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

R. Mekuria, *Student Member IEEE*, K. Blom, P. 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 where 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 encoder and decoder. Objective evaluation shows that a higher rate-distortion (R-D) performance is achieved compared to available point cloud codecs. A subjective study in a state of art mixed reality system shows that introduced prediction distortions are negligible compared to 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 JCT1/SC29/WG11 (MPEG) for the further development of standardized point cloud compression solutions.**

***Index Terms*— Data Compression, Video Codecs, Tele Conferencing, Virtual Reality, Point Clouds**

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. They do not only allow free-view point rendering (for example based on splat rendering), but can also be compositely rendered in a synthetic 3D scene as they provide full 3D geometry coordinates information. Therefore, this type of video representation is preferable in mixed reality systems, such as *augmented reality* where a natural scene is combined with synthetic (authored e.g. computer graphics) objects. Or, vice versa, in *immersive virtual rooms* where 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] [4] [5] [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, 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 in these systems is *real-time performance* for encoding and decoding. Often, in modern systems, parallel computing that exploits multi-core architectures available in current computing infrastructures is utilized. Therefore, the *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 was 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 a sense that it can 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, 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 JCT1/ SC29/WG11 (MPEG)[1]. To facilitate benchmarking, the objective quality metrics and file loaders used in this work have all been included in the package. In addition, the point cloud test data is available publicly.

The rest of the 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 and real-time performance assessment. In section II we provide the overview of the codec and in this section, the context and related work.

A. Contributions

In this work we propose a novel compression framework for progressive coding of time varying point clouds for 3D immersive and augmented video. The major contributions are:

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

B. Augmented and Immersive 3D Video vs 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 AVC 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 view point 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.

In such systems, rendering is usually done based on object geometry such as meshes or 3D point clouds using generic computer graphics API's (e.g. OpenGL, Direct3D etc.). This is in line with common practice in computer games and virtual worlds. Therefore, to enable true convergence between naturalistic and synthetic content in immersive and augmented reality, object based video compression of point clouds and 3D Meshes is a remaining 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, either represented by 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

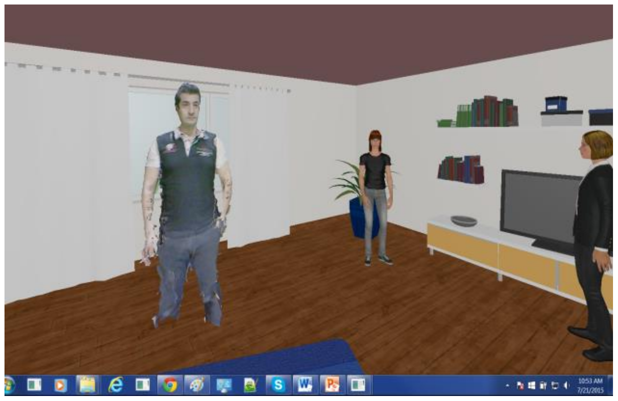


Fig. 1 Screenshot of composite rendering in virtual room based on the Reverie System. The Point cloud naturalistic content is rendered compositely with synthetic content. Navigation and user support enables interaction between naturalistic point cloud and synthetic avatar users

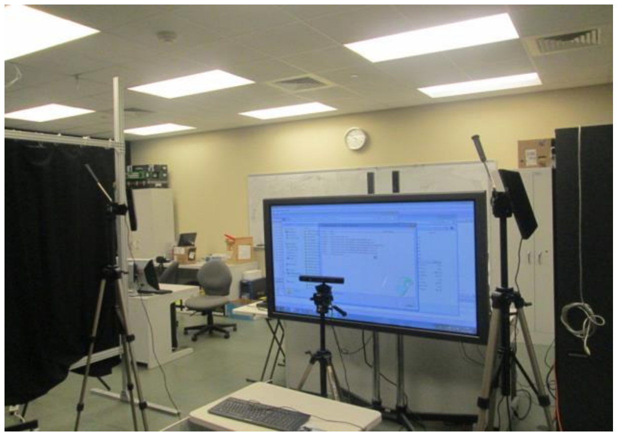


Fig. 2 Low cost point cloud capturing setup, point clouds are reconstructed from multiple 3D input streams of calibrated Microsoft Kinect devices.

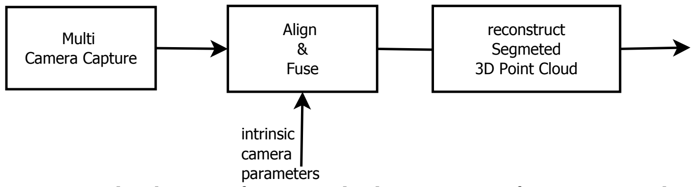

Multi Camera Capture
Align & Fuse
reconstruct Segmeted 3D Point Cloud
intrinsic camera parameters

Fig.3 Example schematic of 3D point cloud reconstruction for immersive video. Many variations can be implemented

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

`

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 (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.

Alternatively, by running the algorithm in Fig. 3 at the 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 (8-9-10 bits 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 of132Kbyte-800Kbyte per frame were achieved by using RGB-D coding, while bit-rates of 40 -265 KiloBytes were obtained using the octree based point cloud compression. In addition, the distortion introduced to the point cloud, which could be unpredictable for low bit rate 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 tele-presence and immersive reality low latency, low bit-rate and bound distortion are critical; we develop compression for time varying point clouds.

C. Related Work

Point Cloud Compression: There has been some work on point cloud compression in the past, but most works only aim at compression of 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. Instead, 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 only significant for scenes with limited movement (which is not the case in out envisioned application with moving humans). Last, [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, resulting in a lossless encoding. This method also includes the color coding method from [6], which defines a small sub-graphs 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.

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 bit-rate overhead [14]. Especially, some recent work has aimed at compression of object based immersive 3D video [15] [16] [17] uses 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 is 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 are important (analogous to video coding in video conferencing systems compared to video on demand).

Multi-View Plus Depth Compression: Multi-View plus depth representation was considered for storing video and depth maps from multiple cameras [8] [7]. Arbitrary viewpoints are then rendered by interpolation between different camera views using techniques from depth image based rendering (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.

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 in a color + depth video), and a purely perception based compression mechanism combined with entropy encoding. In this work the level of detail (polygon size) is adapted to match user/application needs. These needs were derived offline in subjective studies with pre-recorded 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 towards the specific tele-immersive configuration. This lack of generality makes it harder to assess the relevance and performance of the video compression compared to other methods. In our work 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.

