Towards Real-Time and Efficient Compression of Human Time-Varying Meshes

Alexandros Doumanoglou, Dimitrios Alexiadis, Dimitrios Zarpalas, Petros Daras

***Abstract*—In this work, a novel skeleton-based approach to Human Time-Varying Mesh (TVM) compression is presented. The topic of TVM compression is new and has many challenges: handling the lack of obvious mapping of vertices across frames, handling the variable connectivity across frames, while maintaining efficiency with respect to the above are the most important ones. Very few works exist in the literature, while not all of the challenges have been addressed yet. Moreover, developing an efficient and real-time solution, handling the above, obviously is a difficult task. We attempt to address the Human TVM compression problem inspired from video coding by using different types of frames and trying to efficiently remove inter-frame geometric redundancy utilizing the recent advances in human skeleton tracking. The overall approach focuses on compression efficiency, low distortion and low computation time enabling for real time transmission of Human TVMs. It efficiently compresses geometry and vertex attributes of TVMs. Moreover, this work is the first to provide an efficient method for connectivity coding of TVMs, by introducing a modification to the state-of-the-art MPEG-4 TFAN [1] algorithm. Experiments are conducted in the MPEG-3DGC TVM database [2]. The method outperforms the state-of-the-art standardized static mesh coder MPEG-4 TFAN at low bit-rates, while remaining competent at high bit-rates. It gives a practical proof of concept that in the combined problem of geometry, connectivity and vertex attribute coding of TVMs, efficient inter-frame redundancy removal is possible, establishing ground for further improvements. Finally, this work proposes a method for motion-based coding of Human TVMs that can further enhance the overall experience when Human TVM compression is used in a Tele-Immersion (TI) scenario.**

***Index Terms*—Human capturing, time varying mesh, mesh compression, skeleton, entropy-constraint vector quantization, real-time, tele-immersion**

I. Introduction

DURING the previous years a lot of effort has been devoted in developing efficient compression algorithms for static meshes and dynamic meshes found in typical 3D animation sequences. Static mesh compressors target single individual meshes, while dynamic mesh compressors target animation sequences exploiting temporal redundancy between frames. Contrariwise to the synthetic animation sequences (dynamic meshes) created by 3D artists, reconstructed 3D natural scenes typically constitute Time-Varying Meshes (TVMs) [3]. The main difference between a dynamic mesh and a TVM is that the former consists of a series of meshes that have constant vertex and face count, as well as constant connectivity across frames, but variable geometry (vertex coordinates). On the other hand, TVM is described as sequence of meshes whose geometry, connectivity, vertex and face count are all varying across frames, introducing complexities that are not faced in dynamic mesh compression. The most important difference is that no one-to-one correspondence exists between vertices across frames and consequently the varying connectivity. In addition, since TVMs are meshes of live captured natural scenes that need to be compressed and streamed over the network in real-time, form a causal signal. On the contrary, in order to maximize exploitation of temporal redundancy across frames and maximize the compression ratio, dynamic meshes are typically treated as non causal signals.

In this work, we address geometry, connectivity and per vertex color compression of Human Time-Varying Meshes (H-TVMs), while remaining concerned about real-time performance and thus enabling real-time tele-immersion (TI) application scenarios [4], [5], [6]. Additional support for other vertex attributes apart from per vertex color (i.e. normals, texture coordinates, etc) is straightforward. The proposed scheme exploits recent advances in motion capturing that retrieves human skeleton pose in real-time from depth-maps [7]. The main contributions of this paper can be summarized to the following:

- It is the first work to provide an efficient method for connectivity coding in TVMs.
- It proposes a method that shows competent experimental results when compared to the current state-of-the-art static coders, especially in the low bit rates.
- It is the first to give a practical proof of concept that, when considering the combined problem of geometry, connectivity and per vertex attribute coding of TVMs, efficient exploitation of inter-frame redundancy is possible.
- It proposes an innovative method for motion-based coding based on human skeleton extraction.

Overall, the paper introduces a chain of innovative ideas that push research in TVM compression one step forward, aiming to establish ground for further improvements.

The rest of the paper is organized as follows: In Section II, previous related work in the area of mesh compression is discussed. In Section III, an overview of the proposed H-TVM compression framework is presented, while Sections III-C and IV contain in-depth analysis of the proposed compression scheme. Increasing further the effectiveness of a TI system, in Section V motion-based coding is discussed. Section VI follows with the presentation of the proposed algorithm's results, with respect to both computational time and compression ratio. In the same section, the state-of-the-art coder MPEG-

The authors are with the Information Technologies Institute, Centre for Research and Technology - Hellas, 6th km Charilaou - Thermi, GR-57001, Thessaloniki, Greece, Tel. +30 2310 464160 (ext. 277), Fax. +30 2310 464164 (e-mail: aldoum@iti.gr; dalexiad@iti.gr; zarpalas@iti.gr; daras@iti.gr)

4 TFAN [1] is compared against the proposed one. Finally, Section VII summarizes this work and discusses open issues and future work.

II. Related Work

Related work in the field of mesh compression can be categorized based on the nature of the mesh, being either static, dynamic or time varying.

A. Static Mesh Compression

The field of static mesh compression has been extensively explored during the previous two decades.

1) Connectivity Coding: The first attempt for a compact triangular representation was made by Deering [8]. Taubin et al. [9] introduced a mesh compression scheme based on vertex and triangle spanning trees to encode geometry and connectivity, respectively. Touma et al. [10] encoded mesh connectivity by applying entropy coding on the valence of mesh vertices, using a predefined traversal scheme resulting in practice 0.2 - 3.5 bits per vertex (bpv). A slightly more efficient approach than [10], was presented by Alliez et al. [11], based on the same valence-driven concept. Rossignac in [12], by providing both a rigorous theoretical analysis and an outstanding worst case upper bound on the connectivity compression, proposed a triangle conquest approach to encode triangle mesh connectivity. This algorithm, namely Edgebreaker, constitutes a state-of-the-art technique for triangular mesh connectivity coding. The initial method of [12] has a theoretical upper bound of 4 bpv. Since then, advances have been proposed that improve this theoretical upper bound [13], [14]. Later, Coors et al. [15] presented Delphi, a connectivity coder that improves EdgeBreaker by introducing geometry driven parallelogram connectivity prediction. A more recent work [1], which has been adopted by MPEG-4 standard for static mesh compression, exploits triangle fan representation for efficient connectivity coding presenting state-of-the-art results.

2) Geometry Coding: While connectivity coding is lossless, geometry is usually compressed in a lossy manner. A geometry coder typically aims to reduce coordinate precision for a better compression ratio. It is common for connectivity to drive geometry coding. In that case, usually, geometry data are encoded in a specific order determined by the way the connectivity is coded. Most geometry coders pre-quantize vertex positions, predict the quantized positions and entropy code prediction residuals. In the pre-quantization step of [10] and similar to most relevant coders, vertex coordinates are uniformly quantized with 8-16 bit precision. Following uniform quantization, the prediction step exploits the geometric correlation between adjacent vertices. Different geometry prediction schemes have been proposed in the literature ([8], [9], [10]), with parallelogram prediction [10] being the most common. Lee et al. [16] proposed Vector Quantization (VQ) for geometry coding reporting 6.7 bpv as opposed to [10] that achieves 9 bpv at the same resolution, while Karni et al. [17] proposed geometry coding by spectral compression. In a more recent work [18], connectivity and geometry are encoded simultaneously after segmenting the mesh in polygonal regions using principal plane analysis.

B. Progressive Mesh Compression

Progressive transmission permits rendering of a rough approximation of the model and progressively refines it as more data arrive to the receiver. Hoppe [19], [20], whose encoding algorithm was based on mesh simplification by edge contraction, was the first to introduce progressive mesh representation. Taubin et al. [21] improved [19] by proposing split operations to be performed in batches, while further improvements were later proposed by Pajarola et al. [22]. Alliez et al. [23] proposed valence driven decimation to guide progressive compression. The simplification process of the progressive mesh was based on vertex removal and hole retriangulation. Valete et al. [24] introduced a wavelet-based progressive compression scheme, while Peng et al. [25] proposed a progressive mesh encoding algorithm based on octree decomposition. Later, Valette et al. [26] redefined progressive mesh compression as a mesh generation problem. The results presented in their work show superior quality compared to either [24] or [27].

C. Dynamic Mesh Compression

In dynamic mesh compression, with connectivity being constant across frames, the focus is on efficient geometry compression. Dynamic mesh compression algorithms try to exploit temporal geometric redundancy of vertex trajectories across frames. Ibaria et al. [28] used a spatio-temporal prediction scheme to compress animated mesh sequences. The algorithm of Karni et al. [29] was based on Principal Component Analysis (PCA) and linear prediction coding (LPC). In [30], PCA is applied on the trajectory of vertices and each vertex's trajectory is expressed in the PCA space. The prediction residuals and the PCA basis functions are then encoded. The same authors proposed COBRA [31] as a method for efficiently compressing the PCA basis. Payan et al. [32] tackled the problem using wavelets. Later, Mamou et al. [33] proposed a method consisting of a piecewise affine predictor coupled with a skinning model and a temporal Discrete Cosine Transform (DCT) representation of the residual errors. Their work with some improvements, was standardized under MPEG-4 Frame-Based Animated Mesh Compression (FAMC) [34]. The proposals included in [34] as well as the combination of [30] and [31] are considered to be today's state-of-the-art.

D. Geometry Images and Videos

Based on the interesting work of [35], an arbitrary manifold static mesh can be remeshed into a completely regular mesh structure, called "geometry image". A geometry image effectively captures geometry (vertex coordinates) information into a 2D square array of $[X, Y, Z]$ values, while connectivity is implicitly defined. Consequently, geometry images can be encoded using standard image compression techniques.

As an extension to geometry images, the concept of "geometry videos" [36] was introduced by Briceno et al. In [36], an animated (i.e. dynamic) mesh sequence is treated as a video, offering the possibility to apply traditional video compression techniques. Taking into consideration the isometric nature of

3D articulated motions, "conformal geometry videos" were recently proposed in [37] by Quynh et al., in order to efficiently represent articulated motions, such as human motion.

Since geometry images and their extensions can be compressed using mature ideas from standard image/video coding, their compression can be obviously very efficient. However, geometry images and videos themselves constitute actually a remeshing process, rather than a compression scheme. Additionally, since the generation of geometry videos results into a constant number of vertices and connectivity, coding of geometry videos lies actually in the field of dynamic mesh compression. Geometry images and consequently their video counterparts, can represent only manifold input 3D surfaces [35], which is not the usual case in real-world 3D captures. Finally, the parametrization (remeshing) process requires significant computational effort. Thus, geometry video concepts are not applicable in scenarios that involve compression of TVMs, especially considering real-time constraints.

