MOTION-COMPENSATED COMPRESSION OF POINT CLOUD VIDEO

R. L. de Queiroz

Universidade de Brasilia
Brasilia, Brazil
queiroz@ieee.org

P. A. Chou

8i Labs Inc.
Bellevue, WA, USA
pachou@ieee.org

ABSTRACT

3D immersive communications are trending as real-time point clouds capture and display of point cloud video become feasible. This paper presents a novel motion-compensated approach to encoding dynamic voxelized point clouds (VPC) at low bit rates. A simple coder breaks the VPC into blocks which are intra-frame coded or replaced by a motion-compensated version of a block in the previous frame. The decision is optimized in a rate-distortion sense, encoding with distortion both geometry and the color, at 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.

1. INTRODUCTION

In immersive 3D communication systems [1]-[5], a 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) [6]. 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*, *polygonal meshes*, or *point clouds* . An alternative to point clouds are *sparse voxel arrays*, or *voxel clouds*, which are arbitrary collections of voxels with a volumetric aspect. Yet another possible representation of 3D data is simply a set of color and depth maps, sometimes called multiview video plus depth (MVD) [7]. This is a low-level representation close to the RGBD sensors. Closely related to color and depth maps are elevation maps [8] and multi-level surface maps [9].

Here, the 3D representation of choice is sparse voxel arrays, or *voxelized point clouds*. Compared to arbitrary point clouds, they have implementation advantages and are highly efficient for real-time processing of captured 3D data [6].

Each representation employs its own compression techniques, such as [10]–[20] for polygonal meshes, and [7], [21] for multiview video plus depth. Previous approaches to compressing point clouds include [22]–[28]. The most efficient of these are based on voxelization. We believe [25] and [26], until recently, represented the state-of-the-art in (voxelized) point cloud color compression. We have recently developed a coder that is able to match or outperform existing methods at a reduced cost [29]. 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 [30].

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

2. VOXELIZED POINT CLOUDS

We represent 3D data by voxelized point clouds. A point cloud is a set of points $\{\nu\}$, each point ν 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, ()e.g. Y, U, V). 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. 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 $\mathbb{Z}_W = \{0, \ldots, W-1\}$, $W = 2^D$ is its width, and D is an integer. 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 $\{\nu_i\}$ in which each voxel

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

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 $\mathcal{F}(t) = \{\nu_{it}\}$, which is represented as a list of voxels

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

Note that different frames may be represented by different lists of voxels, so there is no real relation between $\nu_{i,t}$ and $\nu_{i,t+1}$, since the indexing of the voxels in the lists is arbitrary. Moreover, different frames may have different numbers of occupied voxels.

The temporal correspondence among voxels of adjacent frames is not treated here, and left for the fuller version of this paper [30].

A large amount of work exist in the subject being found for example in [31],[33]-[41] It is, in general, a very complex process that is not friendly to real-time communications.

3. DISTORTION METRICS AND RESIDUALS

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 [6],[28]. *Inter-frame* compression of the geometry can also be performed using octtrees and exclusive-OR between sets of occupied voxels in successive frames [26],[28]. Here, we attempt to *predict* the geometry as well as the color, and to code the prediction residuals in a rate-distortion optimized way.

The geometry distortion (combined or not with the color distortion) has not been well defined yet. We can devise a correspondence- and a projection-based distortion computation.

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 $\mathcal{F}(t)$. From all pairing associations we can compute the mean-squared error (MSE) for both geometry and colors, and, from it, the corresponding peak signal-to-noise ratio (PSNR). In particular, the MSE we use only accounts for the Y component and linearly blends both distortion measures (δ_c and δ_g).

$$\delta_{Y+G} = \delta_c + \beta\delta_g. \tag{3}$$

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 distortion metric is the MSE (or PSNR) in between the two corresponding composite images, each with the 6 orthogonal projections of the original or reconstructed Y frames.

4. 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.

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.