`

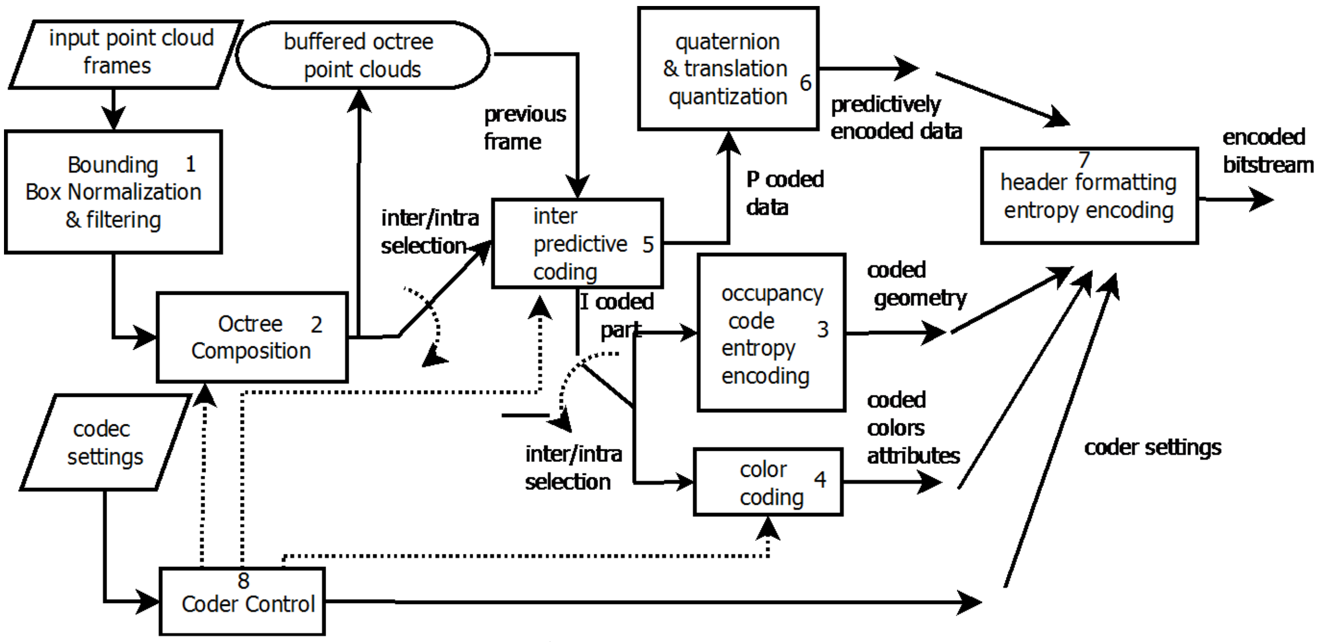


Fig. 4 Schematic of time-varying point cloud compression codec

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]:

Partial Bit Stream Decoding: it is possible to decode a coarse point cloud and refine it.

Lossless Compression: the reconstructed data is mathematically identical to the original.

Lossy Compression: compression with parameter control of the bit-rate.

Time Variations and Animations: temporal variations i.e. coding of point cloud sequences, should be supported

Low Encoder and Decoder Complexity: this is not a strict requirement but desirable for building real-time systems.

B. *Schematic Overview*

The architecture design of the proposed 3D Video based on point clouds combines features from common 3D Point cloud (octree based) codecs ([12] [11]) and common hybrid video codecs such as MPEG-4 p.10 AVC and HEVC (that 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.

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 both the effectiveness of the bounding box alignment and of inter predictive coding, and should be filtered out from the input clouds. See III.A for all details.

Constructing the Progressive Octree (2): The encoder recursively subdivides the point cloud aligned bounding box into eight children. Only non-empty children voxels continue to be subdivided. This results in an octree data structure, where the position of each voxel is represented by its cell center. Its attributes (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 level of detail (LoD). The final level of detail is specified by the octree bit settings. This is a common scheme for both regularizing the unorganized point cloud and for compressing it. (See III.B for all details).

Coding of the Occupancy Codes (3) (**LOD**): To code the octree structure efficiently, the first step is the encoding of (LOD's) as sub-divisions of non-empty cells. Contrary to previous work [12] [11], we develop a position coder that uses an entropy coder in the corresponding LoD's. In addition, by avoiding surface approximations as in [12] and occupancy re-ordering as in [11] we keep the occupancy code encoder as simple and fast (See III.B for all details). In particular we set the level of decodable LoD's to two, thus reducing overhead.

`

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 final LoD's. 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 fast way suitable for real-time 3D tele-immersion.

Inter Predictive Frame Coding (5): We introduce a method for inter-predictive encoding of frames based on previous inputs that 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 compute a rigid based transform based on the iterative closest point algorithm or not. If this is the case and the iterative closes point algorithm is successful (converges) the computed rigid transformation can be used as a predictor. The rigid transform is the stored compactly in a quaternion and translation quantization schema **(6).** This prediction scheme can typically save upto 30% bit-rate. This is important to reduce the data volume at high capture rates.

Coder Control (8) and Header Formatting (7): The coder uses pre-specified 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 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.

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 it is 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 with 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_min – bb_exp, y_min – bb_exp, z_min – bb_exp) to upper corner (x_max+ bb_exp, y_max+ bb_exp,z_max+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, the P is normalized on BB_IE. BB_IE is subsequently used as the octree root and the 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 the cloud P is normalized on BB_PE. In this case, the cloud P is intra coded instead, as our predictive coding algorithm only works when the bounding boxes are aligned.

The bounding box operation is an important pre-processing step to enable efficient intra and inter-coding schemes. We assume that point clouds represent segmented objects reconstructed from multi-camera 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 alignment scheme, we have implemented filters to pre-process 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.

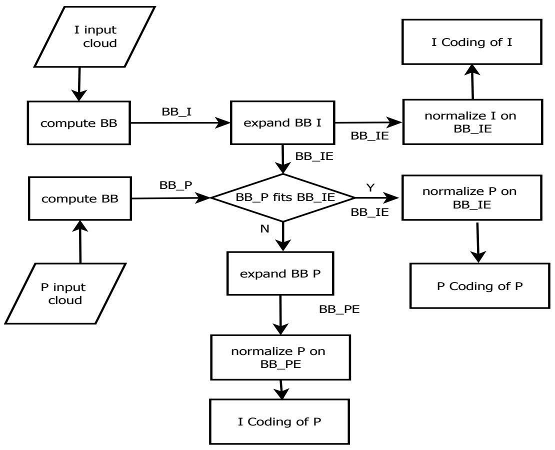


Fig 5.Bounding box alignment scheme

B. *Octree Subdivision and Occupancy Coding*

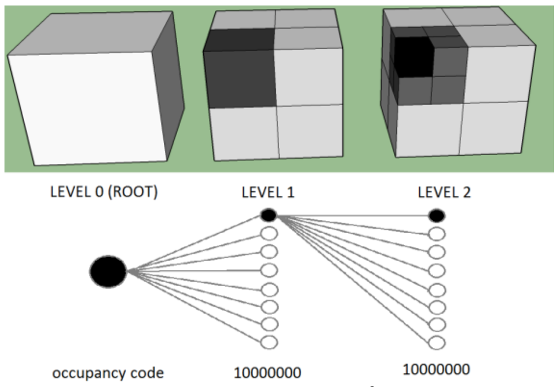


Fig 6.Octree Composition of Space

`

Starting from the root node, each octree subdivision results in an 8 bit occupancy code as shown in Fig 6. Only non-empty voxels are sub divided 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 in point cloud codecs such [13] [12]. 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 8 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 as taken in [3] (based on a carry less byte based range coder), which we applied to the different decodable LoD's. In our experiments this gave better results to compression of all LoD's at once

