Motion-Compensated Compression of Dynamic Voxelized Point Clouds

Ricardo L. de Queiroz, *Fellow, IEEE*, and Philip A. Chou, *Fellow, IEEE*

***Abstract*— Dynamic point clouds are a potential new frontier in visual communication systems. A few articles have addressed the compression of point clouds, but very few references exist on exploring temporal redundancies. This paper presents a novel motion-compensated approach to encoding dynamic voxelized point clouds at low bit rates. A simple coder breaks the voxelized point cloud at each frame into blocks of voxels. Each block is either encoded in intra-frame mode or is replaced by a motion-compensated version of a block in the previous frame. The decision is optimized in a rate-distortion sense. In this way, both the geometry and the color are encoded with distortion, allowing for reduced bit-rates. In-loop filtering is employed to minimize compression artifacts caused by distortion in the geometry information. Simulations reveal that this simple motion-compensated coder can efficiently extend the compression range of dynamic voxelized point clouds to rates below what intra-frame coding alone can accommodate, trading rate for geometry accuracy.**

***Index Terms*— Telepresence, dynamic point clouds, point cloud compression, data compression, motion compensated video.**

I. Introduction

With the emergence of inexpensive consumer electronic systems for both 3D capture and 3D rendering, visual communication is on the threshold of advancing beyond traditional 2D video to immersive 3D communication systems [1]–[8]. Dynamic 3D scene capture can be implemented using color plus depth (RGBD) cameras, while 3D visualization can be implemented using stereoscopic monitors or near-eye displays to render the subject within a virtual or augmented reality. The processing for capture and display can be done in real time using powerful graphics processing units (GPUs) [9]. However, representing a complex, dynamic 3D scene generates a large amount of data. Compression is therefore an essential part of enabling these emerging immersive 3D systems for communication.

There are many choices for representing 3D data, and the most appropriate choice depends on the situation. For example, *dense voxel arrays* may be best for representing dense volumetric medical data, while *polygonal meshes* may be good for representing surfaces of 3D objects typically found in computer graphics. *Point clouds* are well-suited to sampled real world objects for which the data are volumetrically sparse, especially if the topology is not necessarily a 2D manifold. An alternative to point clouds are *sparse voxel arrays*, or *voxel clouds*, which are arbitrary collections of voxels. Unlike points, voxels have a volumetric aspect, which may be important in some scenarios. Point clouds and sparse voxel arrays obviate some of the common problems that 2D manifolds have, such as dealing with boundary conditions on cut edges, and topological changes over time. Yet another possible representation of 3D data is simply a set of color and depth maps, sometimes called multiview video plus depth (MVD) [10]. This is a low-level representation close to the RGBD sensors. Closely related to color and depth maps are elevation maps [11] and multi-level surface maps [12].

In our work, the 3D representation of choice is sparse voxel arrays, which we call *voxelized point clouds* in this paper. Neglecting the volumetric aspect of voxels, voxelized point clouds can be considered simply as point clouds whose points are restricted to lie on a regular 3D grid or lattice. As mentioned above, for the kinds of data expected in 3D scene capture, voxelized point clouds are a more natural fit than dense voxels arrays, and they obviate the kinds of problems that polygonal meshes have with sampled data. Compared to color and depth maps, voxelized point clouds are a higher level representation, in that redundancies and inconsistencies between overlapping sensor maps have already been removed in a multi-camera sensor fusion step. Compared to arbitrary point clouds, voxelized point clouds have implementation advantages and are highly efficient for real-time processing of captured 3D data [9].

Each representation employs its own compression techniques, such as [13]–[23] for polygonal meshes, and [10], [24] for multiview video plus depth. Previous approaches to compressing point clouds include [25]–[31]. The most efficient of these are based on voxelization. We believe [28] and [29],

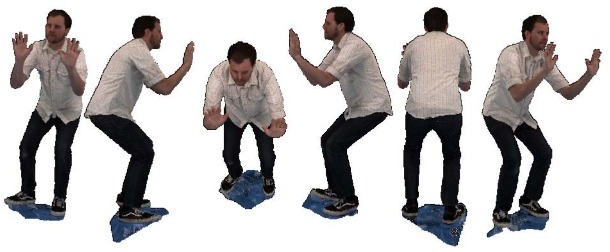


Fig. 1. Different viewpoints of a 3D point cloud frame.

Manuscript received April 12, 2016; revised April 16, 2017; accepted May 17, 2017. Date of publication May 24, 2017; date of current version June 13, 2017. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Ce Zhu. *(Corresponding author: Ricardo L. de Queiroz.)*

R. L. de Queiroz is with the Computer Science Department, Universidade de Brasilia, Brasilia 70910-900, Brazil (e-mail: queiroz@ieee.org).

P. A. Chou was with Microsoft Research, Redmond, WA USA. He is now with 8i Labs., Bellevue, WA 98004 USA (e-mail: pachou@ieee.org).

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/TIP.2017.2707807

1057-7149 © 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Authorized licensed use limited to: Ricardo De Queiroz. Downloaded on May 04,2020 at 17:43:43 UTC from IEEE Xplore. Restrictions apply.

until recently, represented the state-of-the-art in (voxelized) point cloud color compression, with the former focusing on intra-frame color compression and the latter extending that work to inter-frame color compression. Both use the same codec, applied respectively to the colors directly or to their prediction residuals, based on an orthogonal graph transform and arithmetic coding of carefully modeled coefficients. The graph transform is a natural choice for the spatial transform of the color signal due to the irregular domain of definition of the signal. Unfortunately, the graph transform requires repeated eigen-decompositions of many and/or large graph Laplacians, rendering the approach infeasible for real-time processing. We have recently submitted a work on a coder that is able to match or outperform existing intra-frame color compression methods at a reduced cost [32]. Such a still-frame coder is based on a region-adaptive hierarchical transform (RAHT) specially developed for point clouds and is used as a fundamental building block in the present framework for our dynamic point cloud coder, which can be considered a 3D video coder.

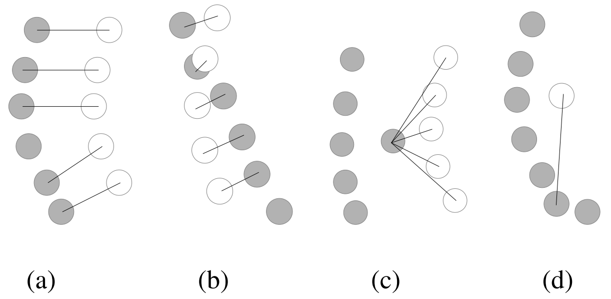


Fig. 2. Correspondences. Gray circles illustrate points (or voxels) in $\mathcal{F}(t)$, while the white ones represent points or voxels in $\mathcal{F}(t+1)$. (a) Even for a simple linear motion, an Euclidean-distance criterion may lead to a non-uniform correspondence field. (b) Voxels in consecutive frames are typically misaligned. (c) A stray voxel, for example created by noise, can act as a "lightning rod" atractting all local correspondence. (d) Correspondence based on color can lead to unpredictable fields, since associations can be made to any voxel far away as long as it has a similar color.

II. Voxelized Point Clouds

We represent 3D data by voxelized point clouds. A point cloud is a set of points $\{v\}$, each point v having a spatial position (x, y, z) and a vector of attributes such as colors, normals, or curvature. In this paper we assume the attributes are colors represented as (Y, U, V) tuples in YUV color space. A point cloud may be *voxelized* by quantizing the point positions to a regular lattice. A quantization cell, or *voxel*, is said to be *occupied* if it contains a point in the point cloud and is *unoccupied* otherwise. (A polygonal mesh may be likewise voxelized, in which case a voxel is occupied if and only if it contains a point of the surface.) An occupied voxel derives its color from the color(s) of the point(s) within the voxel, possibly by averaging, but this is outside the scope of this paper. We assume simply that each occupied voxel has a color. Without loss of generality, we may assume that the voxels are addressed by positions in the integer lattice $\mathbb{Z}_W^3$, where $\mathbb{Z}_W = \{0, \dots, W-1\}$, $W = 2^D$ is its width, and D is an integer. Thus, we take x, y, and z to be D-bit unsigned integers. These are analogous to row and column indices in ordinary 2D image processing. Similarly, we take Y, U, and V to be 8-bit unsigned integers. Thus, our voxelized point cloud is a finite set or arbitrarily indexed list of occupied voxels $\{v_i\}$ in which each voxel

$$v_i = [x_i, y_i, z_i, Y_i, U_i, V_i] \tag{1}$$

comprises a unique integer spatial location (x_i, y_i, z_i) and an integer color vector (Y_i, U_i, V_i).