4.1. Cube motion compensation

The coder, whose diagram is depicted in Fig. 2, is very similar to any traditional video coder in its essence, but is quite different in the details. The frame is broken into blocks of $N \times N \times N$ voxels. So, the occupied block at integer position (b_x, b_y, b_z) is composed of occupied voxels ν_{it} within the block boundaries, i.e., occupied voxels $\nu_{it} = [x_{it}, y_{it}, z_{it}, Y_{it}, U_{it}, V_{it}]$ such that

$$\begin{aligned} b_x N \le x_{it} < b_x N + N, \\ b_y N \le y_{it} < b_y N + N, \\ b_z N \le z_{it} < b_z N + N. \end{aligned} \tag{4}$$

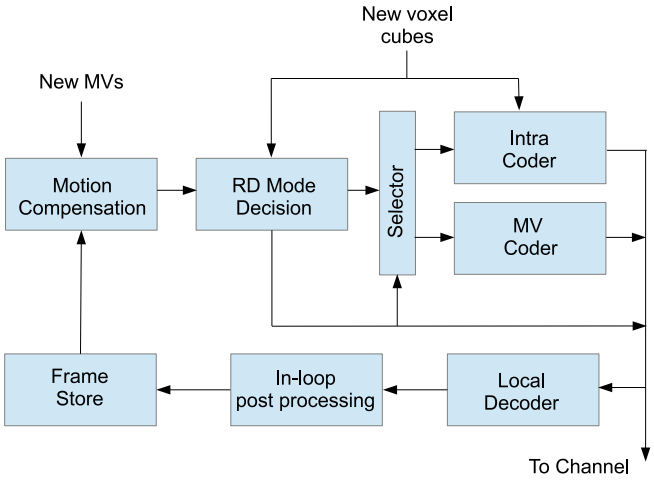

Fig. 2. Encoder diagram.

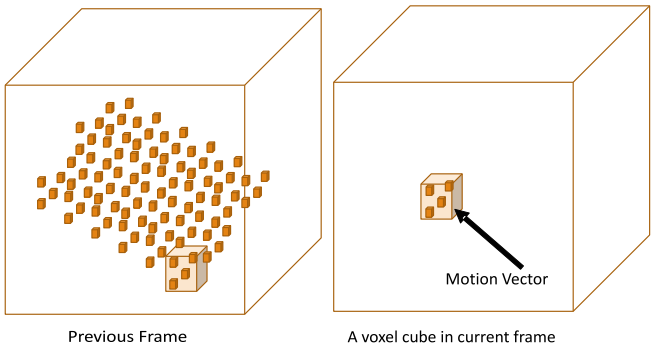

Fig. 3. Motion compensation of an occupied block in the present frame with voxel data within a translated block in the reference frame.

An occupied block may have between 1 and N^3 occupied voxels.

The motion compensation process is illustrated in Fig. 3. 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 $\mathcal{F}(t)$. Then, Ω can be predicted from the set of voxels $[x_{i,t-1}, y_{i,t-1}, z_{i,t-1}, Y_{i,t-1}, U_{i,t-1}, V_{i,t-1}]$ originally in frame $\mathcal{F}(t-1)$ such that

$$\begin{aligned} b_x N - M_x \le x_{i,t-1} < b_x N + N - M_x, \\ b_y N - M_y \le y_{i,t-1} < b_y N + N - M_y, \\ b_z N - M_z \le z_{i,t-1} < b_z N + N - M_z. \end{aligned} \tag{5}$$

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 $[x_{i,t-1} + M_x, y_{i,t-1} + M_y, z_{i,t-1} + M_z, Y_{i,t-1}, U_{i,t-1}, V_{i,t-1}]$. The set Ω_p is used as a predictor of Ω.