While previous work on progressive static point cloud compression enabled many decodable LoD's for interactive rendering, we limited the number of decodable LoD's in our implementation to 2 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 DPCM [12], colorization (keeping a frequency table of colors) [13] and based on octree structures have been proposed [12]. However, these methods are all characterized by low compression efficiency as they cannot exploit correlation between multiple co-located points. An attempt to improve this was made by Zhang et al in [6], by modelling subsets of points as a weighted graph that can be directly derived from the octree. The color attributes are then modelled as a signal on this graph. The signal values are then projected on each of the eigen vectors, resulting in spectral representation. This method imposes a large compuational overhead in computing the eigen values and vectors. Therefore, instead we introduce a method based on existing legacy JPEG codec that exploits correlation present in the point cloud recontructed 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 the colors based on a zig-zag pattern. As shown in Fig . the colors are written to 8x8 blocks based on the depth first octree taversal. The grid entries in Fig . 7 correspond to the order of the octree traversals, while the grids themselves correspond to pixels in the pre-allocated image grid. So based on the depth first octree traversal pixels are written to an image grid. This grid is re-allocated 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 co-located and therefore correlated. The DCT transform in the JPEG codec later exploits this from the mapping to 8x8 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.

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

IV. INTER FRAME CODER

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

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 coded that contains the vertices that could not be predictively coded and a *P coded part (**C_p**)* with

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

`

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 subsection III.A). The Macroblocks ***M_i*** are (1) generated at level K above the final LoD of the octree *O* that was generated coding the intra frame in the previous iteration. We chose K=4 resulting in macroblocks of 16x16x16. While K=3 resulting in 8x8x8 or K=5 resulting in 32x32x32 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*** (that are non-empy 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 that 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 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 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 to 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 only perform inter prediction 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 iterative closest point algorithm, which only takes 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 low variance corresponding macroblocks we can then finally compute a motion vector computed 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 4 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 ***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 the rigid transformation **T** is as follows. It is first composed as a rotation matrix **R**, and a translation vector **t**. The rotation matrix **R** is converted to a quaternion **q_or(s,t,u,v)** and quantized using a quaternion quantization scheme of only 3 numbers Quat1, Quat2, Quat3 (16 bits per number). The translation vector **t** is quantized using 16 bits per component (T1,T2,T3). Further details on the scheme for rigid transform coding resulting from ICP are presented in the next subsection IV.B.

Last, a color offset is optionally coded to compensate for color difference offsets between the corresponding macroblock. This color offset can optionally be used to compensate fixed color offsets between blocks due to changes in lighting that have resulted in a brightness difference. The position in ***M_p*** is stored as a key (x,y,z) *k* using 16 bit integer values for each component and corresponds to a macroblock in ***M_i*** (*k*).

This key can be used to decode the predicted blocks directly from the previously decoded octree frame ***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 bytes, and the data structure consists of 18-21 bytes (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 ***M_p_i*** are stored in a single point cloud that is coded using the method developed in section III

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		

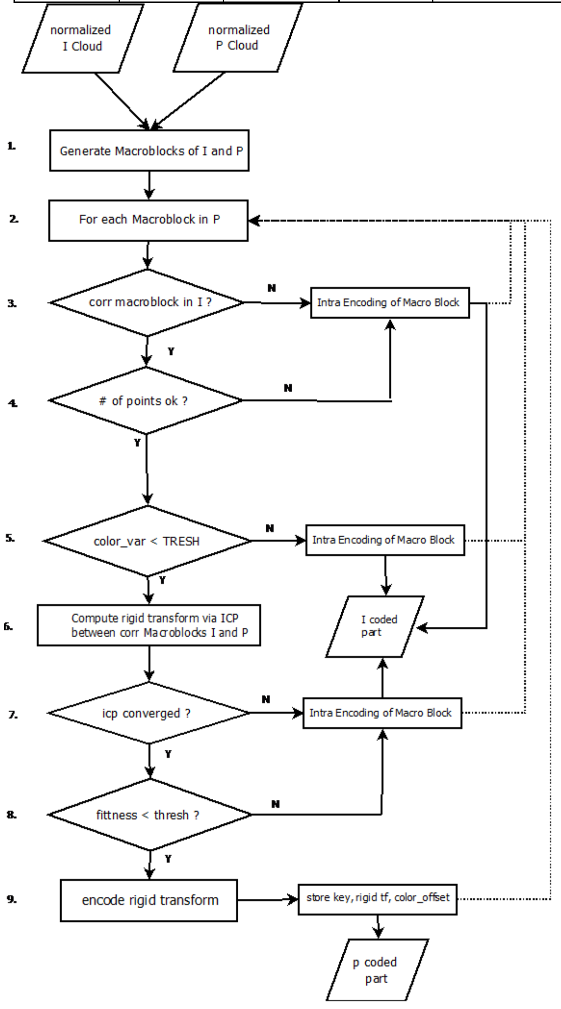


Fig. 8 Inter Predictive Point Cloud Coding Algorithm

`

B. *Rigid Transform Coding*

The 4 by 4 rigid transform matrix **T** resulting from ICP based prediction is composed into a (unitary) 3 by 3 rotation matrix **R** and a translation 3 by 1 translation vector **t**.

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

The matrix rotation $\boldsymbol{R}$ is more compactly represented as a 4 element quaternion (see Appendix for conversion formulae).

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

The quaternion is not only more compact, but also makes the rotation more susceptible to round off errors that are introduced in 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. Based on [23], we chose the largest component of $\boldsymbol{q}$ to be coded implicitly. In addition, the sign of this element needs 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 $\left[0,1/\sqrt{2}\right]$ 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 3 elements that we quantize using 16 bits (including 1 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 linear independent two 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 the rotation in all cases. The vector $\boldsymbol{t}$ is quantized using 16 bits 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 ***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 GPU. Most importantly, the ICP prediction can be parallelized, as the ICP prediction on each macroblock in ***M_s*** can happen independently. Even more, 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 Open MP for multi core processor Intel architectures.

The I coded (***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 ***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, for our work we focus parallelization of the ICP prediction, which distinguishes our codec and uses a large fraction of the computation time aswell.