We are mostly interested in live video and dynamic point clouds. Thus at every discrete time t we have the frame $\mathcal{F}(t) = \{v_{it}\}$, which is represented as a list of voxels

$$v_{it} = [x_{it}, y_{it}, z_{it}, Y_{it}, U_{it}, V_{it}]. \tag{2}$$

Note that the list is unordered and can be shuffled with no loss in representation, such that there is no direct relationship between $v_{i,t}$ and $v_{i,t+1}$, i.e. between voxels in the same list position, but at different frames. Moreover, different frames often have different numbers of occupied voxels.

III. Temporal Voxel Correspondence

We assume that each voxel $v_{i,t+1} \in \mathcal{F}(t+1)$ can be assigned to a corresponding voxel $v_{jt} \in \mathcal{F}(t)$. Intuitively, one expects voxels to be shifted locally from one frame to another, due to more or less rigid translational motion, as illustrated in Fig. 2(a)(b). However, since the shifts are not exactly rigid, in general this is a many-to-one mapping from $\mathcal{F}(t+1)$ to $\mathcal{F}(t)$. This mapping may be described by a $||\mathcal{F}(t+1)|| \times ||\mathcal{F}(t)||$ matrix $\mathbf{S}(t) = \{s_{ijt}\}$ where $s_{ijt} = 1$ if the i-th voxel at frame $t+1$ corresponds to the j-th voxel in frame t, and is 0 otherwise. (Here, $||\mathcal{F}(t)||$ is the number of occupied voxels in $\mathcal{F}(t)$.) If we stack all the row vectors with the geometry information $\mathbf{G}(t) = \{[x_{jt}, y_{jt}, z_{jt}]\}$ and do the same with the colors $\mathbf{C}(t) = \{[Y_{jt}, U_{jt}, V_{jt}]\}$, the geometry and color correspondence residues may be written:

$$\mathbf{E}_g = \mathbf{G}(t+1) - \mathbf{S}(t)\mathbf{G}(t), \tag{3}$$

$$\mathbf{E}_c = \mathbf{C}(t+1) - \mathbf{S}(t)\mathbf{C}(t). \tag{4}$$

Note that the geometry residues may be considered motion vectors, while the color residues may be considered color prediction errors.

One way to find an appropriate correspondence $\mathbf{S}(t)$ is to minimize a distortion measure based on these residues, for example by minimizing

$$\delta = \alpha_g||\mathbf{E}_g||^2 + \alpha_c||\mathbf{E}_c||^2, \tag{5}$$

where α_g and α_c are multipliers indicating the relative importance of the magnitudes of the color and geometry residues.

Minimizing both geometry and color residuals is important. If only geometry were considered, then the minimization of (5) would result in the assignment to each voxel $v_{i,t+1} \in \mathcal{F}(t+1)$ the voxel $v_{jt} \in \mathcal{F}(t)$ whose position is closest in Euclidean space, often resulting in poor correspondences including "lightning rods," as illustrated in Fig. 2(c). Alternatively, if only color were considered, then the minimization of (5) would result in the assignment to each voxel $v_{i,t+1} \in \mathcal{F}(t+1)$

Authorized licensed use limited to: Ricardo De Queiroz. Downloaded on May 04,2020 at 17:43:43 UTC from IEEE Xplore. Restrictions apply.

the voxel $v_{jt} \in \mathcal{F}(t)$ whose color is closest in color space. In effect, the voxels in $\mathcal{F}(t+1)$ would simply use the voxels in $\mathcal{F}(t)$ as colors in a palette, and the correspondences would be chaotic as in Fig. 2(d).

As important as the magnitudes of the geometry and color residuals to finding good correspondences, however, is smoothness of the correspondence field. Hence many works [29], [30], [33] include a geometry residual smoothness term, e.g.,

$$\delta = \alpha_c||\mathbf{E}_c||^2 + \alpha_g||\mathbf{E}_g||^2 + \alpha_s tr(\mathbf{E}_g^T \mathbf{L}(t+1)\mathbf{E}_g), \quad (6)$$

where $\mathbf{L}(t+1)$ is a Laplacian operator, whose quadratic form is the weighted sum of squared differences between the values at each voxel and its neighbors, and the trace operator simply sums up the separate contributions for x, y, and z components. There is evidence that humans perceive correspondence in terms of smoothed motion [34]; hence the smoothness term is a good way to select correspondences that match human perception.

Recognizing that there is much prior work on determining smooth correspondence fields for 3D flow [33], [35]–[43], in this paper we leave such *motion estimation* outside the scope of the paper, and we treat such motion estimates as given. Specifically, in this paper we use correspondence fields given by [43], and focus instead on how to use the correspondences for efficient coding. Moreover, we use the correspondences to measure perceived geometric and color quantization error.

IV. Distortion Metrics and Residuals

We are interested in encoding and transmitting the voxelized point clouds of sequences of people and objects. For the purposes of visual reconstruction, the unoccupied and interior voxels do not need to be encoded, but only the external "shell" of the person or object. The sparsity of these occupied voxels allows efficient still frame or *intra-frame* compression of the geometry using octtrees [9], [31]. Multiple frame or *inter-frame* compression of the geometry can also be performed using octtrees, by octtree coding the exclusive-OR (XOR) between sets of occupied voxels in successive frames [29], [31]. However, this method can also increase the data rate when the geometric distance between corresponding voxels is typically more than a unit voxel [29]. Moreover, this method codes the geometry losslessly and hence is not optimized from a rate-distortion point of view, nor can it achieve very low bit rates. Thus, in this paper, we attempt to *predict* the geometry as well as the color, and to code the prediction residuals in a rate-distortion optimized way.

Specifically, from (3) and (4), we have

$$\mathbf{G}(t+1) = \mathbf{S}(t)\mathbf{G}(t) + \mathbf{E}_g \quad (7)$$
$$\mathbf{C}(t+1) = \mathbf{S}(t)\mathbf{C}(t) + \mathbf{E}_c, \quad (8)$$

so we can obtain $\mathbf{G}(t+1)$ and $\mathbf{C}(t+1)$ by encoding the correspondences $\mathbf{S}(t)$ and the residuals $\mathbf{E}_g$ and $\mathbf{E}_c$.

In order to evaluate the distortion, although objective metrics such as the squared error have traditionally been very useful in evaluating video coders, for evaluating dynamic point cloud coders, geometry is a complicating factor, because when the geometry changes there is no one obvious metric space

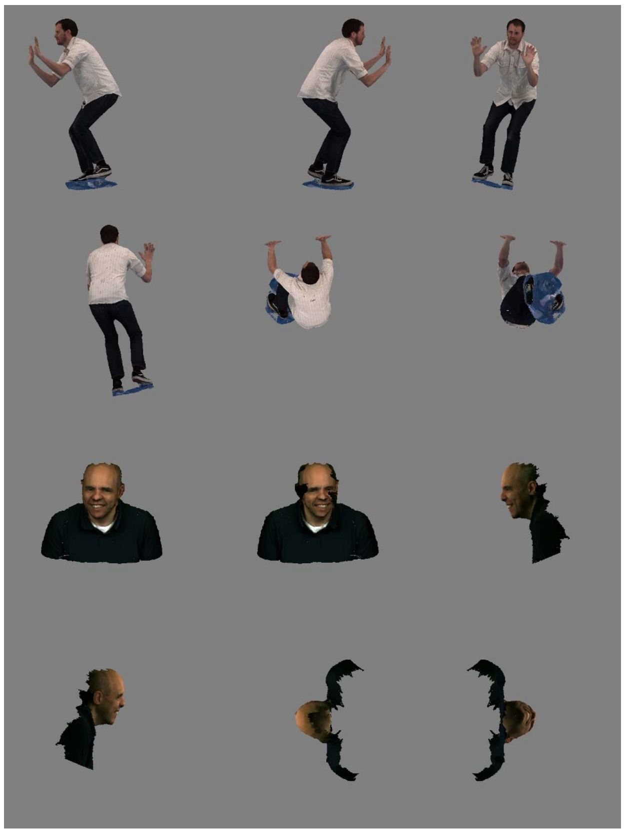


Fig. 3. Six principal projections of a frame of sequences "Man" and "Ricardo".

in which to compute either a geometric or color error, or its norm. Thus we take two approaches to computing distortion for point clouds: one based on correspondence and another based on projection.