E. Time-Varying Mesh Compression

To the best of our knowledge the first attempts to compress Time-Varying Meshes was described in [38] and [39]. In [39], Gupta et al. used the Iterative Closest Point (ICP) [40] algorithm to exploit temporal geometric redundancy across frames. This scheme, uses I and P frames, in a sense similar to traditional MPEG-2 Video Coding. This method was the first to deal with TVM connectivity coding.

Block matching is a common technique used in 2D Video for motion compensation (MC). Concerning geometry compression in TVMs, Han et al. in [3] extended this concept in 3D space, forming the Extended Block Matching Algorithm (EBMA). After MC the residuals are transformed using DCT, uniformly quantized and entropy coded. Motion vectors are encoded as well. In their method, additional information, such as vertex indices and block sizes, is needed in order to decode mesh geometry. The authors chose to use simple entropy coding to compress this kind of information. Later, the same authors, used coarse and fine levels of quantization [41] and were able to improve geometry compression of TVMs as opposed to [3]. Yamasaki et al. [42] proposed patch-based compression of TVMs showing significant improvement among previous attempts. Additionally, they handle per vertex color and encode triangle connectivity by re-indexing triangle vertices.

In the context of real-time compression of TVMs being used in a TI system, Lien et al. [43] used a skeleton-based approach for compressing geometry of H-TVMs. In their innovative work, apart from exploiting skeletal motion data for efficient prediction, they also employ traditional image compression techniques to encode the prediction residuals.

F. Discussion

Following the literature overview, in this subsection we highlight the gaps of existing methodologies in TVM compression that are addressed in the proposed work. In [3], the problem is addressed only in terms of point-cloud compression. No connectivity coding is considered. Moreover, side-information is proven to be very expensive ($\sim$ 6 bpv), as later was admitted in [42]. Contrariwise, in this paper we discuss all side-information aspects and give a theoretical proof for its encoding cost. In the proposed method this side-information is reasonably small ($\sim$ 2 bpv). Similarly, in [41] only point-cloud compression is considered, while in [42] a very expensive approach for connectivity compression is employed that requires approximately 55 bpv. Furthermore, to the best of our knowledge, no appropriate connectivity coding approach has been proposed that can be combined with the previous geometry coding methods, in order to form an actual mesh (geometry and connectivity) encoder. All the above mentioned works could not efficiently deal with mesh connectivity in the combined problem of TVM geometry and connectivity coding, thus inevitably their performance is inferior when compared to the state-of-the-art static coders. The need for developing efficient connectivity coding algorithms for TVMs has been admitted by the authors in [42]. The hereby work proposes a solution to the TVM connectivity coding problem, by modifying the existing state-of-the-art TFAN [1] algorithm.

The method of [43], while utilizing a skeleton-based approach like the proposed one, does not deal with mesh connectivity. On top of that, it is not using a bijective projection function to encode geometry residual and consequently is highly sensitive to skeleton errors. The proposed method, as verified in Section VI, is robust to skeleton errors and does not compromise visual quality.

The methods of [38] and [39], are indeed tackling the combined problem of geometry and connectivity coding. However, they rely on assumptions about the inter-frame variation of vertex count (requiring 1%-2% inter-frame variation), as well as for the reconstruction method used to produce the meshes (delaunay triangulation). The proposed method is not restricted by such constraints.

Finally, among the TVMs compression methods reported in the literature, only the method of [43] is concerned with real-time performance in a TI application scenario, similarly to our requirements.

III. Proposed System Overview

In this section, a high level overview of the proposed system is given. The system aims to efficiently compress H-TVMs and has been put into practice with the multi-sensor Microsoft Kinect setup of [44], [45].

A. Requirements - Skeleton Motion Tracking

The system takes advantage of recent technological advances in motion capturing that allow for real-time extraction of human skeleton from depth-maps. An enhanced version of the work presented in [7] is already implemented in Microsoft Kinect SDK, while Primesense's OpenNI SDK contains an equivalent proprietary algorithm (NITE). In addition, a method to extract skeleton information from TVMs appears in [46]. In this work, the OpenNI skeleton tracking module is used to extract the skeleton motion information, since studying the problem of skeleton extraction is beyond the scope of this paper. The field of skeleton extraction from 3D data is expected to become even maturer in the near future, thus offering a valuable and reliable tool.

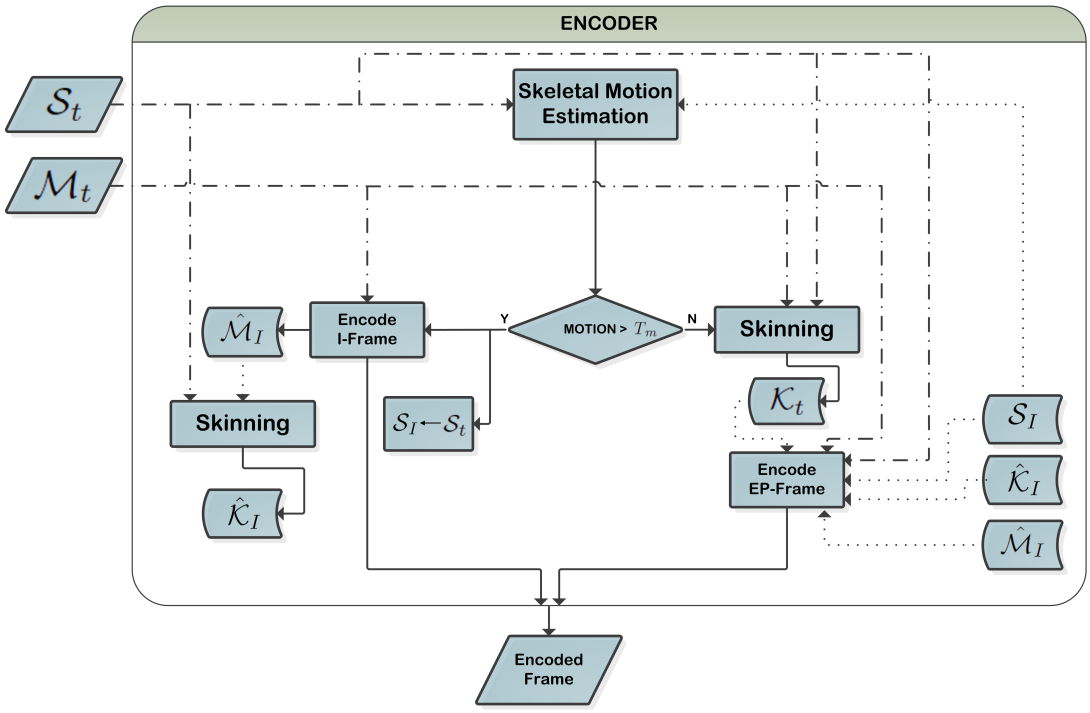


Fig. 1: *Proposed system's encoder. Dashed lines denote external input to the encoder, while dotted lines denote data input to internal subsystems from internal variables.*

B. Core analysis

Let $\mathcal{M}_t$ denote the triangulated human mesh to be encoded in frame F_t, while $\mathcal{S}_t$ its respective skeleton information as , $t \in \mathbb{N}$. Skeleton information includes joint positions and bone orientations.

The input to our framework consists of $\mathcal{M}_t$ and $\mathcal{S}_t$. The output, inspired from animation sequence compression [34], as well as traditional video compression methods, consists of 3 types of frames: I Frames, P Frames and Enhanced-P (EP) Frames. I-Frames are intra-coded, i.e. they can be reconstructed without any reference to other frames. P-frames are forward predicted from the last encoded I-Frame using vertex transformations and taking advantage of skeleton information. A P-Frame is reconstructed by animating the last encoded I-Frame using rigid skinning. EP-Frames are similar to P-Frames, with the addition of carrying prediction corrections for improved mesh accuracy and quality. Hence, P-Frames can be treated as a special case of EP-Frames, as discussed in subsection IV-B. The compressed sequence consists of I, P and EP frames, alternating at non constant intervals.

Fig. 1 provides a brief overview of the encoder. The first frame is always encoded as I-Frame. Next frames are encoded either as P, EP or I, depending on the overall motion obtained from skeleton information. When the "skeletal motion" is lower than a specified threshold, the frame is encoded as a P frame. For a larger "skeletal motion", the frame is encoded either as EP or I Frame, with I-Frames being favored for the very large "skeletal motions", considering appropriate thresholds. P and EP-Frames are encoded with respect to F_I, i.e. the last encoded I-Frame. Thus, the "skeletal motion" of F_t is calculated with respect to F_I. The "skeletal motion" metric used in this work is defined as the sum of all "per-bone motions". The "per-bone motion" is calculated as the shift (euclidean distance) of the specific bone's center from frame F_I to frame F_t.

The encoded mesh at time t, being either an I or EP Frame, is denoted as $\hat{\mathcal{M}}_t$. Let also $\hat{\mathcal{M}}_I$ and $\mathcal{S}_I$ denote the decoded mesh of F_I and its corresponding skeleton information, respectively. The encoder initially performs a skinning process, associating each vertex and triangle of $\mathcal{M}_t$ to a skeleton bone of $\mathcal{S}_t$. We denote the output of the skinning process for frame F_t as $\mathcal{K}_t$ and for F_I as $\hat{\mathcal{K}}_I$. The encoding of EP-Frames relies on $\mathcal{S}_t$, $\mathcal{K}_t$, $\mathcal{M}_t$, $\mathcal{S}_I$, $\hat{\mathcal{K}}_I$ and $\hat{\mathcal{M}}_I$ in order to exploit temporal geometric redundancy across frames and encode prediction corrections along with the mesh connectivity.

In the following sections in-depth information for each separate module of Fig. 1 is provided. For the reader's convenience, the most commonly used notations in this paper are depicted in Table I.

C. I-Frame Coding

I-Frames are intra-coded and theoretically any static mesh encoder can be used for their compression. Depending on usage scenario, either a highly efficient static mesh compression algorithm (like [1]) can be used, or instead a low-complexity real-time algorithm like [47] can be preferred. The rest of the proposed system is not affected by this choice.