`

V. Experimental Results

A. *Datasets, Experimental setup*

The hardware used in the experiments are 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. The datasets used for the evaluation (http://vcl.iti.gr/reconstruction/) have been used, which are realistic tele-immersive reconstructed data based on [2] and [1]. The software was implemented based on MS VC++ 2010 on windows platform based on algorithms available in [4] and using gcc on the Linux operating system. Note that we compare to the available real time point cloud codec in [3] which uses an octree schema and DPCM color coding and was used as base for our software implementation

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 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 both position and color information):

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

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

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

The full reference quality metric Q_{point_cloud} is computed by comparing V_{or} and V_{deg} as follows. Instead of distance to surface we take the distance to the most nearby point in V_{deg}, v_{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 of the original cloud to the most nearby point in the degraded cloud v_{nn_deg} is used to compute the PSNR per YUV component (psnr_y psnr_u etc) (7). The v_{nn_deg} is efficiently computed via a K-d tree in L2 distance norm based on algorithms available in [4].

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

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

$$psnr_{geom}=10log_{10}(||max_{x,y,z}(V_{deg})||_2^2/(d_{sym\,rms}(V_{or},V_{deg}))^2) \qquad (5)$$

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

$$psnr_y=10log_{10}(||max_y(V_{deg})||_2^2/(d_y(V_{or},V_{deg}))^2) \qquad (7)$$

Additional metrics for the quality assessment include partial bit rates 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 evaluation of time varying point cloud

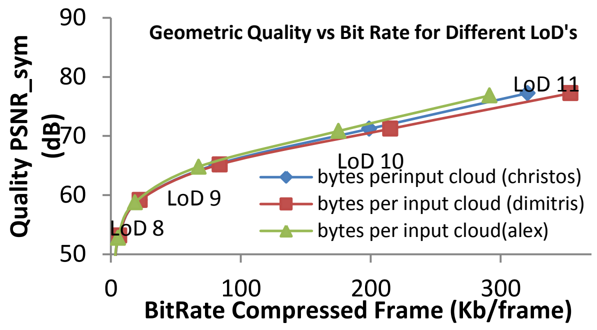


Fig. 9 Level of detail BitRate vs Quality

compression in 3D tele-immersive virtual room we will also perform subjective tests with real users.

C. Intra Coding Performance

To illustrate the effect of Level of Detail decoding the bit stream, we show the level of detail (LoD), bit rate and quality in Fig 9.From our experience with the given datasets 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 against the bitrate in Kilobytes per frame. In Fig 10 we show the results of the bitrate of the colors per output point for different LoD's. For the DPCM quantization of 4,5,6 and 7 bits per color component was set. For the JPEG color coding scheme, 75 and 80 have been used for the JPEG quality parameter. For higher LoD's 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, bit-rate is upto 10 times lower compared to DPCM based coding. Such configuration is very useful for the targeted application of 3D Tele-immersion, where low bit rates are needed. In addition, the difference in objective quality does not necessarily correspond well to the subjective quality. In V.E we will show in the subjective studies that the color quality degradation introduced by the color coding method is negligible even compared to 8-bits 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 formula (7) for the Y, U and V Components for each of the different LoDs (7-11). As for higher LoD's, the quality increases (more points) and the bytes per output point decreases, therefore, curves show higher quality for lower bitrates. The JPEG based scheme gives qualities compared to 4, 5 and sometimes even 6 bit DPCM at up to 10 times less bytes per output point. This is the main advantage gained by the color coding methods, i.e. low bit rate coding of the color attributes. As the method re-uses a fast optimized JPEG implementation, we do not introduce any extra computation time compared to the DPCM based methods (i.e. the encoding speed and decoding speed have been similar for both methods), which is an advantage compared to methods such as [6]. For the U and V components the achieved quality with JPEG is slightly lower compared to the Y component.

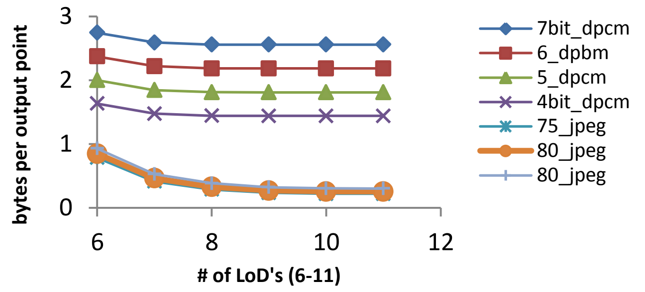


Fig. 10 Bytes per point for color coding per LoD

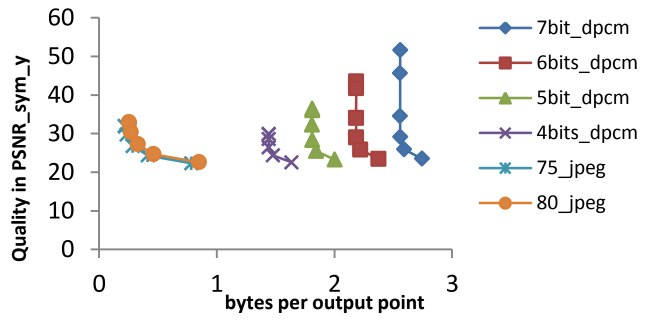


Fig. 11 Rate Distortion for color coding (Y component)

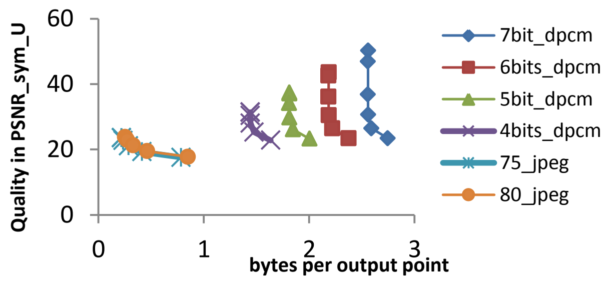


Fig. 12 Rate Distortion for color coding (U component)

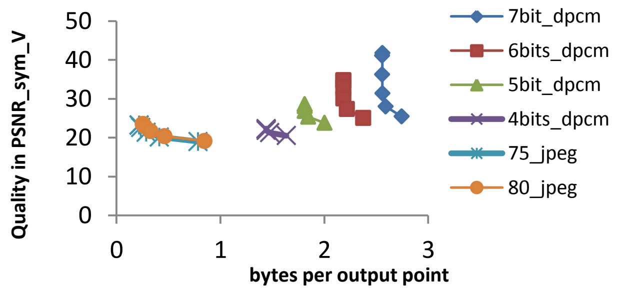


Fig. 13 Rate Distortion for color coding (V component)

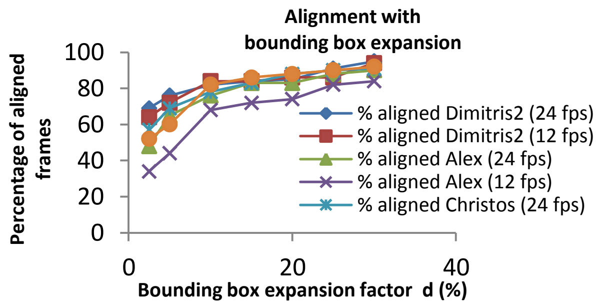


Fig. 14 Bounding box alignment

`

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2016.2543039, IEEE Transactions on Circuits and Systems for Video Technology

D. *Inter Predictive Coding Performance*

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

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 (16x16x16 blocks), (d=20%) and the ICP convergence percentage. For all tested datasets (based on capturing at 12 or 24 fps) over 60% of macroblocks are shared in aligned frames, and approximately 30% are both shared and converging in the ICP stage (these are used for prediction). The percentage that is shared and converged is shown in red, and relates largely to the bitrate savings (which is over 30% for the Dimitrios dataset that has the largest percentage of shared blocks). In Fig 16 we compare the compressed size of intra coded frames with predictive frames, for the Alex dataset the size was reduced by 20.5 percent compared to 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 and intra coded frames is shown in Fig 17 based on the metric (5). The loss in geometric PSNR in predictively encoded frames is about 10 dB in each of the tested datasets. We will explore the subjective effect of this degradation in the next section. In Figs 18 to 21 we show the results for both predictive and JPEG.