In a correspondence-based distortion metric, we first establish a correspondence between the original frame $\mathcal{F}(t)$ and the reconstructed frame $\hat{\mathcal{F}}(t)$ using for example proximity, i.e., a voxel in $\hat{\mathcal{F}}(t)$ is associated to its spatially closest voxel in $\mathcal{F}(t)$. Once the association is made we compute the distortion using a function of $\mathbf{E}_g$ or $\mathbf{E}_c$. From all pairing associations we can compute the mean squared error (MSE) and, from it, the corresponding peak signal-to-noise ratio (PSNR) in dB. Let there be N_v occupied voxels in the current frame and let $\mathbf{E}_c$ compute only the residual of the Y color component. In particular, the MSE we use is

$$\delta_{Y+G} = \frac{1}{N_v}\left(||\mathbf{E}_c||^2 + \beta||\mathbf{E}_g||^2\right) \quad (9)$$

where β weights the mixture of geometry and color distortions. From δ_{Y+G} we compute PSNR-Y+G.

In a projection-based distortion measure, a projection view of the point cloud is rendered for a viewer. We assume an orthogonal projection of the point cloud over the six sides of a cube at the limits of the voxel space. The observer is assumed far away from the scene so that the rays from it to the voxels are parallel and the background is assumed at a mid level of gray. The frontal orthogonal projection within our 9-level test set is a 512×512-pixel image and the projections are illustrated in Fig. 3. The distortion metric between two

Authorized licensed use limited to: Ricardo De Queiroz. Downloaded on May 04,2020 at 17:43:43 UTC from IEEE Xplore. Restrictions apply.

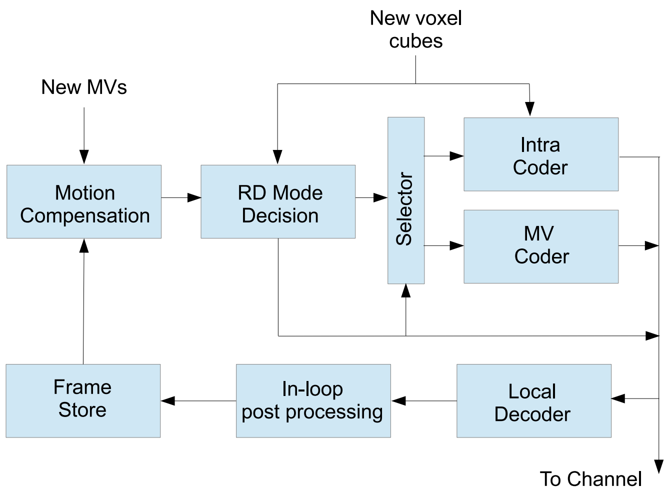


Fig. 4. Encoder diagram.

point clouds, original and reconstructed, is then the MSE δ_P between the two corresponding composite images, each with the 6 orthogonal projections of the original or reconstructed Y frames. We compute the corresponding PSNR and refer to it as the projection PSNR or PSNR-P.

In this paper, we compute both PSNR-P and PSNR-Y+G as $10\log_{10}(255^2/MSE)$, where MSE is the mean projection-based or correspondence-based error δ_P or δ_{Y+G} (with $\beta = 0.35$ in (9)), as appropriate.

V. The Motion-Compensated Coder

Our objective is to build a coder for dynamic point clouds that can be implemented in real time with existing technology, which would outperform the use of RAHT and octtrees to compress color and geometry, respectively.

In order to do this, we decided to explore the temporal dimension to remove temporal redundancies, i.e., to explore the fact that the geometry and color of the point cloud may not change much from one frame to another and to use $\mathcal{F}(t)$ as a predictor for $\mathcal{F}(t+1)$. Furthermore, we decided to explore 3D analogs of traditional video compression techniques. Unlike previous approaches, we introduce the use of motion estimation and motion compensation into the compression of dynamic point clouds, in order to achieve higher compression ratios at the expense of lossy coding of the geometry. As far as we know, the present work is the first to explore such a framework.

A. Cube Motion Compensation

The coder, whose diagram is depicted in Fig. 4, is very similar to any traditional video coder in its essence, but is quite different in the details. In many video coders, the frame is broken into blocks of $N \times N$ pixels. Here, the frame is broken into blocks of $N \times N \times N$ voxels, i.e., the voxel space is partitioned into blocks and the list of occupied voxels is likewise partitioned into occupied blocks. So, the occupied block at integer position (b_x, b_y, b_z) in a frame at instant $t+1$ is composed of occupied voxels $v_{i,t+1}$

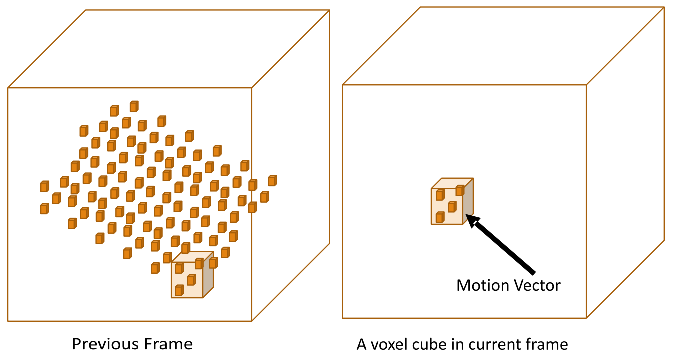


Fig. 5. Motion compensation of an occupied block in the present frame with voxel data within a translated block in the reference frame.

within the block boundaries, i.e., occupied voxels $v_{i,t+1} = [x_{i,t+1}, y_{i,t+1}, z_{i,t+1}, Y_{i,t+1}, U_{i,t+1}, V_{i,t+1}]$ such that

$$\begin{aligned} b_xN &\le x_{i,t+1} < b_xN + N,\\ b_yN &\le y_{i,t+1} < b_yN + N,\\ b_zN &\le z_{i,t+1} < b_zN + N. \end{aligned} \tag{10}$$

Note that each occupied block may have between 1 and N^3 occupied voxels.

The motion compensation process is illustrated in Fig. 5. Each occupied block is associated with a motion vector (MV), whose components (M_x, M_y, M_z) indicate a block in a reference frame that will be used to predict the current block. Let Ω be the set of occupied voxels in a block at position (b_x, b_y, b_z) in frame $\mathcal{F}(t+1)$. Then, Ω can be predicted from the set of voxels $[x_{i,t}, y_{i,t}, z_{i,t}, Y_{i,t}, U_{i,t}, V_{i,t}]$ originally in frame $\mathcal{F}(t)$ such that

$$\begin{aligned} b_xN - M_x &\le x_{i,t} < b_xN + N - M_x,\\ b_yN - M_y &\le y_{i,t} < b_yN + N - M_y,\\ b_zN - M_z &\le z_{i,t} < b_zN + N - M_z. \end{aligned} \tag{11}$$

This set is motion compensated by adding the motion vectors to its coordinates ($x_i \to x_i + M_x$, $y_i \to y_i + M_y$, and $z_i \to z_i + M_z$) to obtain the set Ω_p of voxels $[x_{i,t} + M_x, y_{i,t} + M_y, z_{i,t} + M_z, Y_{i,t}, U_{i,t}, V_{i,t}]$. The set Ω_p is used as a predictor of Ω.

In order to compute a local distortion δ between Ω and Ω_p, we use both correspondence- and projection-based metrics.

For the correspondence-based distortion, as in (5), the correspondences between occupied voxels in Ω and Ω_p are computed as follows.

Let Ω have N_Ω voxels and let Ω_p have N_{Ω_p} voxels. If $N_{\Omega_p} \ge N_\Omega$, then

- Compute all $N_\Omega N_{\Omega_p}$ Euclidean distances across the sets.
- Set $\delta = 0$.
- Find the smallest distance and associate those voxels.
- Let their geometric distance be δ_g and let their color distance be δ_c.
- Update $\delta \to \delta + \delta_g + \beta\delta_c$.
- Remove each voxel from its set (from both Ω and Ω_p).
- Repeat the process until all voxels in Ω are gone.