In order to compute a local distortion δ between Ω and Ω_p, with N_Ω and N_{Ω_p} voxels, respectively, we set $\delta = 0$ and compute all $N_\Omega N_{\Omega_p}$ Euclidean distances across the sets. We associate the voxels with the shortest distance, remove them, and update $\delta \to \delta + \delta_g + \beta\delta_c$, where δ_g and δ_c are the geometry and color distances. We repeat the process until all voxels in Ω are gone. If $N_{\Omega_p} < N_\Omega$, we may duplicate voxels in Ω_p beforehand. If we opt for a projection-based distortion, the MSE in between the projections of Ω and Ω_p is

computed instead.

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 [41]. 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 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.

4.2. Coding mode decision

Geometry prediction and residuals are a distinct problem. We, so far, have been unable to encode the geometry prediction-error at a rate significantly lower than 2.5-3.0 bpv, which is achieved by encoding the geometry using octtrees without any inter-frame prediction. 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: (R_g^{intra}, R_c^{intra}, D_g^{intra}, D_c^{intra}) and (R_g^{inter}, R_c^{inter}, D_g^{inter}, and D_c^{inter}). We then compute

$$R_{intra} = R_g^{intra} + R_c^{intra} \approx 2.5||\Omega|| + R_c^{intra} \tag{6}$$

$$R_{inter} = R_g^{inter} + R_c^{inter} = R_{MV} \tag{7}$$

$$D_{intra} = D_g^{intra} + \beta D_c^{intra} = \beta D_c^{intra} \tag{8}$$

$$D_{inter} = D_g^{inter} + \beta D_c^{inter} = \delta, \tag{9}$$

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

$$D_{intra} + \lambda R_{intra} < D_{inter} + \lambda R_{inter} \tag{10}$$

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

The points (R_{inter}, D_{inter}) and (R_{inter}, D_{inter}) are very distinct points in the distortion-rate plane, with (typically) $R_{inter} < R_{intra}$ and $D_{inter} > D_{intra}$. Let

$$\lambda^* = \frac{D_{inter} - D_{intra}}{R_{intra} - R_{inter}} > 0 \tag{11}$$

be the magnitude of the slope of the line connecting the two points. Then, the intra mode criterion (10) reduces to $\lambda < \lambda^*$. 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.

4.3. 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, 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. We simplify the process correlating λ and Q as $\lambda = f_\lambda(Q)$. Details can be found in [30].

4.4. 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:

$$\hat{x}_i = \frac{\sum_{j,d_{ij}<\eta} x_j \rho^{d_{ij}}}{\sum_{j,d_{ij}<\eta} \rho^{d_{ij}}} \tag{12}$$

where $d_{ij} = ||\nu_i - \nu_j||$ 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 that the coder can use the frames processed in-loop to perform motion compensation for future frames.

5. 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 two datasets, both with $W = 512$, 200 frames, and captured at a rate of 30 frames per second. One sequence ("man") 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 GOP 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

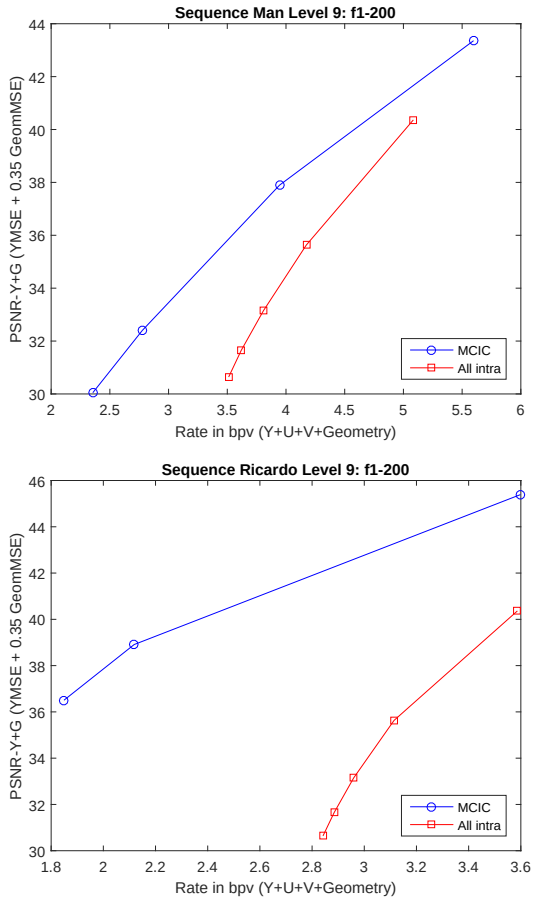

Fig. 4. RD plots for sequence "man" and "upper body" using PSNR-Y+G. RD points are averages over the entire sequence.