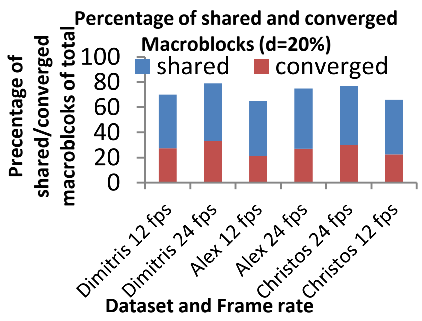


Fig. 15 shared macroblock and convergence percentage

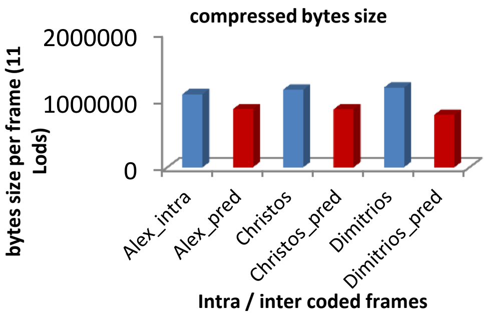


Fig. 16 frame size predicted vs intra

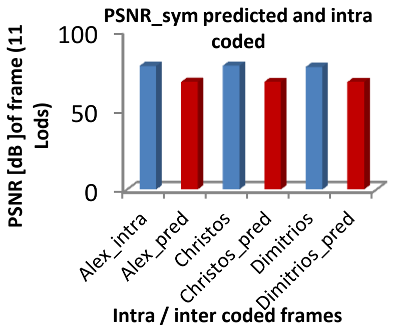


Fig. 17 Quality predicted frames vs intra frames

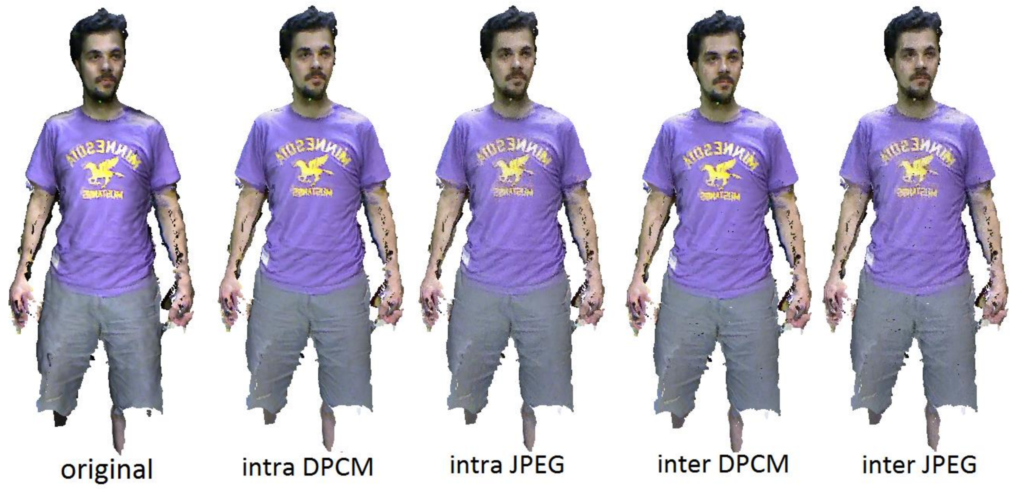


Fig. 18 Results at LoD Christos 11 (11 bit octree)

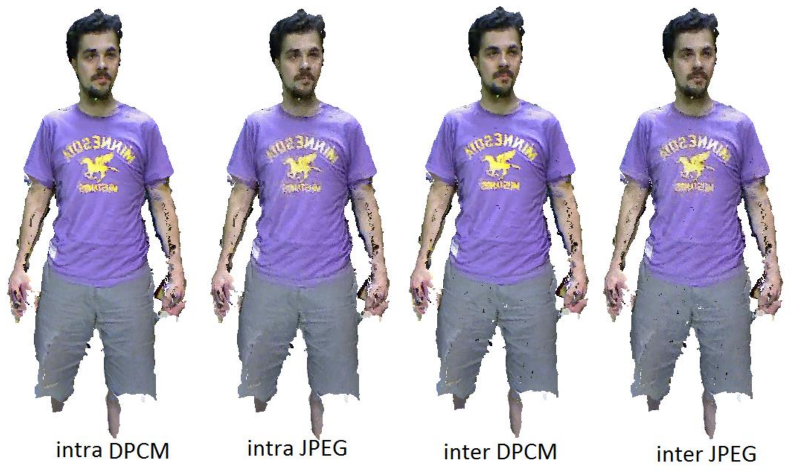


Fig. 19 Results at LoD Christos 10 (10 bit octree)

`

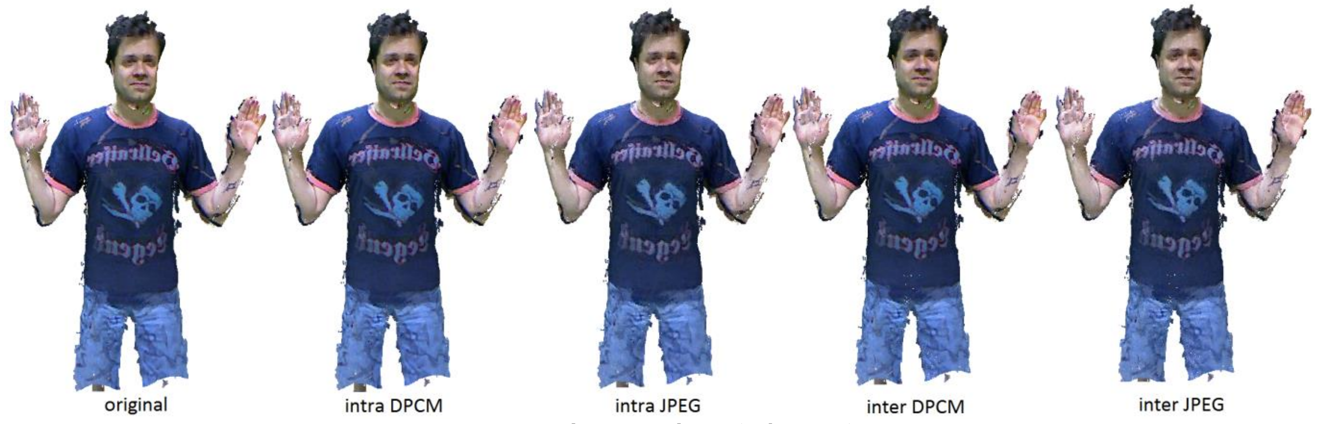


Fig. 20 Results at LoD Alex 11 (11 bit octree)