$\beta = \alpha_c/\alpha_g$ is a constant we use to linearly combine the distortions and maintain their relative importance.

Authorized licensed use limited to: Ricardo De Queiroz. Downloaded on May 04,2020 at 17:43:43 UTC from IEEE Xplore. Restrictions apply.

If $N_{\Omega_p} < N_{\Omega}$, we may duplicate voxels in Ω_p before computing δ.

If, however, the distortion metric is based on projection, we project the voxels in Ω and Ω_p onto their six sides and compute the mean squared error of the Y channel of the individual projections.

Note that the sum of the cube-to-cube distortions will not add up to the overall point cloud distortion under either of the two metrics, because of the occlusions of the projections and of possible correspondences across cube boundaries. Nevertheless we hope the local distortion measures can serve as good approximations to the global ones.

In this article, we do not examine how best to perform motion estimation to establish what the MVs are. Instead we re-use the correspondences that are calculated in the 3D surface reconstruction processes immediately prior to compression [43]. In those processes, each voxel may have a correspondence to a voxel in the previous frame, but we need to use one MV per occupied block. From the correspondences, we first produce a voxel-oriented field of MVs. In other words, we associate each voxel in Ω with a MV. In order to find one MV for the whole set, we take the existing MV in Ω that is the closest to the average of all MVs in Ω. That "median" MV is then assigned to the block containing Ω.

B. Coding Mode Decision

Unlike traditional video coding, where the pixel position is known and its color is to be encoded, here, the need to encode the geometry along with the color makes it a distinct problem. We, so far, have been unable to encode the geometry information at a rate significantly lower than 2.5-3.0 bpv, which is achieved by encoding the geometry using octtrees without any prediction from $\mathcal{F}(t)$ to $\mathcal{F}(t+1)$. Because of that we do not encode geometry residuals. Our coder operates in two modes: either a block is purely motion compensated or it is entirely encoded in intra mode.

We designate frames as types I (intra-coded) and P (predicted). For an I-frame, all blocks are encoded in intra mode, for example using octtree encoding for the geometry and RAHT encoding for the color components. For a P-frame, there should be a reference frame stored in the frame store, typically the previous frame. In a P-frame, we make a mode decision for each occupied block, whether it should be inter-coded (motion-compensated) or intra-coded. In effect, we test whether motion compensation alone produces a good enough approximation of the block. If so, we replace Ω by Ω_p. If not, we encode Ω independently using octtree and RAHT encoding.

The decision is optimized in a rate-distortion (RD) sense. The choice is about representing the block by Ω (*intra*) or by Ω_p (*inter*). Each choice implies rates and distortions for both geometry and color components, i.e. (R_g^{intra}, R_c^{intra}, D_g^{intra}, D_c^{intra}) versus (R_g^{inter}, R_c^{inter}, D_g^{inter}, and D_c^{inter}). Assume octtrees would encode geometry at about 2.5 bits/voxel, i.e. $R_g^{intra} \approx 2.5||\Omega||$. There is no color residue in *inter* mode so that the *inter* rate is only the cost of encoding motion vectors. Similarly, there is no geometry distortion in *intra* mode using

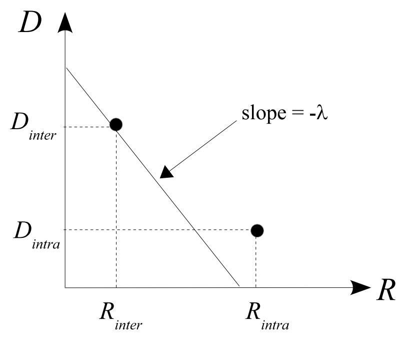


Fig. 6. Rate-distortion performance of different coding modes for a block: (R_{inter}, D_{inter}) for inter-mode and (R_{intra}, D_{intra}) for intra-mode. Minimizing the Lagrangian cost is equivalent to picking the point supported by the line of slope $-\lambda$. Note that a continuum of line slopes may support the same point.

octtrees. Hence, we compute:

$$R_{intra} = R_g^{intra} + R_c^{intra} \approx 2.5||\Omega|| + R_c^{intra} \quad (12)$$
$$R_{inter} = R_g^{inter} + R_c^{inter} = R_{MV} \quad (13)$$
$$D_{intra} = D_g^{intra} + \beta D_c^{intra} = \beta D_c^{intra} \quad (14)$$
$$D_{inter} = D_g^{inter} + \beta D_c^{inter} = \delta, \quad (15)$$

where R_{MV} is the average rate to encode one MV. We build Lagrangian costs for each mode and choose the mode with the smallest cost, i.e., we decide on intra mode if and only if

$$D_{intra} + \lambda R_{intra} < D_{inter} + \lambda R_{inter} \quad (16)$$

and decide on inter mode otherwise, for any fixed $\lambda > 0$.

The decision threshold for a block is illustrated in Fig. 6. Graphically, the points (R_{inter}, D_{inter}) and (R_{intra}, D_{intra}) are very distinct points in the distortion-rate plane, with (typically) $R_{inter} < R_{intra}$ and $D_{inter} > D_{intra}$. Let

$$\lambda^* = \frac{D_{inter} - D_{intra}}{R_{intra} - R_{inter}} > 0 \quad (17)$$

be the magnitude of the slope of the line connecting the two points. Then, the intra mode criterion (16) reduces to

$$\lambda < \lambda^*. \quad (18)$$

That is, a block is encoded as *intra* if and only if its value of λ^* is greater than the globally advertised value of λ, which is fixed across the sequence. One can see that the mode decision for a given block is not excessively sensitive to λ since the choice is between only two RD points and many values of λ may lead to the same mode decision for a given block.

For a P-frame, there are two extremes: intra-coding the whole frame or motion-compensating the whole frame. By varying $0 \leq \lambda < \infty$ from one extreme to the other, the frame can be encoded at these extremes or at various points in between. This is illustrated in Fig. 7.

C. Rate or Distortion Control

The rates and distortions of the color and motion vectors are controlled by a quantizer step Q. Like λ, Q is also a means to trade off rate and distortion. Like similar video coders,

Authorized licensed use limited to: Ricardo De Queiroz. Downloaded on May 04,2020 at 17:43:43 UTC from IEEE Xplore. Restrictions apply.

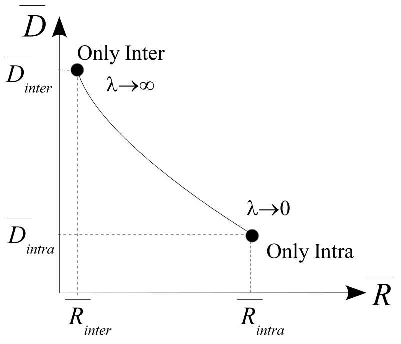


Fig. 7. The extremes of a P-frame are completely motion-compensating the whole frame with average RD performance $(\bar{R}_{inter}, \bar{D}_{inter})$ or encoding it as an I-frame with average RD performance $(\bar{R}_{intra}, \bar{D}_{intra})$. By varying λ, the frame can be encoded at its extremes or at various points in between.

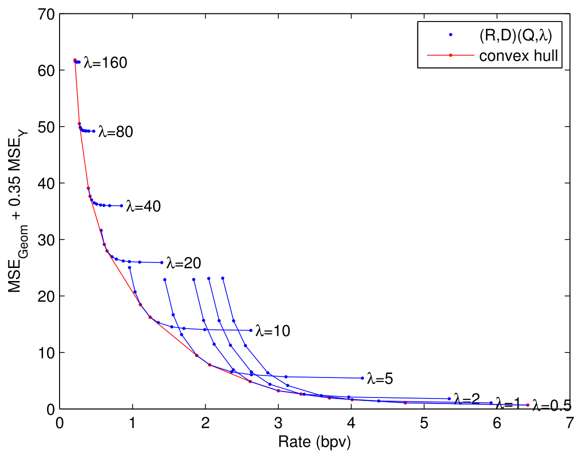


Fig. 8. RD plots for compressing the 56th frame of sequence "Man" as a P-frame using the decompressed 55th frame as an I-frame reference. We fixed λ and varied Q for various values of λ and Q. The convex hull of all points is also shown, which can model a curve relating λ and Q.

the overall coder essentially maps Q and λ to an overall rate R and distortion D. We would like the coder to operate on the lower convex hull (LCH) of all the RD points produced by spanning all Q and λ combinations. Thus, we want to find the λ and Q points that are mapped into the LCH. In order to speed up the process it is sometimes useful to derive a relation $\lambda = f_\lambda(Q)$ that would provide good results across many images. In other words, if the coder is a mapping from (λ, Q) to (R, D), then we look for the curve defined by f_λ in the (λ, Q) plane that is to be mapped into the LCH in the RD plane. It is not always feasible to obtain such a function.