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 sequences "man" and "Ricardo" shown in Fig 4. The RD points are averages over all the 200 frames of the sequences. From the figure, one can easily infer the superior performance of the MCIC over purely intra coder under this metric.

We do not have room to present and discuss the results using a projection-based distortion metric (PSNR-P), which were left to the full version of the present paper [30].

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. 6.

Fig. 5. Front projection rendering comparison of frame 58 in sequence "man" at 3.7 bpv. Original, MCIC, MCIC designed under a projection-based distortion metric, RAHT.

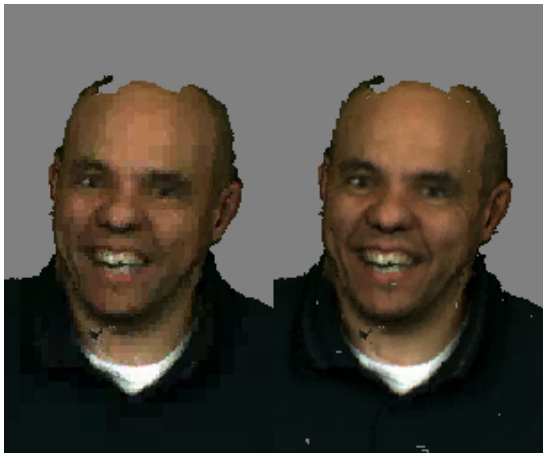

Fig. 6. Front projection rendering comparing MCIC (right) and intra-coder (RAHT, left). Frame 60 at 2.6 bpv.

6. CONCLUSIONS

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. However, much still needs to be done.

7. REFERENCES

[1] J. Lanier, "Virtually There," *Scientific American*, pp. 66-75, Apr. 2001.

[2] Y. Altunbasak, et al., "Realizing the vision of immersive communication," *IEEE Signal Processing Magazine*, Jan. 2011.

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

[4] J. G. Apostolopoulos, et. al., "The road to immersive communication," *Proc. IEEE*, vol. 100, no. 4, pp. 974–990, 2014.

[5] C. Zhang, et al., "Viewport: a fully distributed immersive teleconferencing system with infrared dot pattern," *IEEE Multimedia*, vol. 20, no. 1, pp. 17–27, 2013.

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

[7] P. Merkle, et al., "Multi-view video plus depth representation and coding," in *IEEE Int. Conf. Image Processing*, 2007.

[8] M. Hebert, et al., "Terrain mapping for a roving planetary explorer," in *Proc. IEEE Int. Conf. Robotics & Automation*, 1989.

[9] R. Triebel, P. Pfaff, and W. Burgard, "Multi-level surface maps for outdoor terrain mapping and loop closing," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems* , 2006.

[10] O. Devillers and P. Gandoin, "Geometric compression for interactive transmission," in *IEEE Visualization*, pp. 319–326, 2000.

[11] X. Gu, S. J. Gortler., and H. Hoppe, "Geometry images," in *Annual Conf. Computer Graphics and Interactive Techniques*, pp. 355–361, 2002.

[12] H. M. Briceno, et al., "Geometry Videos: A New Representation for 3D Animations ," in *ACM Symp. Computer Animation*, pp. 136–146, 2003.

[13] 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.

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

[15] J. Peng, Chang-Su Kim, and C. C. Jay Kuo, "Technologies for 3D mesh compression: A survey," *J. Vis. Comun. and Image Representation*, vol. 16, no. 6, pp. 688–733, Dec. 2005.

