{1}{4s}(r_{21} - r_{12}).$$

Rotation matrix R can be recovered from $q(u,v,t,s)$ by:

$$\boldsymbol{R} = \begin{bmatrix} 1 - 2u^2 - 2v^2 & 2(tu - vs) & 2(tv + us) \\ 2(tu + vs) & 1 - 2t^2 - 2v^2 & 2(uv - ts) \\ 2(tv - us) & 2(ts + uv) & 1 - 2t^2 - 2u^2 \end{bmatrix}.$$

Acknowledgment

The authors would like to thank all partners in the Reverie FP7 consortium for contributing to the Reverie framework source code and their expertise. In particular, they would like to thank D. Alexiadis and P. Daras at CERTH/ITI for providing the point cloud data sets.

References

[1] A. Doumanoglou, D. S. Alexiadis, D. Zarpalas, and P. Daras, "Toward real-time and efficient compression of human time-varying meshes," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 12, pp. 2099–2116, Dec. 2014.

[2] D. S. Alexiadis, D. Zarpalas, and P. Daras, "Real-time, full 3-D reconstruction of moving foreground objects from multiple consumer depth cameras," *IEEE Trans. Multimedia*, vol. 15, no. 2, pp. 339–358, Feb. 2013.

[3] http://www.jpeg.org/items/20150320_pleno_summary.html

[3] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach, "Real-time compression of point cloud streams," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2012, pp. 778–785.
[4] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Shanghai, China, May 2011, pp. 1–4.
[5] D. Thanou, P. A. Chou, and P. Frossard. (2015). "Graph-based compression of dynamic 3D point cloud sequences." [Online]. Available: http://arxiv.org/abs/1506.06096
[6] C. Zhang, D. Florêncio, and C. Loop, "Point cloud attribute compression with graph transform," in *Proc. IEEE Int. Conf. Image Process.*, Paris, France, Oct. 2014, pp. 2066–2070.
[7] A. Vetro, T. Wiegand, and G. J. Sullivan, "Overview of the stereo and multiview video coding extensions of the H.264/MPEG-4 AVC standard," *Proc. IEEE*, vol. 99, no. 4, pp. 626–642, Apr. 2011.
[8] K. Müller *et al.*, "3D High-Efficiency Video Coding for multi-view video and depth data," *IEEE Trans. Image Process.*, vol. 22, no. 9, pp. 3366–3378, May 2013.
[9] (Aug. 2015). *REal and Virtual Engagement in Realistic Immersive Environments, Reverie FP7*. [Online]. Available: http://www.reveriefp7.eu/
[10] 3DLife. (2015). *3D Human Reconstruction and Action Recognition From Multiple Active and Passive Sensors*. [Online]. Available: http://mmv.eecs.qmul.ac.uk/mmgc2013/
[11] Y. Huang, J. Peng, C.-C. J. Kuo, and M. Gopi, "A generic scheme for progressive point cloud coding," *IEEE Trans. Vis. Comput. Graphics*, vol. 14, no. 2, pp. 440–453, Mar./Apr. 2008.
[12] R. Schnabel and R. Klein, "Octree-based point-cloud compression," in *Proc. SPBG*, 2006, pp. 111–120.
[13] J. Peng, C.-S. Kim, and C.-C. J. Kuo, "Technologies for 3D mesh compression: A survey," *J. Vis. Commun. Image Represent.*, vol. 16, no. 6, pp. 688–733, Dec. 2005.
[14] R. Mekuria, P. Cesar, and D. Bulterman, "Low complexity connectivity driven dynamic geometry compression for 3D tele-immersion," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Florence, Italy, May 2014, pp. 6162–6166.
[15] J. Hou, L.-P. Chau, Y. He, and N. Magnenat-Thalmann, "A novel compression framework for 3D time-varying meshes," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Melbourne, VIC, Australia, Jun. 2014, pp. 2161–2164.
[16] A. Doumanoglou, D. Alexiadis, S. Asteriadis, D. Zarpalas, and P. Daras, "On human time-varying mesh compression exploiting activity-related characteristics," in *Proc. IEEE ICASSP*, Florence, Italy, May 2014, pp. 6147–6151.
[17] R. Mekuria and P. Cesar, "A basic geometry driven mesh coding scheme with surface simplification for 3DTI," *IEEE Comsoc MMTC e-Letter*, vol. 9, no. 3, pp. 6–7, May 2014.
[18] K. Mamou, T. Zaharia, and F. Prêteux, "TFAN: A low complexity 3D mesh compression algorithm," *Comput. Animation Virtual Worlds*, vol. 20, nos. 2–3, pp. 343–354, Jun. 2009.
[19] K. Mamou, T. Zaharia, and F. Prêteux, "FAMC: The MPEG-4 standard for animated mesh compression," in *Proc. 15th IEEE Int. Conf. Image Process. (ICIP)*, San Diego, CA, USA, Oct. 2008, pp. 2676–2679.
[20] W. Wu, A. Arefin, G. Kurillo, P. Agarwal, K. Nahrstedt, and R. Bajcsy, "Color-plus-depth level-of-detail in 3D tele-immersive video: A psychophysical approach," in *Proc. 19th ACM Int. Conf. Multimedia (MM)*, 2011, pp. 13–22.
[21] P. Kauff and O. Schreer, "An immersive 3D video-conferencing system using shared virtual team user environments," in *Proc. 4th Int. Conf. Collaborative Virtual Environ. (CVE)*, 2002, pp. 105–112.
[22] J. Ostermann, *MPEG-4 Requirements*, document N14662, JTC1/SC29/WG11, MPEG, Sapporo, Japan, 2014.
[23] Z. Adami, "Quaternion compression," in *Game Programming Gems*, D. Treglia, Ed. Boston, MA, USA: Charles River Media, 2002, ch. 2.4, pp. 187–191.
[24] B. Hariharan and S. Aluru, "Efficient parallel algorithms and software for compressed octrees with applications to hierarchical methods," *Parallel Comput.*, vol. 31, nos. 3–4, pp. 311–331, Mar./Apr. 2005.
[25] L. Bivolarski, R. Mekuria, and M. Preda, *Preliminary Guidelines for Evaluation of Point Cloud Compression*, document N15578, ISO/IEC JCT1 SC29 WG11 (MPEG), Geneva, Switzerland, 2015.