In one example, shown in Fig. 8, the 56th frame of sequence "Man" is compressed as a P-frame using the decompressed 55th frame as a reference. The distortion metric is based on correspondence. Both λ and Q are varied. In the figure one can see many RD curves, each for a fixed λ and varying Q. The LCH of all those curves is also plotted in Fig. 8. The loci of all (λ, Q) in the LCH curve is empirically approximated (using ten frames of sequence "Man") by

$$\lambda = f_\lambda(Q) = Q^2/60, \tag{19}$$

which is one relation we use in our coder.

D. Encoded Data

The compressed frame is locally reconstructed and then post-processed, in a process we will explain later, before being placed in a frame store where it will be available for motion compensation and prediction of the next frames (see Fig. 4).

The encoded data comprises:

- Sequence parameters: group-of-frames (GOF) length (i.e. repetition period of intra-coded-only frames), W, N, η, γ (see next subsection).
- Number of occupied blocks.
- List of occupied blocks.
- Coding mode per occupied block.
- For intra-coded blocks: occupied voxels encoded using octtrees and color data encoded using RAHT and quantizer stepsize Q.
- For inter-coded blocks: motion vectors encoded using RAHT and quantizer stepsize Q_{mv}.

E. In-Loop Processing for Geometry Distortions

Unlike previous coders for dynamic point clouds, in this work, we apply lossy coding of the geometry. Even though our distance metric applied to two sets of point clouds may be useful as an objective measurement of the coding quality, small distortions to the geometry can cause blocking artifacts that are quite annoying. We decided to smooth the surfaces and to fill the gaps (rips) using adaptations to 3D voxels of traditional operators. Our processing is essentially an in-loop deblocking filter adapted to 3D voxels.

We first filter the geometry elements to smooth the surface discontinuities caused by mismatch in the motion compensation process. Without loss of generality, for example, using the first dimension (x), the filter is:

$$\hat{x}_i = \frac{\sum_{j, d_{ij} < \eta} x_j \rho^{d_{ij}}}{\sum_{j, d_{ij} < \eta} \rho^{d_{ij}}} \tag{20}$$

where $d_{ij} = ||v_i - v_j||$ is the distance between voxels i and j, and η controls the neighborhood size and the intensity of filtering. Such an operation may cause further holes in the geometry surfaces. Because of that, assuming the discontinuities will be more prominent at the cube boundaries, we only replace voxels that are near the boundaries of a motion-compensated cube. Furthermore, to avoid creating more holes, we do not allow the voxel position to move away from the border. In effect, if x_i is at the border, x_i is not changed but y_i and z_i are replaced by $\hat{y}_i$ and $\hat{z}_i$, respectively.

After filtering, we try to close gaps using morphological operations on the voxel geometry. We define simple dilation as replicating each existing occupied voxel to its 26 volumetric neighbors, if such a neighbor is not occupied yet. We also define simple erosion by erasing a given voxel if any of its 26 neighbors in 3D is unoccupied. We repeat the dilation γ times and do the same number of erosion operations. The process is parallel to a morphological closing operation. Holes up to 2γ voxels wide may be patched by the process.

The operation is only applied to inter-coded cubes. Processed cubes not only are made available to the decoder and to the display, but can also be placed in the frame store so

Authorized licensed use limited to: Ricardo De Queiroz. Downloaded on May 04,2020 at 17:43:43 UTC from IEEE Xplore. Restrictions apply.

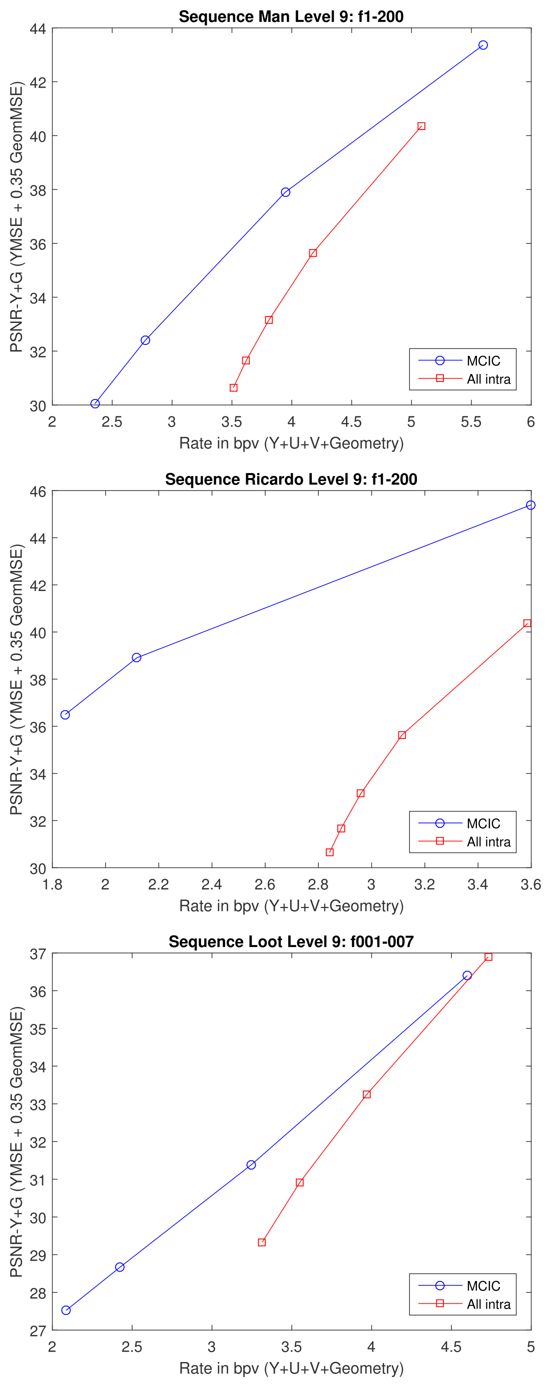


Fig. 9. RD plots for sequence "Man", "Ricardo", and "Loot" using correspondence-based distortion: PSNR-Y+G. RD points were obtained by averaging rate or distortion for each frame over the entire sequence. (GOF = 8, $\eta = \gamma = 2$, $\lambda = Q^2/60$, coding mode decision based on δ_{Y+G}).

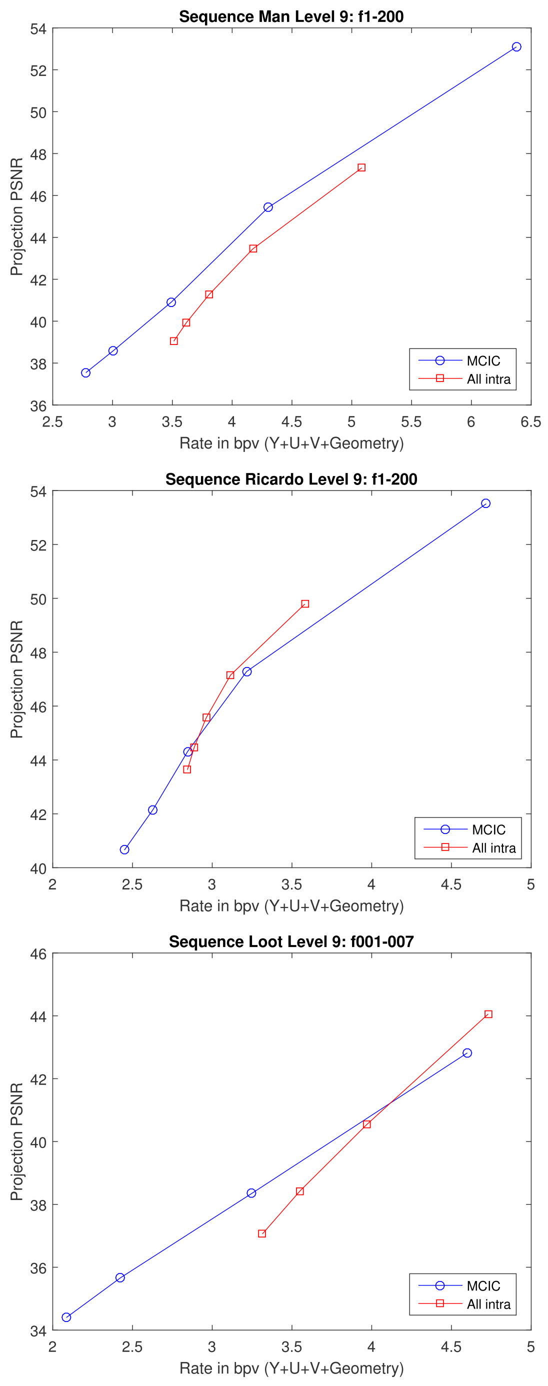


Fig. 10. RD plots for sequences "Man", "Ricardo" and "Loot" using a projection-based distortion metric. RD points were averaged for each frame over the entire sequence. (GOF=8, $\eta = \gamma = 2$, exponential $f_\lambda(Q)$, coding mode decision based on projections.)