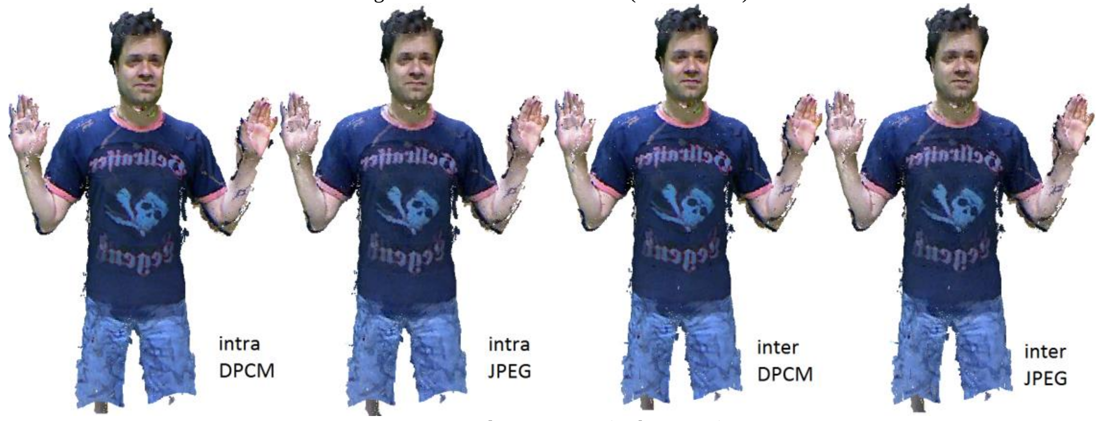


Fig. 21 Results at LoD 10 (10 bit octree)

encoded/decoded frames and the original for the Christos and Alex datasets. All point clouds have been rendered under the exact same conditions (lighting, camera position) using MeshLab Software[2]. For the LoD 11and Lod 10 prediction artifacts are not visible. For the data under test, a good convergence and shared block percentage was achieved for LoD 11 (11 bit octree setting, i.e. 11 quantization bits per direction). For different datasets, some tuning could be necessary to find the best setting. The software implementation of codec can be configured via a parameter_config text file where the codec can be configured.

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 where 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, rotate) and look around using 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 users cannot navigate through any of the occupied 3D space, point cloud, avatar or other world objects.

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

20 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 signed the consent form for using this information for scientific purposes. The group consisted of 4 women and 16 men, in age range from 24 to 62. All were working in Information Technology (electronic engineering, computer science or mathematics). Nationalities were mixed (10 Dutch, 3 Spanish, 2 Italian, 2 Chinese, 2 German, 1 English), education levels above B.Sc and the participants have been exposed to different versions of the point clouds at LoD 11 shown in Fig 20 and Fig18 (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 datasets are rendered at 23 fps when presented to the user (this corresponds to the capture rate). Each of the users have been exposed to the three codecs (DPCM, JPEG, Inter predicted) conditions on two different datasets (Alex, Christos) in a random order, resulting in a total of six test conditions. The controlled setup consisted of a 47 inch LG TV (model num. 47LB670V) screen positioned at 1.5 meters from the users running the Reverie system, shown in Fig 25, and a laptop + mouse to control the avatar. After each test condition the users have been requested to fill the questionnaire on 8 different quality aspects on a 1-5 scale ranging from Bad to Excellent based mean opinion score scale (MOS) (Bad=1, poor=2, fair=3,good=4, excellent=5). These aspects include the 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” as

`

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2016.2543039, IEEE Transactions on Circuits and Systems for Video Technology

being in a room with a real user comparing the avatar and the point cloud representations. The results in Figs. 24 to 29 can be interpreted as follows. The values 1-5 correspond to the Mean Opinion Scores of the results for each quality aspect. The error bars represent the standard deviation around the mean opion score. Due to the usage 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 a 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 dataset the color coding and prediction do not introduce a significant degradation in quality, while for Christos some difference is observed (but not signficant). The assessment of color quality

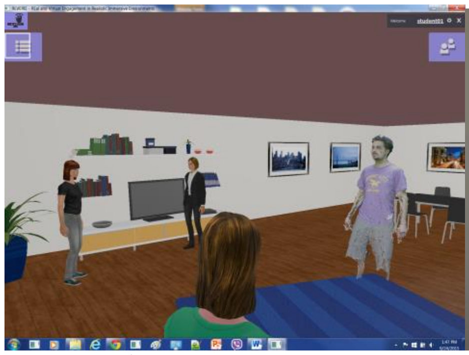
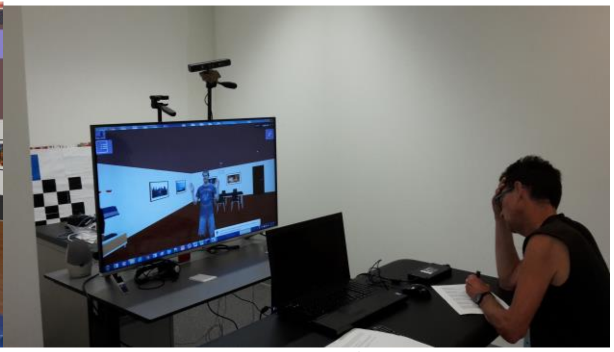


Fig. 22 Screenshot of Point Cloud Rendering in Virtual World.