Rufael Mekuria (S'15) received the B.Sc. and M.Sc. degrees in electrical engineering from Delft University of Technology, Delft, The Netherlands. He is currently working toward the Ph.D. degree with Vrije Universiteit Amsterdam, Amsterdam, The Netherlands

He joined Centrum Wiskunde & Informatica, Amsterdam, as a Researcher, in 2011. He was involved in the FP7 Project Reverie and has been active in international standardization activities in the International Organization for Standardization/International Electrotechnical Commission (ISO/International Electrotechnical Committee) The Moving Picture Experts Group since 2013. He has published in leading conferences and journals of the IEEE/Association for Computing Machinery/International Society for Optics and Photonics in the area of image processing and multimedia systems (including four invited and one distinguished paper). His research interests include 3D teleimmersion, 3D compression of point clouds and meshes, 3D network streaming and protocols, and multimedia systems.

Kees Blom has been a Scientific Programmer with Centrum Wiskunde & Informatica, Amsterdam, The Netherlands, since 1983. He has also been involved in implementations of graphics standards and languages for parallel and distributed systems and interactive multimedia systems. He is currently involved in the implementation of point cloud codecs.

Pablo Cesar (M'07) received the Ph.D. degree from Helsinki University of Technology, Espoo, Finland, in 2005.

He leads the Distributed and Interactive Systems Group with Centrum Wiskunde & Informatica, Amsterdam, The Netherlands. He is involved in standardization activities, such as W3C, The moving picture Experts Group, and International Telecommunication Union, and has been active in a number of European projects. He has co-authored over 70 articles (conference papers and journal articles) about multimedia systems and infrastructures, social media sharing, interactive media, multimedia content modeling, and user interaction.

Dr. Cesar has given tutorials about multimedia systems in prestigious conferences, such as Association for Computing Machinery Multimedia and the World-Wide Web Conference.