that the coder can use the frames processed in-loop to perform motion compensation for future frames.

F. Complexity

Our target is real-time implementation at 30 frames per second and our tests were all carried using Matlab® which is not a parameter for real-time implementation. However, the most time consuming part is the motion estimation (point cloud correspondence), which was not considered in this paper as discussed in Sec. III, but it has been implemented in a real time system. We know from other efforts that RAHT has also been implemented in real time. In relative terms RAHT is a significant portion, as is mode decision, since it computes many projections. The coder itself was designed to be simple. The in-loop filtering, specially the morphological closing, is one of the most time-consuming operation as it was implemented in a straightforward brute-force way. One needs to develop fast morphological closing algorithms for sparse

Authorized licensed use limited to: Ricardo De Queiroz. Downloaded on May 04,2020 at 17:43:43 UTC from IEEE Xplore. Restrictions apply.

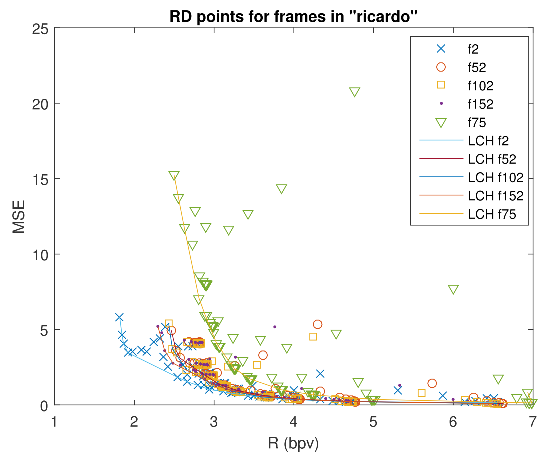


Fig. 11. RD plots for different frames in sequence "Ricardo" indicating their LCH under a projection-based distortion metric.

clouds, in the same way there are fast algorithms for *dense* binary images. We estimate current technology already may allow for real time implementation of the proposed coder.

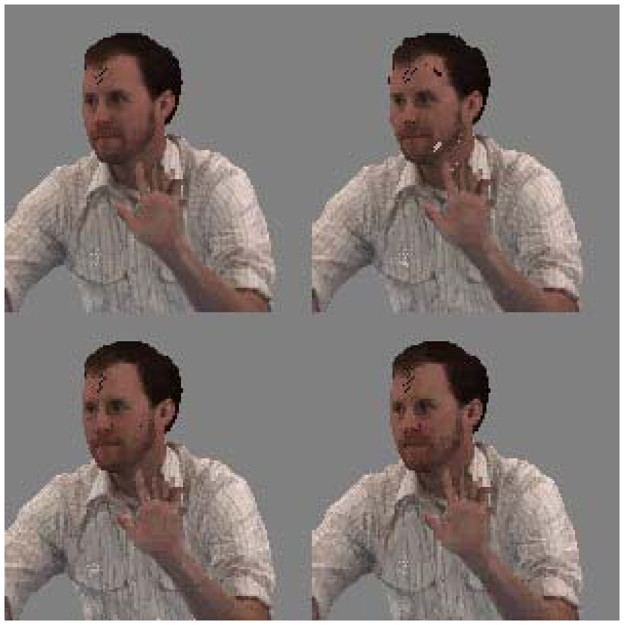
VI. Simulation Results

Simulations were carried out to demonstrate the capabilities of the proposed system, to which we refer as a motion-compensated intra-frame coder (MCIC). We have three datasets, all with $W = 512$, and captured at a rate of 30 frames per second. Two sequences ("Man" and "Loot") represents a full body in 3D, while the other sequence ("Ricardo") is intended for video conferencing and thus just a frontal upper body is represented. We used a GOF length of 8, i.e. interspersing 7 P-frames between every I-frame. Since P-frames are degraded versions of I-frames (lower rate and higher distortion) and assuming the compression ratio should be similar for every intra-coded part in every frame, the rate peaks at every I-frame. We used $Q_{mv} = 1$ and varied Q to obtain our RD curves. Values of Q in the range of 5 through 45 were used for the MCIC, while the purely intra coder used quantizer values ranging from 10 to 50.

As for the in-loop filtering, after many tests, we have chosen to use $\gamma = 2$ and $\eta = 2$. Nevertheless the best choice of parameters and the filtering approach is far from being settled.

For the correspondence-based distortion metric (PSNR-Y+G), a simple approximation to $f_\lambda(Q)$ yields very good results for both sequences we tested. We derived the function from one single frame and yet it performs adequately for all other frames under this metric. Using $\lambda = f_\lambda(Q) = Q^2/60$, we obtained the RD plots for the three sequences shown in Fig. 9. The RD points are averages over all the 200 frames of the sequences "Man" and "Ricardo" and 8 frames of sequence "Loot". From the figure, one can easily infer the superior performance of the MCIC over purely intra coder under this metric.

For the projection-based distortion metric (PSNR-P), we were able to find a curve f_λ that works well for sequence "Man," and the RD curves are presented in Fig. 10. The RD points were obtained by averaging the rate or distortion of each frame in the entire sequence. The curve f_λ was set from the results of a single frame with a simple least-squares



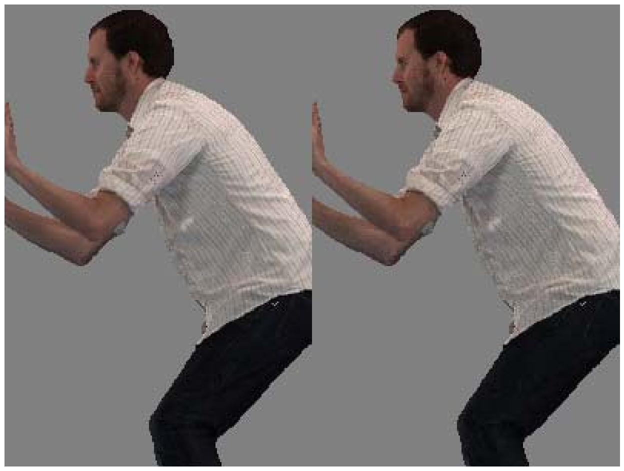


Fig. 12. Top comparison: front projection rendering comparing MCIC against intra-coder (RAHT) for frame 58 in sequence "Man". (top left) original; (top right) MCIC under a correspondence-based distortion metric; (bottom left) MCIC under a projection-based distortion metric; (bottom right) RAHT. Rates are about 3.7 bpv for all three compression cases. Bottom comparison: front projection rendering comparing the original frame (left) with its MCIC reconstructed (right) after a compression to 4.4 bpv.

fitting in the log domain, $f_\lambda(Q) = e^{a_3\ Q^3 + a_2\ Q^2 + a_1\ Q + a_0}$, for $a_0 = -71873$, $a_1 = 0.35998$, $a_2 = -7.6525 \times 10^{-3}$, $a_3 = 5.6329 \times 10^{-5}$. Despite the sub-optimality of fitting with least squares instead of a proper RD criterion, the solution yields good results for sequence "Man," as depicted in Fig. 10.