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

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

[18] R. Mekuria, et al., "A 3D Tele-Immersion System Based on Live Captured Mesh Geometry," *ACM Multimedia Systems Conference*, Oslo, Feb. 2013.

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

[20] J. Hou, et al., "Compressing 3D human motions via keyframe-based geometry videos," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 25, no. 1, pp. 51–62, 2015.

[21] W. Sun, et al., "Rate-constrained 3D surface estimation from noise-corrupted multiview depth videos," in *IEEE Trans. Image Processing*, Jul. 2014.

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

[23] R. Schnabel and R. Klein, "Octree-based point-cloud compression," in *Symposium on Point-Based Graphics*, Jul. 2006.

[24] Y. Huang, et al., "A generic scheme for progressive point cloud coding," *IEEE Trans. Vis. Comput. Graph.*, vol. 14, no. 2, pp. 440–453, 2008.

[25] C. Zhang, D. Florêncio, and C. Loop, "Point cloud attribute compression with graph transform," in *IEEE Int. Conf. Image Processing*, Sep. 2014.

[26] D. Thanou, P. A. Chou, and P. Frossard, "Graph-based motion estimation and compensation for dynamic 3D point cloud compression," in *IEEE Int. Conf. Image Processing*, Sep. 2015.

[27] D. Thanou, P. A. Chou, and P. Frossard, "Graph-based compression of dynamic 3D point cloud sequences," in *IEEE Trans. Image Processing*, to appear.

[28] J. Kammerl, et al., "Real-time compression of point cloud streams," in *IEEE Int. Conf. Robotics and Autom.*, May 2012.

[29] R. de Queiroz and P. Chou, "Compression of 3D point clouds using a region-adaptive hierarchical transform," *IEEE Trans. on Image Processing*, Vol. 25, pp. 3497-3956, Aug. 2016.

[30] R. de Queiroz and P. Chou, "Motion-compensated compression of dynamic voxelized point clouds," preprint.

[31] Y. Wang, et al, "Handling occlusion and large displacement through improved RGB-D scene flow estimation," *IEEE Trans. Circuits Syst. Video Techn.*, to appear.

[32] A. L. Yuille and N. M. Grzywacz, "A mathematical analysis of the motion coherence theory," *Int'l J. of Computer Vision*, vol. 3, no. 2, pp. 155-175, June 1989.

[33] S. Hadfield and R. Bowden, "Scene particles: Unregularized particle based scene flow estimation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 36, no. 3, pp. 564–576, 2014.

[34] J.-M. Gottfried, J. Fehr, and C. S. Garbe, "Computing range flow from multi-modal Kinect data," In *Advances in Visual Computing*, pages 758–767. Springer, 2011.

[35] E. Herbst, X. Ren, and D. Fox, "RGB-D flow: Dense 3-D motion estimation using color and depth," In *IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2013.

[36] J. Quiroga, F. Devernay, and J. L. Crowley, "Local/global scene flow estimation," In *IEEE Int'l Conf. on Image Processing (ICIP)*, 2013.

[37] X. Zhang, et al., "Dense scene flow based on depth and multi-channel bilateral filter," in Springer Computer Vision (ACCV), pp. 140–151, 2012.

[38] M. Hornacek, A. Fitzgibbon, and R. Carsten, "Sphereflow: 6 DoF scene flow from RGB-D pairs," in IEEE Int'l Conf. on Computer Vision and Pattern Recognition (CVPR), 2014.

[39] J. Quiroga, et al., "Dense semi-rigid scene flow estimation from RGBD images," in Springer Computer Vision (ECCV), pp. 567–582, 2014.

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

[41] M. Dou, et al., "3D Scanning Deformable Objects with a Single RGBD Sensor," in IEEE Int'l Conf. on Computer Vision and Pattern Recognition (CVPR), 2015.