F_t	The frame to be encoded in time t.
F_I	The last encoded I-Frame
$\mathcal{M}_t$	The mesh to be encoded in frame F_t.
$\mathcal{S}_t$	The skeleton of the mesh in frame F_t.
$\mathbb{V}_t$	The set of vertices contained in $\mathcal{M}_t$.
N_t	The number of vertices in $\mathcal{M}_t$.
$\mathbf{v}_t^k$	The k-th vertex of $\mathcal{M}_t$.
$\hat{\mathcal{M}}_I$	The decoded mesh of the last encoded I-Frame F_I.
$\hat{\mathcal{S}}_I$	The skeleton of the mesh in frame F_I.
$\mathbb{V}_I$	The set of vertices contained in $\hat{\mathcal{M}}_I$.
N_I	The number of vertices in $\hat{\mathcal{M}}_I$.
$\mathbf{v}_I^k$	The k-th vertex of $\hat{\mathcal{M}}_I$.
$\mathcal{M}_P$	The predicted mesh for frame F_t.
$\mathbb{V}_P$	The set of vertices contained in $\mathcal{M}_P$.
N_P	The number of vertices in $\mathcal{M}_P$.
$\mathbf{v}_P^k$	The k-th vertex of $\mathcal{M}_P$.
b_t^i	The i-th bone in skeleton $\mathcal{S}_t$.
$\mathbf{j}_t^{(i,0)}$	The first joint of b_t^i.
$\mathbf{j}_t^{(i,1)}$	The second joint of b_t^i.
$\mathbb{P}(b_t^i)$	The set of vertices in $\mathcal{M}_t$ that can be orthogonally projected on bone b_t^i.
$\mathbb{A}(b_t^i)$	The set of vertices in $\mathcal{M}_t$ that are associated with bone b_t^i.
$\mathbf{e}_t^k$	The k-th prediction residual when encoding $\mathcal{M}_t$.
$\mathbf{e}_{(k_p)}^{k_t}$	The k_t-th prediction residual when encoding $\mathcal{M}_t$, whose reference vertex in $\mathcal{M}_P$ is $\mathbf{v}_P^{k_P}$.
$\hat{\mathbf{v}}_t^k$	The k-th decoded vertex when decoding frame F_t.

TABLE I: *Notation table*

IV. Enhanced-P Frame Coding

In contrast to I-Frames, which are intra-coded for both geometry and connectivity, EP-Frames are inter-frame coded with respect to the last encoded I-Frame for geometry and intra-coded for connectivity. Encoding EP-Frames is a three step process: 1) predict geometry, 2) find and encode geometric prediction errors and finally, 3) encode connectivity. While geometry coding can be lossy, connectivity coding shall not.

An overview of the EP-Frames encoding algorithm is depicted in Fig. 2. The process of encoding $\mathcal{M}_t$ starts by predicting its geometry from $\hat{\mathcal{M}}_I$ through per bone vertex transformations by utilizing skeletal animation and the Iterative Closest Points (ICP) algorithm. To accomplish per bone vertex transformations, mesh vertices are required to be associated with skeleton bones. Thus, a mesh skinning process is employed for both $\hat{\mathcal{M}}_I$ and $\mathcal{M}_t$. Geometry prediction residuals are calculated afterwards in a per bone basis. The prediction residuals are encoded with respect to their nearest predicted vertex. Finally, connectivity is encoded using a modification of TFAN [1], which allows geometry to be encoded independently from connectivity (and thus exploiting inter-frame redundancy) as opposed to the original algorithm. In the following subsections the steps of EP-Frame coding are discussed in-depth.

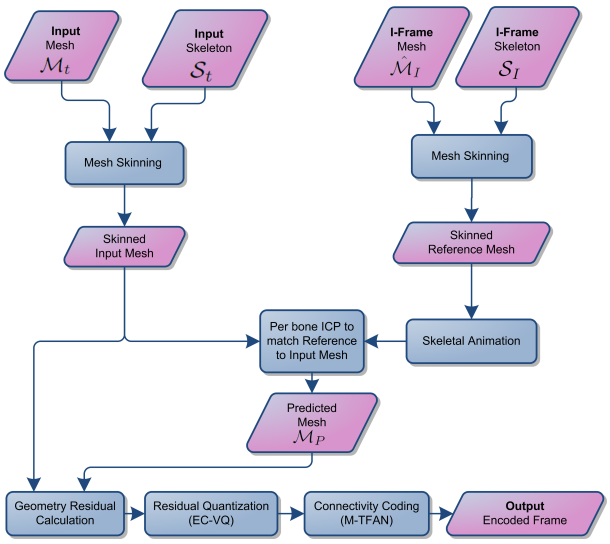


Fig. 2: *Encoding of EP-Frames algorithm.*

A. Mesh Skinning

For geometry coding in EP-Frames, the prediction scheme used by the proposed encoder employs a skinning process in order to associate the mesh's vertices to bones. In general, the skinning module approximates the human mesh with bone cylinders whose axis are the skeleton bones themselves. To accomplish the skinning task as accurately as possible, even when the subject (human) stands in an arbitrary pose, an offline skinning calibration process is required. During the skinning calibration process the subject is required to stand in a T-pose (as in Fig. 3) and the radii of the cylinders are estimated. Later, when the user stands in an arbitrary pose, the mesh skinning is achieved by associating vertices to their closest bone, depending on their distance to the bone's cylinder surface. In the rest of the section we formalize this concept using mathematical notation.

In the mesh skinning module of the proposed system, depicted in Fig. 2, the input consists of $\mathcal{S}_t$ and $\mathcal{M}_t$. For notation simplicity, we drop the time parameter t in the rest of this section. Considering the typical skeleton in T pose of Fig. 3, each skeleton bone is represented using a vector. Each bone is given an ID and is referenced as b^i, $i \in \mathbb{B} := \{0, 1, \ldots, B-1\}$, where B denotes the number of bones. Without loss of generality, in Fig. 3 and in the rest of the paper for simplicity we use $B = 14$. Each bone b^i consists of a line segment l^i that is formed by the two skeleton joints, denoted as $\mathbf{j}^{(i,0)}$ and $\mathbf{j}^{(i,1)}$, where $\mathbf{j}^{(i,0)}$ refers to the joint at the beginning of the bone vector and $\mathbf{j}^{(i,1)}$ refers to the joint at the end of the same vector. Additionally, the radii of the bone cylinders are denoted as r^i.

Let $\mathcal{M}$ consist of N vertices, each one denoted as $\mathbf{v}^k$, $k = 0, 1, \ldots, N-1$. The skinning process associates each vertex $\mathbf{v}^k$ with a skeleton bone b^n, when

$$\text{dist}(\mathbf{v}^k, b^n) < \text{dist}(\mathbf{v}^k, b^i) \quad \forall\, i \in \mathbb{B} - \{n\}, \tag{1}$$

namely it associates a vertex to its closest bone. The distance

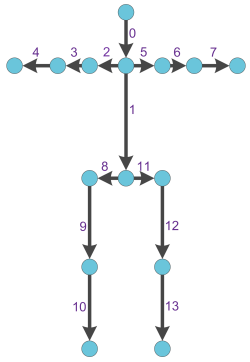


Fig. 3: *Typical skeleton in T pose. Bones are numbered from 0 to 13.*

of a vertex $\mathbf{v}^k$ to a bone b^i is given from:

$$\text{dist}(\mathbf{v}^k, b^i) \coloneqq \begin{cases} \text{dist}(\mathbf{v}^k, l^i) - r^i, \text{ when } \mathbf{v}^k \in \mathbb{P}(b^i) \\ \\ \min(\text{dist}(\mathbf{v}^k, \mathbf{j}^{(i,0)}), \text{dist}(\mathbf{v}^k, \mathbf{j}^{(i,1)})), \text{ otherwise.} \end{cases} \tag{2}$$

where $\mathbb{P}(b^i) \coloneqq \{\mathbf{v}^k : \mathbf{v}^k \text{ projects to } l^i\}$ denotes the set of vertices that project to the line segment l_i, considering typical orthogonal projection. Such a vertex-to-bone association is denoted as $\mathbf{v}^k \rightarrow b^n$.

The radius parameter r^i is constant across frames (invariant to t) and is calculated only once for each bone through the skinning calibration process. In order to calculate r^i, an association between vertices of human mesh in T-pose and its respective skeleton is made, according to the rule of (1) but with $\text{dist}(\mathbf{v}^k, b^i)$ defined as:

$$\text{dist}(\mathbf{v}^k, b^i) \coloneqq \text{dist}(\mathbf{v}^k, l^i), \mathbf{v}^k \in \mathbb{P}(b^i). \tag{3}$$

In the skinning calibration process, from (3) it is possible that some vertices are not associated with any bone, due to the fact that they do not project on any of l^i, $i \in \mathbb{B}$. These vertices are simply ignored. Let $\mathbb{A}(b^i) \coloneqq \{\mathbf{v}^k : \mathbf{v}^k \rightarrow b^i\}$ denote the set of vertices associated with bone b^i. Then, $r^i = \max_k(\{\text{dist}(\mathbf{v}^k, b^i) : \mathbf{v}^k \in \mathbb{A}(b^i)\})$, i.e. r^i is equal to the maximum distance of the associated vertices to the bone and constitutes the cylinder radius of bone b^i. Notice that in eq. (2), $\text{dist}(\mathbf{v}^k, b^i)$ might be negative for some vertices $\mathbf{v}^k$, meaning that those vertices lie inside the cylinder of b^i.

B. Geometry Prediction

In the proposed method, skeleton information is used in a straightforward manner, influenced by skeletal animation used in traditional computer graphics. Let $\mathcal{M}_P$ denote the predicted mesh for frame F_t. The predicted mesh is initialized using skeletal rigid animation of $\hat{\mathcal{M}}_I$ using $\mathcal{S}_I$ and $\mathcal{S}_t$. Let $\mathbf{T}_t^i$ denote the 3×3 orientation matrix of bone b_t^i at frame F_t, which is in fact a rotation matrix. Let also $\mathbf{j}_t^{(i,0)}$ denote the first joint of b_t^i. Similarly, let $\mathbf{T}_I^i$, b_I^i and $\mathbf{j}_I^{(i,0)}$ be the respective symbols for frame F_I. Furthermore, let $\mathbb{V}_I \coloneqq \{\mathbf{v}_I^k \,|\, k = 0, 1, \ldots, N_I - 1\}$, denote the set of vertices in $\hat{\mathcal{M}}_I$, where N_I denotes the number of vertices in $\hat{\mathcal{M}}_I$. Similarly, in the rest of the paper, for the mesh $\mathcal{M}_P$ we use N_P, $\mathbf{v}_P^k$ and $\mathbb{V}_P$. It is obvious that $N_P = N_I$, since $\mathcal{M}_P$ is a transformed version of $\hat{\mathcal{M}}_I$. The set of predicted vertices $\mathbb{V}_P$ is initially calculated using the following rule:

$$\mathbf{v}_P^k \coloneqq \mathbf{T}_t^i(\mathbf{T}_I^i)^{-1}(\mathbf{v}_I^k - \mathbf{j}_I^{(i,0)}) + \mathbf{j}_t^{(i,0)} \quad , \forall \mathbf{v}_I^k \in \mathbb{A}(b_I^i) \tag{4}$$

where $\mathbb{A}(b_I^i)$ is the set of vertices in I-Frame associated with the i-th bone.