The results are not so simple for sequence "Ricardo" under PSNR-P. It is not easy to find a function f_λ that works for most frames. Not only it is a rapidly changing sequence, but its noise level and the fact of being front-faced makes the estimation of the global distortion from local cube projections more difficult. The non-stationarity is illustrated in Fig. 11, which shows the RD points and the corresponding LCH for different frames of the sequence. Note how widely the LCH curves vary among frames. It should be possible to improve significantly the RD performance of MCIC on this sequence by fixing λ and choosing an optimal value of Q separately

Authorized licensed use limited to: Ricardo De Queiroz. Downloaded on May 04,2020 at 17:43:43 UTC from IEEE Xplore. Restrictions apply.

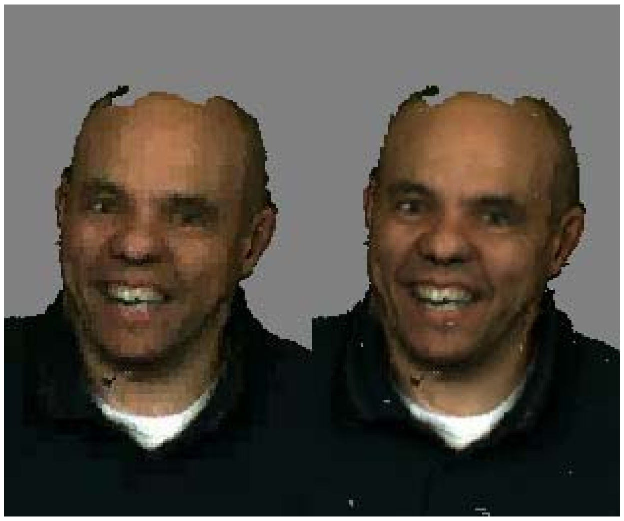


Fig. 13. Front projection rendering comparing MCIC (correspondence-based distortion) on the right against intra-coder (RAHT) on the left, for frame 60 in sequence "Ricardo," both encoded at about 2.6 bpv.

for each frame, e.g., using exhaustive search, though it may be impractical.

It is difficult to evaluate which distortion metric would be more subjectively appropriate. As an example, Fig. 12 shows a comparison between methods for frame 58 of sequence "Man." It shows a zoomed view of the original frame and equivalent views of the frames compressed using MCIC under correspondence- and projection-based metrics, and the intra coder. The rate is about 3.7 bpv for all cases. The correspondence-based result produces the best color, but with a few speckles of noise caused by geometry shifts. Those artifacts are caused by introducing holes in the object shell, which causes the view to show the background color, which usually contrasts against the foreground shell color. These holes (rips) were not corrected by the two post-processing operators, for some reason, and that is an area we may want to investigate in the future. Another comparison can be drawn from the bottom of Fig. 12, which shows a zoomed view of the original 75th frame of sequence "Man," side-by-side with its decompressed version, using MCIC (projection-based distortion for mode selection) at around 4.4 bpv.

We also include a similar comparison for a frame of sequence "Ricardo," which was compressed using MCIC (correspondence-based distortion for mode selection) and intra coding, at a rate around 2.6 bpv, which is shown in Fig. 13.

VII. Conclusion

We have developed a novel motion-compensation scheme for use with dynamic point clouds. The encoder works on dividing the cloud into blocks of occupied voxels and deciding for each one if the block should be intra-coded or simply motion-compensated from the past frame. The replacement of intra-coded data for a slightly distorted (or not) set of voxels saves many bits, but introduces errors not only to the voxel colors, but also also to their positions (geometry). In effect, a P-frame is a degraded I-frame whose extra distortion is found to be "worth it" in an RD sense. With that extra degradation we are able to extend the bit-rate range below where the intra coder can effectively operate and to exceed the performance of the intra coder at any rate under a given objective distortion measure. From that perspective, the coder results are satisfactory.

As a new concept for a new application, much has still to be fine tuned and perfected. For example, the post-processing (in-loop or otherwise) is far from reaching its peak performance. Both the morphological and the filtering operations are not well understood in this context. Similarly, the distortion metrics or the voxel matching methods are not developed to a satisfactory point. There is still plenty of work to be done to extend the present framework to use B-frames (bi-directional prediction) and to extend the GOF to a more typical IBBPBBP... format. Furthermore, we want to use adaptive block sizes, which are optimally selected in an RD sense and we also want to encode both the geometry and the color residues for the predicted (P and B) blocks. Finally, rather than re-using the correspondences from the surface reconstruction among consecutive frames, we want to develop efficient motion estimation methods for use with our coder. Each of these enhancements should improve the coder performance, such that there is a continuous sequence of improvements in this new frontier to be explored.

Acknowledgment

The authors thank the Microsoft HCap group for providing the "Man" sequence, and Charles Loop, Qin Cai, and Mingsong Dou of the Microsoft I3D group for providing the "Ricardo" sequence and its dense correspondence data, as well engaging in fruitful discussions. They also thank 8i Labs, Inc. for providing the sequence "Loot."

References

[1] J. Lanier, "Virtually there," *Sci. Amer.*, pp. 66–75, Apr. 2001.

[2] Z. Yang *et al.*, "TEEVE: The next generation architecture for tele-immersive environments," *Proc. 7th IEEE Int. Symp. Multimedia (ISM)*, Irvine, CA, USA, Dec. 2005, pp. 112–119.

[3] H. Fuchs, A. Sate, and J.-C. Bazin, "Immersive 3D telepresence," *Computer*, vol. 47, no. 7, pp. 46–52, Jul. 2014.

[4] J. Wall *et al.*, "REVERIE: Natural human interaction in virtual immersive environments," in *Proc. IEEE Int. Conf. Image Process.*, Paris, France, Sep. 2014.

[5] Y. Altunbasak, J. Apostolopoulos, P. A. Chou, and B.-H. Juang, "Realizing the vision of immersive communication," *IEEE Signal Process. Mag.*, vol. 28, no. 1, pp. 18–19, Jan. 2011.

[6] P. A. Chou, "Advances in immersive communication:(1) Telephone, (2) television,(3) teleportation," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 9, no. 1s, p. 41, 2013.

[7] J. G. Apostolopoulos, P. A. Chou, B. Culbertson, T. Kalker, M. D. Trott, and S. Wee, "The road to immersive communication," *Proc. IEEE*, vol. 100, no. 4, pp. 974–990, Apr. 2014.

[8] C. Zhang, Q. Cai, P. Chou, Z. Zhang, and R. Martin-Brualla, "Viewport: A fully distributed immersive teleconferencing system with infrared dot pattern," *IEEE Multimedia Mag.*, vol. 20, no. 1, pp. 17–27, Jan. 2013.

[9] C. Loop, C. Zhang, and Z. Zhang, "Real-time high-resolution sparse voxelization with application to image-based modeling," in *Proc. High-Perform. Graph. Conf.*, 2013, pp. 73–79.

[10] P. Merkle, A. Smolic, K. Muller, and T. Wiegand, "Multi-view video plus depth representation and coding," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2007, p. I-201.

[11] M. Hebert, C. Caillas, E. Krotkov, I. Kweon, and T. Kanade, "Terrain mapping for a roving planetary explorer," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 1989, pp. 997–1002.

[12] R. Triebel, P. Pfaff, and W. Burgard, "Multi-level surface maps for outdoor terrain mapping and loop closing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, Oct. 2006, pp. 2276–2282.

Authorized licensed use limited to: Ricardo De Queiroz. Downloaded on May 04,2020 at 17:43:43 UTC from IEEE Xplore. Restrictions apply.

[13] O. Devillers and P.-M. Gandoin, "Geometric compression for interactive transmission," in *Proc. IEEE Vis.*, Oct. 2000, pp. 319–326.

[14] X. S. J. Gu Gortler and H. Hoppe, "Geometry images," in *Proc. Annu. Conf. Comput. Graph. Interact. Techn.*, 2002, pp. 355–361.

[15] H. M. Briceno, P. V. Sander, L. McMillan, S. Gortler, and H. Hoppe, "Geometry videos: A new representation for 3D animations," in *Proc. ACM Symp. Comput. Animation*, 2003, pp. 136–146.

[16] S. Gupta, K. Sengupta, and A. A. Kassim, "Registration and partitioning-based compression of 3D dynamic data," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 13, no. 11, pp. 1144–1155, Nov. 2003.