—Fig. 23. Test setup with user filling in the questionaire

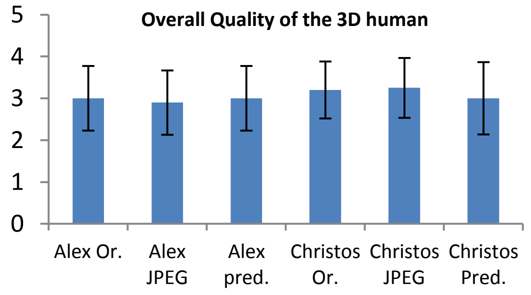


Fig. 24 Subjective Results, perceived quality of the 3D Human

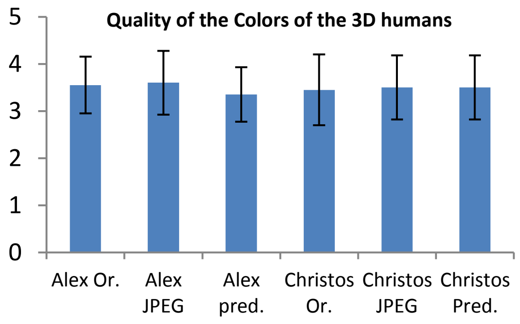


Fig. 25 Subjective Results, colors of the 3D Human

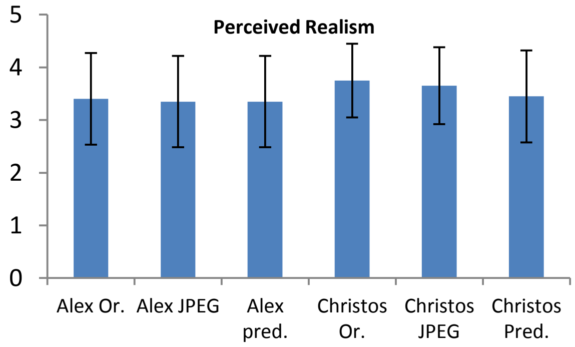


Fig. 26 Subjective Results, Quality of the motion

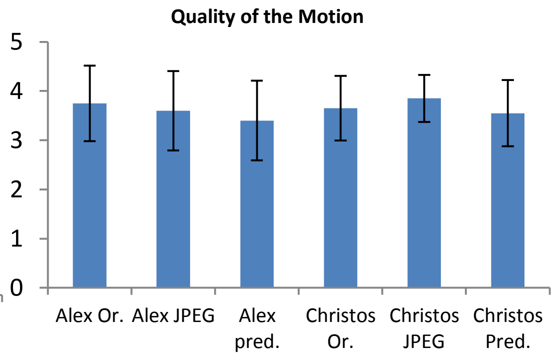
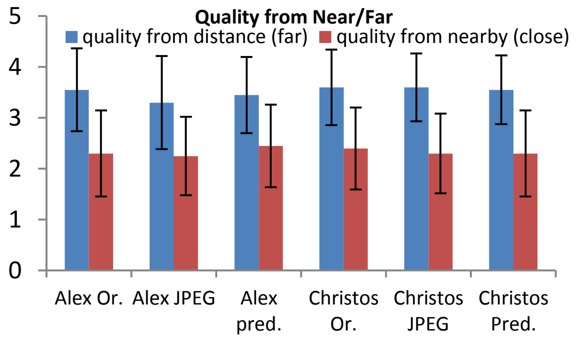
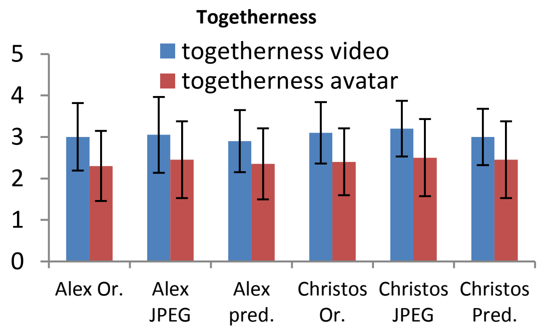


Fig. 27 Subjective Results, perceived realism





`

Fig. 28 Subjective Results Quality, near vs far

Fig. 29 Feeling of togetherness, avatar user versus 3D Video user

in Fig 25 compares the 8 bit DPCM coding to the JPEG coding and inter prediction. Surprisingly, the JPEG color coding method does not introduce significant subjective distortion, as this method codes at upto 10 times lower bit rates compared to the 8 bit DPCM color data. The results on perceived realism and the quality of the motion are shown in Fig.26 and Fig. 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 the point cloud quality is seen as low quality while from distance it is between fair and good. This is mainly due that the rendering of the point clouds from nearby allows you to see through the points, while from a distance this is not the case. Perhaps looking into 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 to a simple computer avatar in terms of "feeling together", which is the ultimate aim of tele-precence technologies. In this respect the point cloud representation scored significantly better than the computer avatar (Fig. 29).

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 LoD's for the intra coder and near real time (<1s) for the inter-predictive encoder.

All tests have used the Dimitrios2-Zippering dataset running on 3 intel based computers, one based on i7 2.8 GHz and win7, one Xeon 3.7 GHz running win7 (both compiled using VC++ 2010) and one i7 3.20 GHz running Ubuntu Linux (gcc compiler suite). The real-time results for intra encoding for different LoD's 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 multi core intel architectures (we measured up to 20 % improvement on the windows platforms). This hints in the direction that 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 LoD's plotting against the number of cores/threads used to compute the ICP.

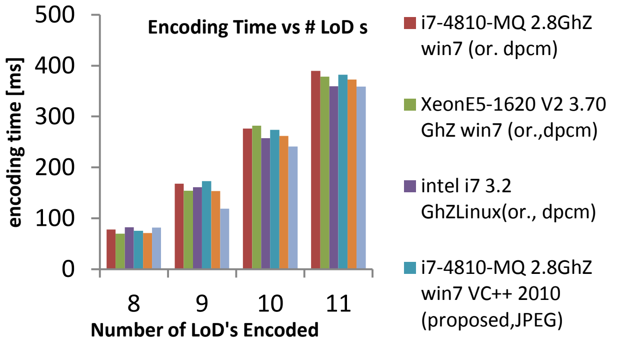


Fig. 30 Real-Time Performance of intra coding for different configuration

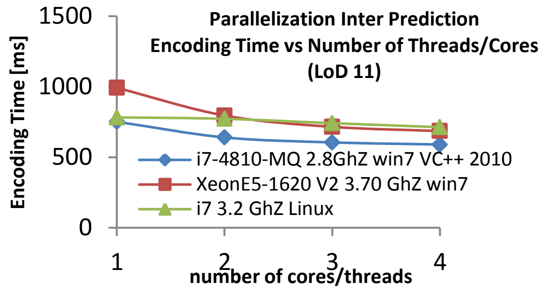


Fig. 31 Real-Time Performance of intra coding for different configuration

VI. Conclusions and Discussion

In this paper we introduce 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 that 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 a 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 to 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.

VII. ACKNOWLEDGEMENT

We thank all partners in the Reverie FP7 consortium for contributing to the reverie framework source code and their expertise. In particular we thank Dimitris Alexiadis and Petros Daras at CERTH/ITI for providing the point cloud datasets. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. ICT-2011-7-287723 (REVERIE project).

VIII. References

[1] A. Doumanoglou, D. Alexiadis, D. Zarpalas, P. Daras , "Towards Real-Time and Efficient Compression of Human Time-Varying-Meshes," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 12, December 2014.

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

`

[2] D. Alexiadis, D. Zarpalas, and P. Daras.,., "Real-Time, full 3-D reconstruction of moving foreground objects from multiple consumer depth cameras," *IEEE Transactions on Multimedia*, vol. 15, pp. 339-358, 2013.