Equation (4) is the typical skeletal animation rule used in computer graphics, where $\mathbf{j}^{(i,0)}$ serves as the animation's rotation point. To further enhance the prediction obtained through skeletal animation, a point-to-point ICP [40] algorithm is applied on a per-bone basis. ICP is applied between the source point-set $\mathbb{A}(b_t^i)$ and the target point-set $\mathbb{A}(b_P^i)\, \forall i \in \mathbb{B}$. The set $\mathbb{A}(b_P^i)$ consists of the vertices of $\mathbb{A}(b_I^i)$ transformed by (4). ICP outputs a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{U}$ that can be used to align $\mathbb{A}(b_t^i)$ to $\mathbb{A}(b_P^i)$. While decoding, the alignment needs to be done in reverse order, i.e. aligning $\mathbb{A}(b_P^i)$ to $\mathbb{A}(b_t^i)$. Thus, the rotation matrix $\mathbf{R}' \coloneqq \mathbf{R}^{\mathrm{T}}$ and the translation vector $\mathbf{U}' \coloneqq \mathbf{R}^{\mathrm{T}}\mathbf{U}$ are used in the decoding process using the formula:

$$\mathbf{v}_P^k \coloneqq \mathbf{R}'\mathbf{v}_P^k + \mathbf{U}' \quad , \forall\, \mathbf{v}_P^k \in \mathbb{V}_P \tag{5}$$

giving the final predicted mesh $\mathcal{M}_P$ for frame F_t. The vertex $\mathbf{v}_P^k$ on the right hand side of (5) is the one obtained by (4).

Connectivity and skinning information are preserved between $\hat{\mathcal{M}}_I$ and $\mathcal{M}_P$. When decoding an EP-Frame, in order to apply prediction, the decoder needs to have access to skeleton information of both $\mathcal{S}_I$ and $\mathcal{S}_t$. Therefore, the encoder embeds the respective skeleton information to I-Frames, whereas for EP-Frames it embeds one transformation for each bone, being the combination of (4) and (5). P-Frames, being a special case of EP Frames, only carry per bone transformation information that is used to animate $\hat{\mathcal{M}}_I$ in the decoder and they do not carry any geometry prediction residuals.

C. Encoding Geometry Prediction Errors

In TVMs, the temporal correlation of vertex coordinates is usually high. Thus, exploiting the temporal redundancy by inter-frame coding of mesh geometry is beneficial. This observation is the leading concept behind encoding the geometry in EP-Frames. In EP-Frames, the geometry of $\mathcal{M}_t$ is encoded with respect to $\mathcal{M}_P$.

1) Calculating Prediction Residuals: Let each vertex of $\mathcal{M}_t$ be denoted as $\mathbf{v}_t^k$, $k = 0, 1, \ldots, N_t - 1$, where N_t is the number of vertices in $\mathcal{M}_t$. Let also $\mathbb{V}_t$ denote the set of vertices of $\mathcal{M}_t$. We use the same notation introduced since the beginning of the paper, replacing t with P for concepts related to mesh $\mathcal{M}_P$. Each vertex $\mathbf{v}_t^{k_t} \in \mathbb{V}_t$ can be expressed with respect to an appropriate vertex $\mathbf{v}_P^{k_P} \in \mathbb{V}_P$ as

$$\mathbf{v}_t^{k_t} = \mathbf{v}_P^{k_P} + \mathbf{e}_t^{k_t}. \tag{6}$$

The objective for efficient geometry coding is to minimize prediction errors $|\mathbf{e}_t^{k_t}|$. Thus, an appropriate reference vertex $\mathbf{v}_P^{k_P}$ in $\mathcal{M}_P$ needs to be found for each vertex $\mathbf{v}_t^{k_t}$ in the

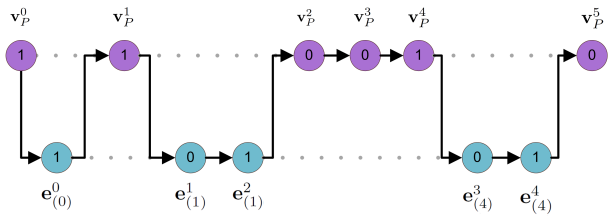


Fig. 4: *Graph mapping between* $\mathbb{V}_P^{ord}$ *and* $\mathbb{E}^{ord}$

original mesh $\mathcal{M}_t$. The per-bone ICP algorithm, described in section IV-B, serves for both minimizing the prediction errors under a rigid transform constraint, as well as finding appropriate reference vertices using the closest-point correspondences established during the last iteration between the source and target point-sets. The reference vertices are searched among the vertices of ICP's target point-set. ICP does not guarantee that the vertices in the target point-set will have a unique correspondence (or even any) in the source point-set. In practice, it is usual that reference vertices of the target point-set may have none or multiple correspondent vertices in the source point-set. Having vertex correspondences calculated from ICP, the prediction error vectors $\mathbf{e}_t^{k_t}$, called residuals, can be computed by (6). Using the correspondence mapping obtained through ICP, an equivalent equation of (6) is also used to calculate residuals for color and any other attributes associated with each vertex, such as normals or texture coordinates.

2) Encoding Prediction Residuals: Let $\mathbb{V}_I^{\text{ord}}$ denote the set of vertices $\mathbb{V}_I$ with its elements ordered in the same order as they appear in the last encoded I-Frame. Let also $\mathbb{V}_P^{\text{ord}}$ denote the set of vertices $\mathbb{V}_P$ with its elements ordered as their respective elements of $\mathbb{V}_I^{\text{ord}}$. The mapping between the elements of $\mathbb{V}_P$ and $\mathbb{V}_I$ is obvious from (4). From now on, we use the notation $\mathbf{e}_{(k_P)}^{k_t}$ to refer to the residual $\mathbf{e}_t^{k_t}$ that when added to $\mathbf{v}_P^{k_P}$ results the vertex $\mathbf{v}_t^{k_t}$, as in (6). Let $\mathbb{E} := \{\mathbf{e}_{(k_P)}^{k_t} : k_t = 0, 1, \ldots, N_t - 1\}$ be the set containing the prediction residuals. The ordered set $\mathbb{E}^{\text{ord}}$ is the set containing the elements of $\mathbb{E}$ in an ascending order with respect to k_P. The set $\mathbb{E}^{\text{ord}}$ is quantized using Entropy-Constrained Vector Quantization (ECVQ) [48] and then entropy coded. Encoding of vertex attributes follow the same procedure. The quantized vertex residuals may produce duplicate vertices when decoded. Thus, the residuals that are responsible for the appearance of duplicate vertices in the decoded mesh, are removed prior to entropy compression, while triangle indices are re-indexed to keep the topology intact.

3) Encoding the residual to reference vertex mapping: Up to this point, the decoding process would be straightforward except that the reference vertex $\mathbf{v}_P^{k_P} \in \mathbb{V}_P^{\text{ord}}$ for each residual $\mathbf{e} \in \mathbb{E}^{\text{ord}}$ in unknown to the decoder. Thus, a mapping between the reference vertices of $\mathbb{V}_P^{\text{ord}}$ and the residuals of $\mathbb{E}^{\text{ord}}$ needs to be encoded.

We consider the directed graph of Fig. 4, which consists of a single path. The purple nodes in the top row map in exact order to the vertices of $\mathbb{V}_P^{\text{ord}}$, while the cyan nodes in the bottom row map in exact order to the residuals of $\mathbb{E}^{\text{ord}}$. Successful decoding can be achieved by traversing the graph as shown in the figure and adding each residual $\mathbf{e}_t^{k_t}$ to the last visited vertex $\mathbf{v}_P^{k_P}$. This way the graph compactly represents the reference vertices for all residuals, as well as all vertices $\mathbf{v}_P^{k_P} \nexists \mathbf{e}_{(k_P)}^{k_t} \in \mathbb{E}^{\text{ord}}$.

Let the graph consist of nodes n_q, $q = 0, 1, \ldots, (N_I + N_t - 1)$, with q increasing along the path. To encode the graph, we assign to each node n_q one bit $c_q \in \{0, 1\}$, with $c_q = 0$ when n_{q+1} is a node in the same row as n_q and $c_q = 1$ otherwise. Let $\mathbb{M}^{\text{ord}} := \{c_q,\, q = 0, 1, \ldots, (N_I + N_t - 1)\}$ be the set whose elements are placed in ascending order with respect to q. It is obvious that the encoding of $\mathbb{M}^{\text{ord}}$ has an upper bound rate of

$$R = \frac{N_I}{N_t} + 1 \tag{7}$$

bpv, with respect to frame F_t. When $N_t \simeq N_I$, which is rather the usual case, (7) gives an upper bound of 2 bpv for the graph coding. In practice, slightly better rates may be achieved by applying run-length coding to the resulting bit sequence $\mathbb{M}^{\text{ord}}$. The graph of Fig. 4 would be run-length encoded as $\{3, 1, 1, 2, 1, 1, 1, 1\}$. Entropy coding is then applied to the resulting sequence.

D. Encoding Connectivity

Concerning TVM connectivity coding, there are two candidate directions. The first direction is the one already taken for geometry, i.e. inter-frame coding, while the second one is intra-coding. The decision taken in this work was driven by the observation of the following two key concepts. Firstly, it seems rather difficult to successfully exploit temporal redundancy in connectivity of a mesh sequence under the lack of objective mapping between meshes' vertices across frames. This concept is supported by the experimental results of [39]. Secondly, state-of-the-art static connectivity coders already show exceptional results and the major bottleneck in static mesh compression is geometry coding. Given these observations, intra-coding connectivity seems rather the reasonable solution.

In [42], connectivity is encoded by outputting 3 vertex indices for each triangle of the mesh. As admitted by its authors [42], this approach is inefficient, since the cost is approximately $3\log_2(N_t)$ bpv. For a more efficient solution, static connectivity coding approaches can be used. However, there is one major issue that needs to be addressed when attempting intra-coding connectivity with existing static connectivity coders under geometry inter-frame coding, regarding the order in which the vertex residuals are encoded.