[17] H. Habe, Y. Katsura, and T. Matsuyama, "Skin-off: Representation and compression scheme for 3D video," in *Proc. Picture Coding Symp.*, 2004, pp. 301–306.

[18] 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.

[19] S.-R. Han, T. Yamasaki, and K. Aizawa, "Time-varying mesh compression using an extended block matching algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 11, pp. 1506–1518, Nov. 2007.

[20] L. Váša and V. Skala, "Geometry-driven local neighbourhood based predictors for dynamic mesh compression," *Comput. Graph. Forum*, vol. 29, no. 6, pp. 1921–1933, 2010.

[21] R. Mekuria, M. Sanna, S. Asioli, E. Izquierdo, D. C. A. Bulterman, and P. Cesar, "A 3D tele-immersion system based on live captured mesh geometry," in *Proc. ACM Multimedia Syst. Conf.*, Oslo, Norway, Feb. 2013, pp. 24–35.

[22] H. Q. Nguyen, P. A. Chou, and Y. Chen, "Compression of human body sequences using graph wavelet filter banks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, May 2014, pp. 6152–6156.

[23] J. Hou, L. Chau, N. Magnenat-Thalmann, and Y. He, "Compressing 3D human motions via keyframe-based geometry videos," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 25, no. 1, pp. 51–62, Jan. 2015.

[24] W. Sun, G. Cheung, P. A. Chou, D. Florencio, C. Zhang, and O. Au, "Rate-constrained 3D surface estimation from noise-corrupted multiview depth videos," *IEEE Trans. Image Process.*, vol. 23, no. 7, pp. 3138–3151, Jul. 2014.

[25] T. Ochotta and D. Saupe, "Compression of point-based 3D models by shape-adaptive wavelet coding of multi-height fields," in *Proc. Eurographics Conf. Point-Based Graph.*, 2004, pp. 103–112.

[26] R. Schnabel and R. Klein, "Octree-based point-cloud compression," in *Proc. Symp. Point-Based Graphics*, Jul. 2006, pp. 111–120.

[27] 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.

[28] C. Zhang, D. Florêncio, and C. Loop, "Point cloud attribute compression with graph transform," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2014, pp. 2066–2070.

[29] D. Thanou, P. A. Chou, and P. Frossard, "Graph-based motion estimation and compensation for dynamic 3D point cloud compression," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2015, pp. 3235–3239.

[30] D. Thanou, P. A. Chou, and P. Frossard, "Graph-based compression of dynamic 3D point cloud sequences," *IEEE Trans. Image Process.*, vol. 25, no. 4, pp. 1765–1778, Apr. 2016.

[31] 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.*, May 2012, pp. 778–785.

[32] R. L. de Queiroz and P. A. Chou, "Compression of 3D point clouds using a region-adaptive hierarchical transform," *IEEE Trans. Image Process.*, vol. 25, no. 8, pp. 3947–3956, Aug. 2016.

[33] Y. Wang *et al.*, "Handling occlusion and large displacement through improved RGB-D scene flow estimation," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 26, no. 7, pp. 1265–1278, Jul. 2016.

[34] A. L. Yuille and N. M. Grzywacz, "A mathematical analysis of the motion coherence theory," *Int. J. Comput. Vis.*, vol. 3, no. 2, pp. 155–175, Jun. 1989.

[35] S. Hadfield and R. Bowden, "Scene particles: Unregularized particle based scene flow estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 3, pp. 564–576, Mar. 2014.

[36] J.-M. Gottfried, J. Fehr, and C. S. Garbe, "Computing range flow from multi-modal Kinect data," in *Advances in Visual Computing* (Lecture Notes in Computer Science), vol. 6938. Berlin, Germany: Springer, 2011, pp. 758–767.

[37] E. Herbst, X. Ren, and D. Fox, "RGB-D flow: Dense 3-D motion estimation using color and depth," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2013, pp. 2276–2282.

[38] J. Quiroga, F. Devernay, and J. L. Crowley, "Local/global scene flow estimation," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2013, pp. 3850–3854.

[39] X. Zhang, D. Chen, Z. Yuan, and N. Zheng, "Dense scene flow based on depth and multi-channel bilateral filter," in *Proc. Springer Comput. Vis. (ACCV)*, 2012, pp. 140–151.

[40] M. Hornacek, A. Fitzgibbon, and R. Carsten, "SphereFlow: 6 DoF scene flow from RGB-D pairs," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2014, pp. 3526–3533.

[41] J. Quiroga, T. Brox, F. Devernay, and J. Crowley, "Dense semi-rigid scene flow estimation from RGBD images," in *Proc. Springer Comput. Vis. (ECCV)*, 2014, pp. 567–582.

[42] Y. Niu, A. Dick, and M. Brooks, "Compass rose: A rotational robust signature for optical flow computation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 1, pp. 63–73, Jan. 2014.

[43] M. Dou, J. Taylor, H. Fuchs, A. W. Fitzgibbon, and S. Izadi, "3D Scanning Deformable Objects with a Single RGBD Sensor," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 493–501.

Ricardo L. de Queiroz (M'88–SM'99–F'17) received the Engineer degree from Universidade de Brasilia, Brazil, in 1987, the M.Sc. degree from Universidade Estadual de Campinas, Brazil, in 1990, and the Ph.D. degree from The University of Texas at Arlington, in 1994, all in electrical engineering.

From 1990 to 1991, he was with the DSP Research Group, Universidade de Brasilia, as a Research Associate. He joined Xerox Corp. in 1994, where he was a member of the Research Staff until 2002. From 2000 to 2001, he was also an Adjunct Faculty with the Rochester Institute of Technology. He joined the Electrical Engineering Department, Universidade de Brasilia, in 2003. In 2010, he became a Full (Titular) Professor with the Computer Science Department, Universidade de Brasilia. During 2015, he has been a Visiting Professor with the University of Washington, Seattle.

Dr. de Queiroz was an Elected Member of the IEEE Signal Processing Society's Multimedia Signal Processing (MMSP) and the Image, Video and Multidimensional Signal Processing (IVMSP) Technical Committees. He was an Editor of the *EURASIP Journal on Image and Video Processing*, the IEEE SIGNAL PROCESSING LETTERS, and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY. He is an Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING. He has been appointed as an IEEE Signal Processing Society Distinguished Lecturer from 2011 to 2012.

His research interests include image and video compression, multirate signal processing, and color imaging. He has been actively involved with the Rochester Chapter of the IEEE Signal Processing Society, where he served as a Chair and organized the Western New York Image Processing Workshop since its inception until 2001. He helped organizing IEEE SPS Chapters in Brazil and chaired the Brasilia IEEE SPS Chapter. He was the General Chair of ISCAS'2011, MMSP'2009, and SBrT'2012. He was also part of the organizing committee of ICIP'2002, ICIP'2012, ICIP'2014 and ICIP'2016. He is a member of the Brazilian Telecommunications Society.

Philip A. Chou (M'81–SM'00–F'03) received the B.S.E degree in electrical engineering and computer science from Princeton University, Princeton, NJ, and the M.S. degree in electrical engineering and computer science from the University of California, Berkeley, , in 1980 and 1983, respectively, and the Ph.D. degree in electrical engineering from Stanford University in 1988. From 1988 to 1990, he was a member of Technical Staff at AT & T Bell Laboratories, Murray Hill, NJ. From 1990 to 1996, he was a member of Research Staff, Xerox Palo Alto Research Center, Palo Alto, CA. In 1997, he was a Manager of the Compression Group, VXtreme, Mountain View, CA, an Internet video startup before it was acquired by Microsoft. From 1998 to 2016, he was a Principal Researcher with Microsoft Research, Redmond, Washington, where he was managing the Communication and Collaboration Systems Research Group from 2004 to 2011. He served as Consulting Associate Professor with Stanford University from 1994 to 1995, Affiliate Associate Professor with the University of Washington from 1998 to 2009, and an Adjunct Professor with the Chinese University of Hong Kong since 2006. He is currently with 8i.com, where he leads the effort to compress and communicate volumetric media for augmented and virtual reality.

Authorized licensed use limited to: Ricardo De Queiroz. Downloaded on May 04,2020 at 17:43:43 UTC from IEEE Xplore. Restrictions apply.