[3] J. Kammerl. , N. Blodow, R.B. Rusu, S. Gedikli, and E. Steinbach M Beetz, "Real-time compression of point cloud streams," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on , pp.778,785, 14-18 May*, 2012.

[4] R.B. Rusu, "3D is here: Point Cloud Library," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, Shanghai, 2011.

[5] Dorina Thanou, Philip A. Chou, and Pascal Frossard, "Graph-based compression of dynamic 3D point cloud sequences," in *Cornell Library, arxiv, Computer Vision and Pattern Recognition*, 2015.

[6] D.Florencio, C. Loop C. Zhang, "Point Cloud Attribute Coding with the Graph transform," in *IEEE International Conference on Image Processing*, Paris, October 2014.

[7] T. Wiegand, and G. J. Sullivan. A. Vetro, "Overview of the stereo and multiview video coding extensions of the H. 264/MPEG-4 AVC standard," *Proceedings of the IEEE*, pp. 626-642, april 2011.

[8] D. Marpe, C. Bartnik, S. Bosse, H. Brust, T Hinz, H. Lakshman, P. Merkle, F.H. Rhee, G. Tech and M. Winken, and T. Wiegand K. Muller H. Schwarz, "3D High-Efficiency Video Coding for Multi-View Video and Depth Data," *IEEE Transactions on Image Processing*, vol. 22, no. 9, pp. 3366-3378, 2013.

[9] REVERIE FP7. (2015, Aug.) reverie fp7. [Online]. http://www.reveriefp7.eu/

[10] 3DLife ACMMM grand challenge. (2015) 3D Human reconstruction and action recognition from multiple active and passive sensors. [Online]. mmv.eecs.qmul.ac.uk/mmgc2013/

[11] Y. Huang, J. Peng, C. Kuo, C. Gopi , "A generic scheme for progressive point cloud coding," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 2, pp. 440-453, 2008.

[12] R. Schnabel, and R. Klein , "Octree-based Point-Cloud Compression," in *SPBG*, 2006, pp. 111-120.

[13] J. Peng, C.S. Kim, C.-C. Jay Kuo , "Technologies for 3D mesh compression: A survey," *Journal of Visual Communication and Image Representation*, vol. 16, no. 6, pp. 688-733, December 2005.

[14] R. Mekuria. and D. Bulterman P. Cesar., "Low complexity connectivity driven dynamic geometry compression for 3D Tele-Immersion," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, Florence, 2014, pp. 6162,6166.

[15] J.Hou, C. Lap-Pui and N. Magnenat-Thalmann Y He, "A novel compression framework for 3D time-varying meshes," in *IEEE International Symposium onCircuits and Systems (ISCAS), 2014*, Melbourne, 2014, pp. pp.2161,2164.

[16] D. Alexiadis, S. Asteriadis, D. Zarpalas, P. Daras A. Doumanoglou, "On human time-varying mesh compression exploiting activity-related characteristics," in *IEEE ICASSP*, Florence, 2014.

[17] R. Mekuria, P. Cesar , "A Basic Geometry Driven Mesh Coding Scheme with Surface Simplification for 3DTI.," *MULTIMEDIA COMMUNICATIONS TECHNICAL COMMITTEE e-letter*, vol. 9, no. 3, pp. 6-7, May 2014.

[18] F. Prêteux., T. Zaharia K.Mamou., "TFAN: A low complexity 3D Mesh Compression Algorithm," *Computer Animation and Virtual Worlds*, vol. 20, no. 12, pp. 343-354, June 2009.

[19] K. Mamou. , T. Zaharia, and F. Preteux, "FAMC: The MPEG-4 standard for Animated Mesh Compression," in *15th IEEE International Conference on Image Processing, 2008. ICIP 2008.*, San Diego, 2008, pp. pp.2676,2679.

[20] Ahsan Arefin, Gregorij Kurillo, Pooja Agarwal, Klara Nahrstedt, and Ruzena Bajcsy Wanmin Wu, "Color-plus-depth level-of-detail in 3D tele-immersive video: a psychophysical approach.," in *Proceedings of the 19th ACM international conference on Multimedia (MM '11)*, 2011, pp. 13-22.

[21] O. Schreer P. Kauff, "An immersive 3D video-conferencing system using shared virtual team user environments," in *Proceedings of the 4th international conference on Collaborative virtual environments (CVE'02)*, 2002, pp. 105-112.

[22] J. Ostermann , "N14662 MPEG-4 requirements (Sapporo Revision)," MPEG, Sapporo, JCT1/SC29/WG11 output document 2014.

[23] Z. Adami, "Quaternion Compression," in *Game Programming Gems*, Dante Treglia, Ed.: Charles River Media, 2002, ch. 2.4., pp. 187-191.

[24] S. Aluru B. Hariharan, "Efficient parallel algorithms and software for compressed octrees with applications to hierarchical methods," *Parallel Computing*, vol. 31, no. 4, pp. 311-331, March-April 2005.

[25] L. Bivolarski, R. Mekuria, M. Preda. , "N15578 Preliminary guidelines for evaluation of Point Cloud compression technologies," ISO/IEC JCT1 SC29 WG11 (MPEG), geneva, mpeg output document MPEG-4, 2015.

Rufael Mekuria received the B.Sc and M.Sc. in Electrical Engineering from Delft University of Technology, Delft. In September 2011 he joined the Centrum Wiskunde & Informatica as a researcher. He was involved in the FP7 Project Reverie and he is active in international standardization activities in ISO IEC Motion Picture Experts Group (MPEG) since April 2013. His research interest are in 3D Tele-immersion, 3D Compression of point clouds and meshes, 3D network streaming and protocols and multimedia systems. He has published in leading conferences and journals of the IEEE / ACM / SPIE in the area of image processing and multimedia systems (including 4 invited and 1 distinguished paper). He is currently pursuing the PhD degree at VU University.

Kees Blom works as a scientific programmer at CWI since 1983. He has worked on implementations of graphics standards (GKS), languages for parallel and distributed systems and interactive multimedia systems (SMIL). He is currently involved in the implementation of point cloud codecs.

Pablo Cesar Pablo Cesar received his PhD from Helsinki University of Technology (2005). He leads the Distributed and Interactive Systems group at Centrum Wiskunde & Informatica: CWI. 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. He is involved in standardization activities (e.g., W3C, MPEG, ITU) and has been active in a number of European projects. He has given tutorials about multimedia systems in prestigious conferences such as ACM Multimedia and the WWW conference.

IX. APPENDIX

A 3 by 3 rotation matrix R consisting of elements r_{ij} can be converted to a quaternion **q**(s,t,u,v) with s the real and t,u,v the imaginary parts as follows:

$$s = \frac{1}{2}\sqrt{1 + r_{11} + r_{22} + r_{33}} \quad , \; t = \frac{1}{4s}(r_{32} - r_{23}) \, ,$$

$$u = \frac{1}{4s}(r_{13} - r_{31}) \, , \qquad v = \frac{1}{4s}(r_{21} - r_{12})$$

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

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

`