Mesh connectivity, as described in Section II for static meshes, is often used to guide geometry compression commonly by employing parallelogram prediction. For efficient coding, the connectivity coder's vertex/triangle traversal algorithm determines the order in which geometry prediction errors are encoded, in order to avoid vertex indexing overhead. In that case, the mapping between connectivity and its respective geometry is made by suitably ordering prediction residuals according to the decoder's connectivity traversal algorithm. Conclusively, in static mesh coding geometry coding is coupled with connectivity coding, while the mapping between geometry and connectivity is implied by the placement order of the prediction residuals.

In TVM compression it is crucial that geometry and connectivity are encoded independently, while maintaining the mapping between the two. Using static connectivity coders in TVM compression triggers the issue of vertex re-ordering. Re-ordering geometry prediction residuals according to connectivity traversal algorithm, as a means of low cost mapping between connectivity and geometry, would probably make the mapping between the residuals and their reference vertices rather expensive (in terms of bit rate). Thus, a solution for referencing vertices to encode connectivity is required, since vertex re-ordering is not an option.

The obvious way to reference geometry while encoding connectivity is to output the indices of each unvisited vertex during the traversal of the mesh. Vertex indexing can be combined with most connectivity coders available, such as [12], [1] and [10]. However, this is an inefficient approach that roughly requires at least $\log_2(N_t)$ bpv just for referencing vertices.

In this work, we address the TVM connectivity coding problem by modifying the state-of-the-art connectivity coder TFAN [1]. TFAN was chosen because of its applicability to manifold as well as non-manifold meshes, its simplicity and its high performance. However, the concepts introduced can be easily incorporated into other static connectivity coders as well. The modification attempts to address the issue of mapping connectivity to geometry via a geometric distance order metric, avoiding the huge indexing overhead of the straightforward solution. In the rest of the section the proposed connectivity scheme, called Modified-TFAN (M-TFAN), is presented in detail. The interested reader is encouraged to refer to [1] for more details on the original TFAN coder.

M-TFAN: The challenge of M-TFAN is to map connectivity to geometry efficiently. Except for the modification dealing with this part, M-TFAN is identical to [1]. Thus, in the rest of the section only the mapping between geometry and connectivity is discussed.

As in [1], a Triangle Fan (TF) of degree d is an ordered set of d triangles, each one denoted as $T_j, j \in \{0, 1, \ldots, d-1\}$ and defined by a sequence of $d+2$ vertices $(v_0, v_1, \ldots, v_{d+1})$ such that $\forall j \in \{0, \ldots, d-1\}, T_j = \{v_0, v_{j+1}, v_{j+2}\}$. The vertex v_0 is so-called the center of the TF.

Simplifying the notation of [1], the traversal of a single TF centered at v_0 is considered. The algorithm treats iteratively all the vertices of the TF. When treating an unvisited vertex v_n of the TF, a symbol $s(n) = 1$ is emitted, indicating that v_n is unvisited. Additionally, a reference to the underlying geometry of v_n is given, as described in the next paragraphs.

In [10], the Euler's formula is used to conclude that the average valence number of a vertex in a mesh equals 6. This means that an arbitrary vertex v participates, on average, in the formation of 6 triangles. We intuitively propose the rather reasonable assumption that those 6 triangles involve the 6 nearest neighbor vertices of v with extremely high probability. This is the key concept guiding the design of M-TFAN. When traversing a TF centered at v_0, the m nearest neighbors of TF's center are ordered with respect to their distance from v_0. Then, when treating an unvisited vertex v_n of the TF, the index of v_n in the ordered set of m nearest neighbors of v_0 is emitted as a means to map connectivity to its underlying geometry. The case of v_n not belonging to the set of m nearest neighbors of v_0 is treated specially. In the rest of the section we formalize this concept in a more precise manner using mathematical notation.

Following Section IV-C, during decoding, each residual $\mathbf{e}_t^k \in \mathbb{E}^{\text{ord}}$ produces a vector $\hat{\mathbf{v}}_t^k$. Let $\hat{\mathbb{V}}_t^{\text{ord}} := \{\hat{\mathbf{v}}_t^k \,|\, k = 0, 1, \ldots, N_t - 1\}$ be the ordered set containing the vertices of the decoded mesh with its elements in the same order as their respective residuals in $\mathbb{E}^{\text{ord}}$. Consider a specific vertex $\hat{\mathbf{v}} \in \hat{\mathbb{V}}_t^{\text{ord}}$ and let $\mathbb{NV}^{\text{ord}}(\hat{\mathbf{v}}, m)$ denote the ordered set of its m nearest neighbor vertices over $\hat{\mathbb{V}}_t^{\text{ord}}$. The m neighbors of $\hat{\mathbf{v}}$ are denoted as $\hat{\mathbf{x}}_0, \hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_i, \ldots, \hat{\mathbf{x}}_{m-1}$, $i \in \{0, 1, \ldots, m-1\}, \hat{\mathbf{x}}_i \in \hat{\mathbb{V}}_t^{\text{ord}}$. The set is ordered based on neighbor vertices' euclidean distance from $\hat{\mathbf{v}}$, i.e. $\mathbb{NV}^{\text{ord}}(\hat{\mathbf{v}}, m) := \{\hat{\mathbf{x}}_0, \hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_i, \ldots \hat{\mathbf{x}}_{m-1}\}$, such that $\text{dist}(\hat{\mathbf{v}}, \hat{\mathbf{x}}_i) \leq \text{dist}(\hat{\mathbf{v}}, \hat{\mathbf{x}}_{i+1})$. Let also $\text{dorder}^{(m)}(\hat{\mathbf{v}}, \hat{\mathbf{x}}_i) := \sum_{j=0}^{i-1} s_j$ denote the distance order of $\hat{\mathbf{x}}_i$ with respect to $\hat{\mathbf{v}}$ over $\mathbb{NV}^{\text{ord}}(\hat{\mathbf{v}}, m)$, where $s_j = 1$ if the vertex $\hat{\mathbf{x}}_j$ is unvisited and $s_j = 0$ otherwise. M-TFAN outputs $\text{dorder}^{(m)}(v_0, v_n)$ each time a vertex v_n is visited for the first time while traversing the TF whose center is v_0.

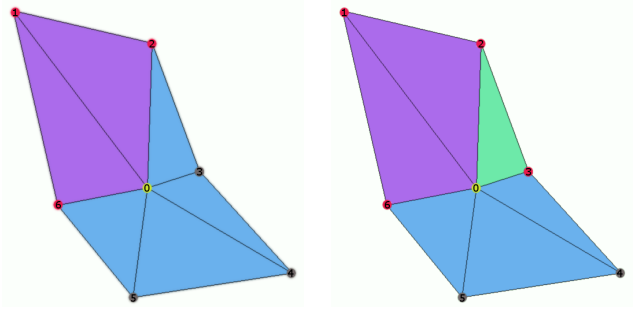
As an example for the above definition, let $\mathbb{NV}^{\text{ord}}(v_0, 8) = \{\hat{\mathbf{x}}_0, \hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_7\}$ be the set of 8 nearest neighbors of v_0 ordered with respect to their distance from v_0. Furthermore, let us assume that the vertices $\hat{\mathbf{x}}_3$ and $\hat{\mathbf{x}}_5$ are already visited. Then, $\text{dorder}^{(8)}(v_0, \hat{\mathbf{x}}_2) = 2$ and $\text{dorder}^{(8)}(v_0, \hat{\mathbf{x}}_6) = 4$. Ignoring the already visited vertices encountered in $\mathbb{NV}^{\text{ord}}(v_0, v_n)$ results an output sequence of lower entropy.

In practice, some fixed value is used for m, e.g. $\{8, 16, 32\}$ while on the implementation side, the search is sped up by using a KD-Tree [49]. In case $v_n \notin \mathbb{NV}^{\text{ord}}(\text{V}, m)$, we encode directly the index $z : v_n = \hat{\mathbf{v}}_t^z$ of the vertex v_n by outputting $-z-1$, so that a negative value indicates direct indexing. Equation (8) summarizes the above discussion and determines the output symbol of M-TFAN whenever a non-visited vertex is encountered while traversing the input mesh.

$$dorder(\hat{\mathbf{v}}, \hat{\mathbf{x}}) := \begin{cases} \text{dorder}^{(m)}(\hat{\mathbf{v}}, \hat{\mathbf{x}}), & \hat{\mathbf{x}} \in \mathbb{NV}^{\text{ord}}(\hat{\mathbf{v}}, m) \\ & \hat{\mathbf{x}} \notin \mathbb{NV}^{\text{ord}}(\hat{\mathbf{v}}, m), \\ -z-1, & \hat{\mathbf{x}} = \hat{\mathbf{v}}^z \in \hat{\mathbb{V}}_t^{\text{ord}} \end{cases} \tag{8}$$

We call $dorder(\hat{\mathbf{v}}, \hat{\mathbf{x}})$ the generalized distance order of vertex $\hat{\mathbf{x}}$ with respect to $\hat{\mathbf{v}}$ for some fixed m.

In Fig. 5, an example while traversing a TF centered at v_0 is illustrated. The already encoded triangles are in purple. The triangle to be encoded in each step is marked green. The yellow vertex denotes the center of the TF. The already visited vertices are in red while unvisited vertices are in gray. Before traversing the TF, the neighbor vertices of v_0 are ordered according to their euclidean distance from v_0. Thus, $\mathbb{NV}^{\text{ord}}(v_0, 6) = \{v_3, v_6, v_5, v_4, v_2, v_1\}$. In step #1, the vertex v_3 is visited. Since it is an unvisited vertex (previously gray), M-TFAN outputs its generalized distance order which is 0. Similarly, in step #2, v_4 is visited for the first time. M-TFAN



(a) M-TFAN Step #0: Before traversing the TF.

(b) M-TFAN Step #1: Visiting vertex v_3

(c) M-TFAN Step #2: Visiting vertex v_4

(d) M-TFAN Step #3: Visiting vertex v_5

(e) M-TFAN Step #4: Encoding the last triangle

Fig. 5: M-TFAN Example. The already encoded triangles are in purple. The triangle to be encoded in each step is marked green. The yellow vertex denotes the center of the TF. The already visited vertices are in red while unvisited vertices are in gray.

outputs its generalized distance order which is 1, since at that point the vertices v_3 and v_6 are already visited and the zero-based index of v_4 in $\mathbb{NV}^{\text{ord}}(v_0, 6)$, ignoring v_3 and v_6, is 1. Following step #3, the previously unvisited vertex v_5 is traversed. Similarly, its generalized distance order 0 is being output by M-TFAN. Finally, in step #4 no unvisited vertex is traversed, thus M-TFAN does not output any generalized distance order in this case. Concluding, the M-TFAN output for the generalized distance order in the example case is {0,1,0}.

V. Motion-Based Coding

Furthermore, for the needs of a TI system, where maximum compression ratio with minimum computational time is desired, a simple extension to the proposed method can be applied.

By using the skinning information of section IV-A, $\mathcal{M}_t$ is segmented into sub-meshes with each sub-mesh being the skin attached to a particular skeleton bone. When encoding EP-Frames, only the sub-meshes that belong to bones that are under significant "per-bone motion" are encoded using skeletal motion thresholding, as described in subsection III-B. For the rest of the mesh, only motion parameters (skeleton bone motion and ICP rotation/translation information) are transmitted. The decoder animates the corresponding sub-meshes of $\hat{\mathcal{M}}_I$ that lack correction residuals and zippers [50] the neighbouring sub-meshes. This simple concept enables to further reduce the execution time of the proposed algorithm, as well as to increase compression rates under a very reasonable distortion. A demonstrative experiment for the motion-based coding concept is presented in Section VI-B5.

VI. Experimental-Results

In this section we present experimental results of our work and compare it against the MPEG-4 TFAN [1] static mesh coder. The datasets used for evaluation are H-TVMs that are publicly available on `http://vcl.iti.gr/reconstruction/`and are considered part of the official MPEG-3DGC database [2]. When evaluating the proposed method, I-Frames were encoded using [1] and their bit-rate and distortion is counted for the overall performance of the proposed method. Moreover, during evaluation we do not use any P Frames at all. All the sequences are compressed using only I and EP-Frames.

A. Distortion Metrics

To evaluate the proposed method we use three different metrics each one serving a different purpose.

1) Point Cloud Metric [M1]: The first metric, presented hereby and denoted as [M1], is the proposed metric for point cloud comparison between the original and the compressed TVM sequence. For the single frame F_t we consider the original, uncompressed mesh $\mathcal{M}_t$ with N_t vertices and its corresponding compressed mesh $\hat{\mathcal{M}}_t$. For any vertex $\mathbf{v}_t^k \in \mathcal{M}_t$, $k \in \{0, 1, \ldots, N_t - 1\}$, let $\mathbf{H}_t(\mathbf{v}_t^k) := \hat{\mathbf{v}}_0 \in \hat{\mathcal{M}}_t$ denote its nearest vertex in $\hat{\mathcal{M}}_t$ i.e. $\text{dist}(\mathbf{v}_t^k, \hat{\mathbf{v}}_0) < \text{dist}(\mathbf{v}_t^k, \hat{\mathbf{v}}) \, \forall \, \hat{\mathbf{v}} \in \hat{\mathcal{M}}_t$ with $\hat{\mathbf{v}} \neq \hat{\mathbf{v}}_0$. Let $\mathbb{SM} := \{\mathcal{M}_t, \, t \in \{0, 1, \ldots, M-1\}\}$ and $\hat{\mathbb{SM}} := \{\hat{\mathcal{M}}_t, \, t \in \{0, 1, \ldots, M-1\}\}$, where M denotes the total number of frames in a sequence. The geometric root mean squared distance of $\mathbb{SM}$ to $\hat{\mathbb{SM}}$, is defined to be:

$$\text{gd}_{\text{rms}}(\mathbb{SM}, \hat{\mathbb{SM}}) := \sqrt{\frac{1}{\sum_{t=0}^{M-1} N_t} \sum_{t=0}^{M-1} \sum_{k=0}^{N_t-1} ||\mathbf{v}_t^k - \mathbf{H}_t(\mathbf{v}_t^k)||_2^2}. \tag{9}$$

Similarly, the geometric root mean squared distance of $\hat{\mathbb{SM}}$ to $\mathbb{SM}$, denoted as $\text{gd}_{\text{rms}}(\hat{\mathbb{SM}}, \mathbb{SM})$, is defined equivalently.

Given (9), the geometric root mean squared error for a sequence of M frames is defined to be:

$$\mathrm{RMSE_g} \coloneqq \max(\mathrm{gd_{rms}}(\mathbb{SM}, \hat{\mathbb{SM}}), \mathrm{gd_{rms}}(\hat{\mathbb{SM}}, \mathbb{SM})). \quad (10)$$

To compute the root mean squared error for color we use equations (9) and (10), but we plug in the color attributes of the vertices $\mathbf{v}_t^k$ and $\mathbf{H}(\mathbf{v}_t^k)$, instead of their geometric coordinates.

2) Surface Error Metric [M2]: To measure RMSE surface errors, the METRO [51] tool was employed. Following the evaluation metric used in [33] and the notation of Section VI-A1, we define the root mean squared surface distance between the sequences $\mathbb{SM}$ and $\hat{\mathbb{SM}}$ to be:

$$\mathrm{sd_{rms}}(\mathbb{SM}, \hat{\mathbb{SM}}) \coloneqq \frac{1}{M} \sum_{t=0}^{M-1} \mathrm{md_{rms}}(\mathcal{M}_t, \hat{\mathcal{M}}_t), \quad (11)$$

where $\mathrm{md_{rms}}(\mathcal{M}_t, \hat{\mathcal{M}}_t)$ denotes the root mean squared distance between $\mathcal{M}_t$ and $\hat{\mathcal{M}}_t$, reported by METRO. Since $\mathrm{md_{rms}}(\mathcal{M}_t, \hat{\mathcal{M}}_t) \neq \mathrm{md_{rms}}(\hat{\mathcal{M}}_t, \mathcal{M}_t)$, i.e the forward distance between two meshes is in general different than the backward distance, the root mean squared surface error for a sequence of M frames is defined to be:

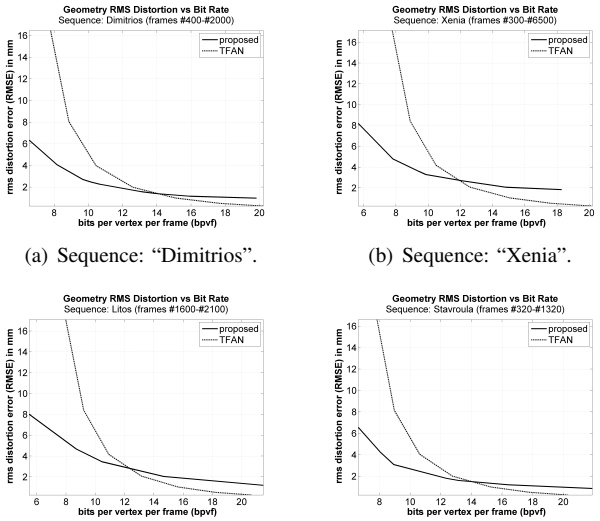
$$\mathrm{RMSE_s} \coloneqq \max(\mathrm{sd_{rms}}(\mathbb{SM}, \hat{\mathbb{SM}}), \mathrm{sd_{rms}}(\hat{\mathbb{SM}}, \mathbb{SM})). \quad (12)$$

The surface error metric is denoted as [M2].

3) PSNR Metric [M3]: According to [52], to measure image-quality error a PSNR metric in image plane, denoted as [M3], is used. The PSNR is calculated for the whole sequence by rendering each mesh of the sequence from 10 views, equally distributed around the mesh, at a resolution of 1280×1024 pixels.

B. Evaluation

1) Geometry and Connectivity Compression [M1]: In this subsection we evaluate against four TVM sequences, considering only geometry and connectivity of TVMs using the metric [M1]. The first sequence, namely "S04_5KW_DimitriosSession4-PoissonLow" ("Dimitrios" for short), consists of 1600 frames with average vertex count per frame 35100. The second one, namely "S17_5KW_Xenia-PoissonLow" ("Xenia" for short), consists of 6200 frames with 35200 vertices per frame on average. Thirdly, we use a subsequence of "S08_5KW_LitosSession4-PoissonLow" ("Litos" for short) consisting of 500 frames with 29667 number of vertices per frame. Finally, we use a subsequence of "S13_5KW_Stavroula-PoissonLow" ("Stavroula" for short) with 33792 number of vertices per frame in 1000 frames total. Geometry distortion versus bit-rate curves are depicted in Fig. 6. The diagrams of Fig. 6 contend that for "Xenia" and "Litos" sequences the proposed method significantly outperforms MPEG-4 TFAN [1] at low bit-rates (below 12 bits per vertex per frame (bpvf)), while for higher rates our method competes with the state-of-the-art static coder. For "Dimitrios" and "Stavroula" a similar fact holds for the margin of 14 bpvf. Thus, when high compression rates are desirable, the



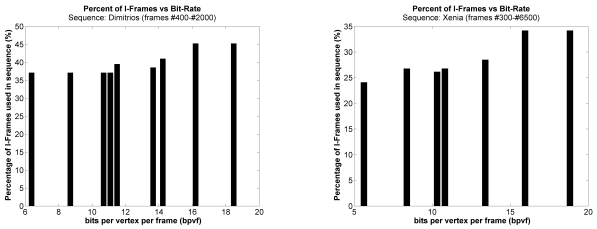
(a) Sequence: "Dimitrios". (b) Sequence: "Xenia".

(c) Sequence: "Litos". (d) Sequence: "Stavroula".

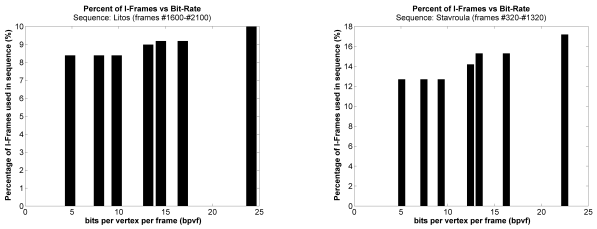
Fig. 6: Geometry Distortion vs Bit-Rate [M1].

proposed method shows an advantage. The bit-rate interval where the proposed method is advantageous when compared to [1] is wider in "Dimitrios" and "Stavroula" sequences than in "Xenia" and "Litos". This difference in the performance of the proposed method across the four sequences might be explained when considering the skeleton quality in each sequence. The skeleton quality, when judged visually, in "Dimitrios" and "Stavroula" is better than the skeleton quality in "Xenia" and "Litos". Moreover, it is worth noting that the convergence rate of the rate-distortion curves depicted in Fig. 6 for the proposed method is lower than the one of [1]. As the experiments were conducted with users standing in arbitrary poses, it was observed that skinning errors may occur (e.g. vertices actually belonging to torso are associated with a hand or vice versa). This is due to the fact that the skeleton is not always accurately aligned to the reconstructed mesh due to imperfect tracking. Consequently, when per-bone prediction is taking place, the mis-associated vertices ofter have large prediction errors which cannot be effectively represented by a vector in the Vector Quantization codebook, resulting lower convergence towards zero of the rate-distortion curves. In Fig. 8 and Fig. 9 two frames are depicted at various bit rates for both methods. Fig. 7 depicts the percentage of I-Frames used versus the bit-rate for all the evaluation sequences. These diagrams where obtained by varying the "skeletal motion" threshold in the range 400-500mm.

2) Geometry and Connectivity Compression [M2]: In addition to the previous evaluation, we further evaluate the proposed method in the sequences "Dimitrios" and "Stavroula" under the metric [M2]. The evaluation results are depicted in Fig. 10. As can be concluded from the diagrams, in terms of surface error, the proposed method shows a significant advantage in the low bit-rates up to 11 bpvf. When considering surface distortion, with respect to point cloud error, the advantageous margin of the proposed method is narrower. This is expected, since the EC-VQ method used for



(a) Sequence: "Dimitrios". (b) Sequence: "Xenia".



(c) Sequence: "Litos". (d) Sequence: "Stavroula".

Fig. 7: Percentage of I-Frames for different bit-rates.

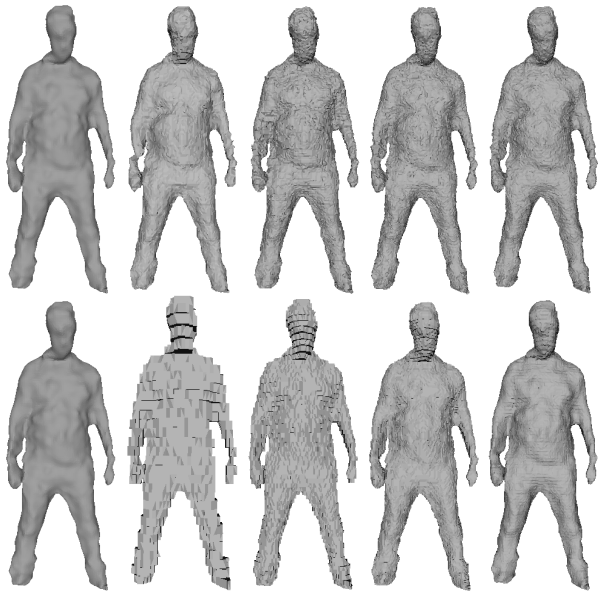


Fig. 8: Frame #500 from "Dimitrios" sequence at various bit rates rendered with flat shading. The first picture of every row is the original uncompressed mesh. Top row from left to right: the proposed method at 6.3 | 8.7 | 10.55 | 11.5 bpvf. Bottom row from left to right: MPEG TFAN [1] at 7.7 | 8.8 | 10.45 | 12.6 bpvf, respectively.

vertex quantization is an optimization technique that aims for minimizing the overall mean squared error in terms of point cloud, not taking into account any surface properties. Thus, the high surface errors is an undesirable side effect stemming from the nature of the optimization technique.

3) Geometry, Connectivity and per Vertex Color Compression [M1]: At this point we consider additionally per vertex color attribute encoding. We evaluate against a subsequence of "Xenia" and a subsequence of "Stavroula" that both consist of 400 frames using the metric [M1]. In Fig. 11, the performance

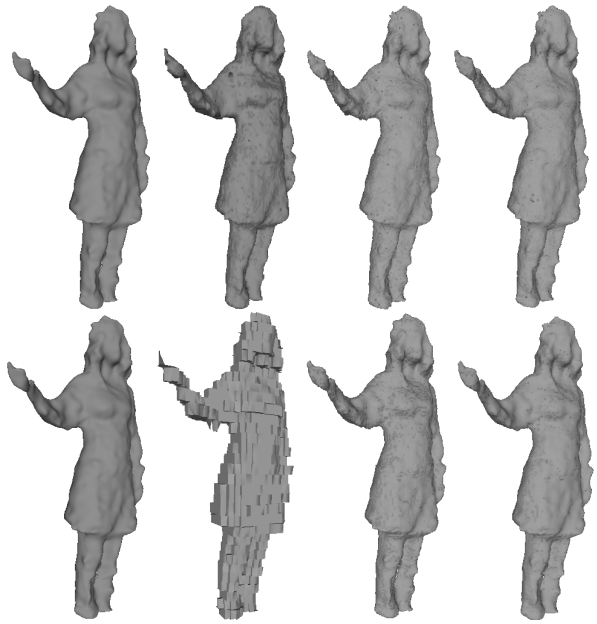
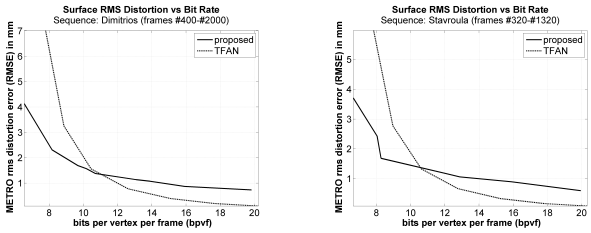


Fig. 9: Frame #336 from "Xenia" sequence at various bit rates rendered with smooth shading. The first picture of every row is the original uncompressed mesh. Top row from left to right: the proposed method at 5.7 | 10.81 | 13.4 bpvf . Bottom row from left to right: MPEG TFAN [1] at 7.5 | 12.58 | 14.75 bpvf respectively.

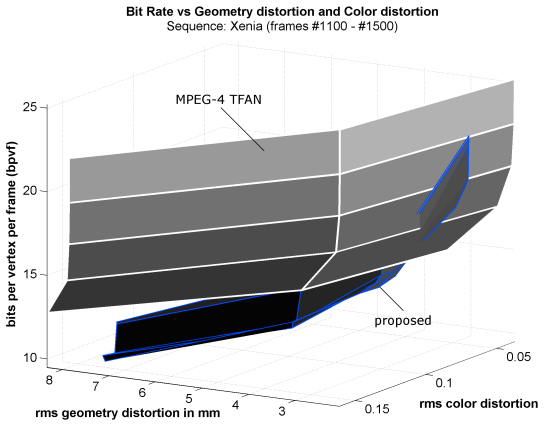


(a) Sequence: "Dimitrios". (b) Sequence: "Stavroula".

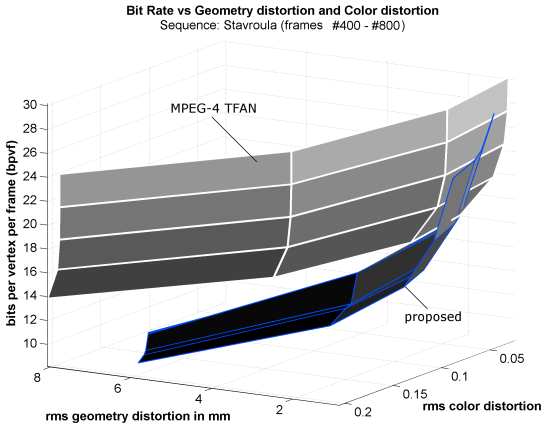
Fig. 10: Surface Distortion vs Bit-Rate [M2].

of the two methods in terms of bit-rate and distortion for both geometry and color is shown. In the x-y plane of this diagram geometry and color distortions are given respectively, while the z-axis depicts the total number of bits per vertex per frame (bpvf) needed to encode all needed information i.e. geometry, color and connectivity. Once again, the proposed method performs better than the static coder in the lower bit-rates and looses ground in the higher ones. However, when dealing with the combined problem of geometry, connectivity and color compression of TVMs, the area where the proposed method is advantageous is wider than when considering geometry and connectivity alone. Figures 12 and 13 further support the last argument by depicting plane intersections of the 3D chart of Fig. 11. In Fig. 14 and Fig. 15 two frames of the two different sequences are shown for visual subjective comparison.

4) Geometry, Connectivity and per Vertex Color Compression [M3]: In this section we evaluate the proposed method in terms of PSNR in the sequences "Dimitrios" and "Stavroula",

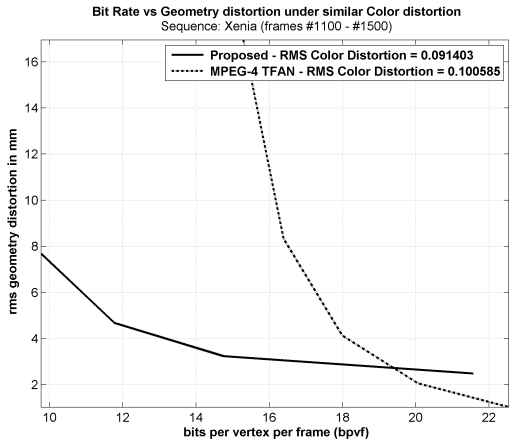


(a) Sequence: "Xenia".

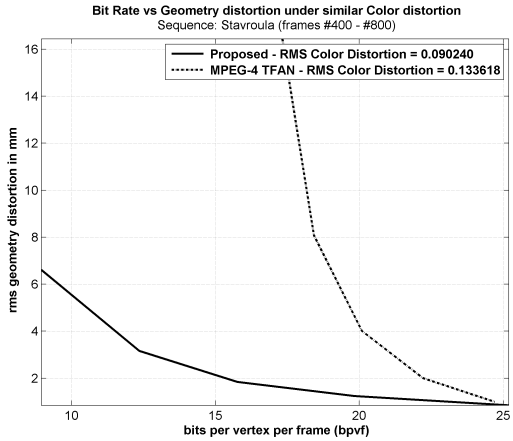


(b) Sequence: "Stavroula".

Fig. 11: Geometry and Color Distortion vs Bit-Rate [M1].



(a) Sequence: "Xenia".



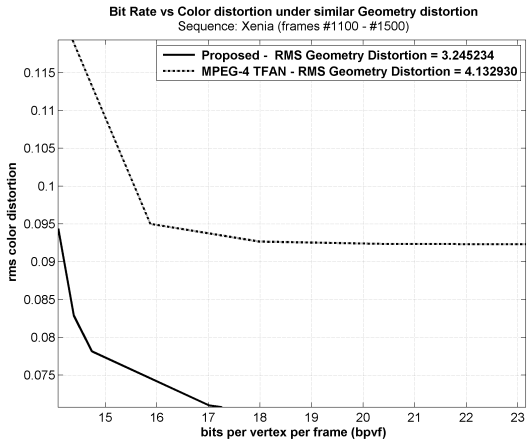
(b) Sequence: "Stavroula".

Fig. 12: Geometry Distortion under similar color distortion vs Bit-Rate [M1].

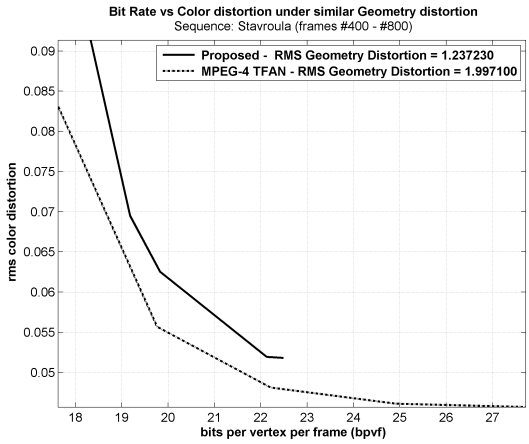
using the image quality-related metric [M3]. The evaluation results are depicted in Fig. 16. The evaluation curves were produced by using a fixed configuration for color distortion and varying the geometry distortion for both [1] and the proposed method. The curves contend that when considering the visual output of both compressors, the proposed method achieves better performance for bit-rates up to 20 bpvf for the "Stavroula" sequence. For "Dimitrios", the proposed method remains better than its competitor for up to 16 bpvf. Then, for up to 19 bpvf the performance of the two methods becomes comparable, while above 19 bpvf the state-of-the-art static coder outperforms the proposed one. Thus, under an image quality-related metric other than [M2], the proposed method shows a significant range of advantageous performance.

5) Motion-Based Coding: When considering the evaluation of Motion-Based Coding (Section V), we choose some fixed quantization parameters and instead of encoding full-body mesh in EP-Frames, we encode only sub-meshes attached to the bones of the skeleton that are under significant "per-bone motion". By varying the used motion thresholds, we can achieve different bit-rates that are not due to different quantization parameters, but due to actually encoding less vertex and color residuals than normal. We evaluate against 480 frames of "Stavroula" sequence. Fig. 17 depicts encoding and decoding times vs bit-rate and Fig. 18 depicts PSNR vs bit-rate. The computational time is measured in the same workstation as in Section VI-B6. In Fig. 19 a frame from "Stavroula" sequence is depicted for subjective evaluation. When the mesh's skeleton quality is good enough, differences between the low and high bit-rates are hard to be visually perceived.

6) Internal Evaluation: In this section we evaluate the internals of the proposed method. The method is evaluated against "Xenia" subsequence, frames #1100 - #1500, at a mean rate of 13.2 bpvf. The internal evaluation involves full-body (i.e. not motion-based) coding of geometry, connectivity and per vertex colors. The output is split into four layers. The geometry layer contains the geometry prediction residuals, while the color layer contains the per vertex color residuals. The mapping layer that maps geometry and color residuals to their reference vertices/colors consists of the encoding of the graph presented in section IV-C. Lastly, the connectivity layer holds the output of M-TFAN. The distribution of the bit-rate for the



(a) Sequence: "Xenia".



(b) Sequence: "Stavroula".

Fig. 13: Color Distortion under similar geometry distortion vs Bit-Rate [M1].

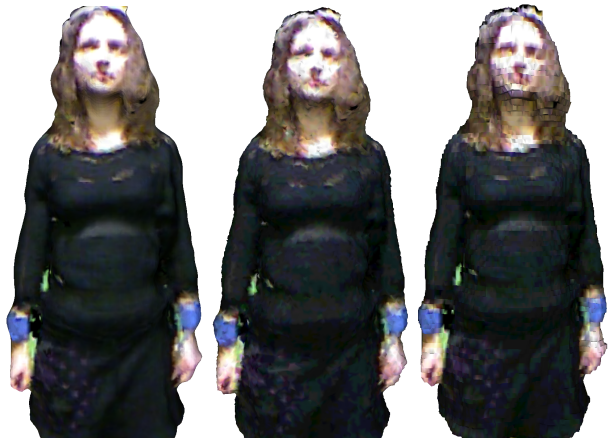


Fig. 14: Frame #1391 from "Xenia" sequence rendered with smooth shading. From left to right: uncompressed mesh | proposed method at 13.2 bpvf | MPEG TFAN [1] at 15.8 bpvf.

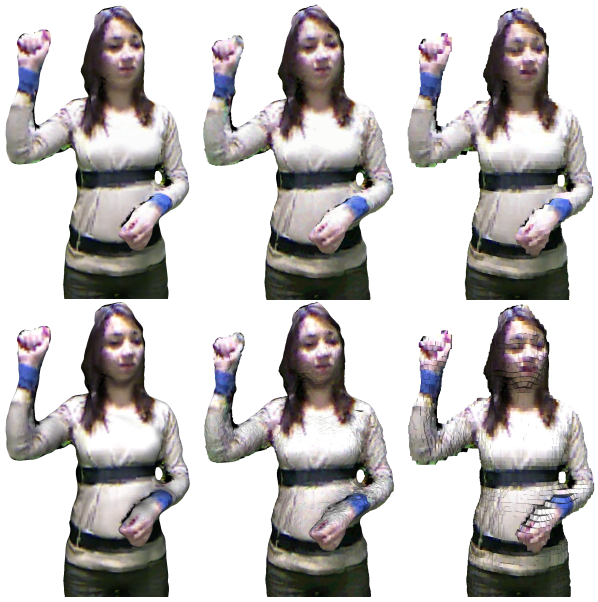
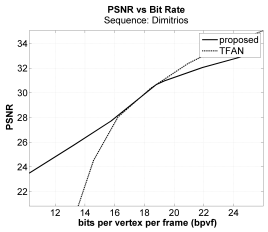
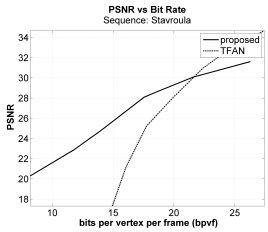


Fig. 15: Frame #615 from "Stavroula" sequence. Top row rendered without lighting and bottom row rendered with flat shading. From left to right: uncompressed mesh | proposed method at 19.8 bpvf | MPEG-4 TFAN [1] at 20.09 bpvf.



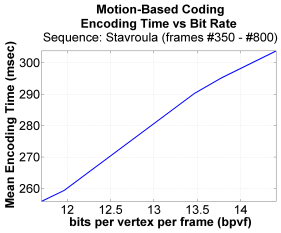
(a) Sequence: "Dimitrios".



(b) Sequence: "Stavroula".

Fig. 16: PSNR vs Bit-Rate [M3].

various layers is depicted in Fig. 21 and Fig. 22. Additionally, in the left part of Fig. 23, the distribution of distance-order values inside a single frame of the TVM sequence is given. The experimental results align with our intuition for designing M-TFAN. The extra cost for connectivity coding that M-TFAN introduces, (compared to the original TFAN [1] is about 1.3 bpvf, as shown in the right part of Fig. 23. Fig. 24 gives the distribution of the total bit-rate among the EP-Frames of the subsequence.



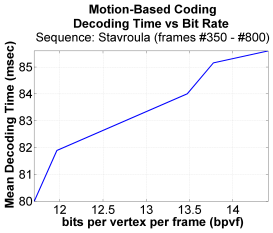


Fig. 17: Average encoding and decoding times for various bit-rates in Motion-Based Coding.

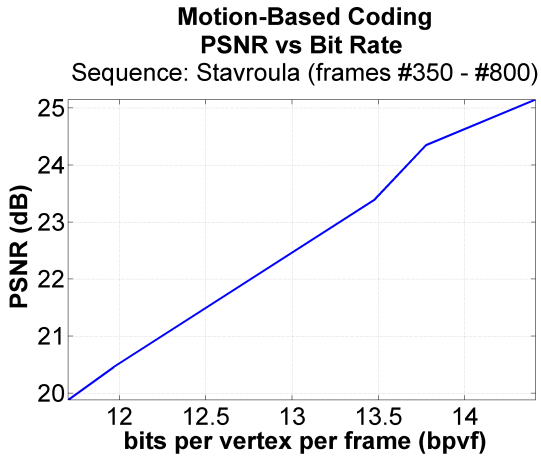


Fig. 18: PSNR vs bit-rate in Motion-Based Coding.

Fig. 19: Frame #400 of "Stavroula" sequence encoded with Motion-Based Coding. From left to right: uncompressed mesh | 11.5 bpvf | 13.2 bpvf | 14.5 bpvf.

Finally, we evaluate the algorithm in terms of execution time over the frames #1500 - #1920 of the same sequence for 13.2 bpvf taking into account geometry, connectivity and per vertex color compression/decompression. Fig. 25 presents the computational time distribution for both encoding and decoding for the mentioned subsequence. Despite the fact that the computational complexity of the proposed method is not cheap, a significant cut-down of the execution time is made possible by utilizing GPGPU computing and multi-threaded programming under a multi-core CPU. The execution times where measured in a workstation equipped with Intel Core i7 2700K 3.5GHz CPU, 8GB RAM and NVIDIA GTX560 GPU. The average frame rate for the encoder is near 5 frames per second (fps), while the decoder is able to run at approximately 12 fps. These frame rates where measured when employing full body EP-Frame coding and not the motion-based coding concept presented in section V. In case motion-based coding is employed, further drop in execution times can be achieved with an obvious distortion penalty, as shown in section VI-B5.

7) Robustness to Skeleton Quality: The method is evaluated for its sensitivity to skeleton quality against the sequence "Alex". This is a single view captured sequence (i.e not full 3D-body), reconstructed using a simple terrain (step-

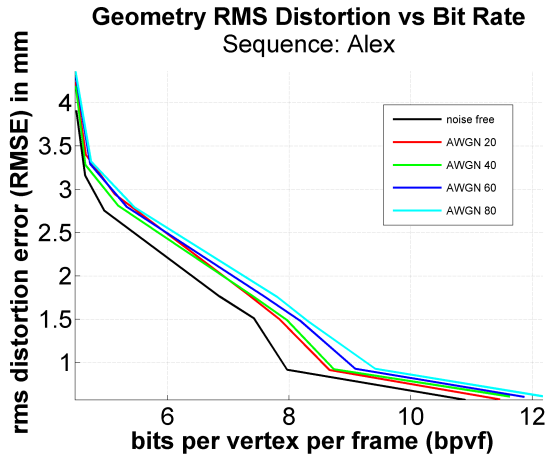
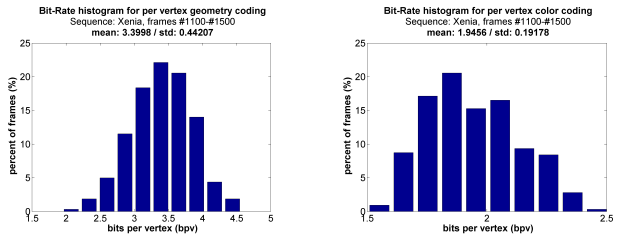


Fig. 20: Bit-Rate vs Geometry Distortion [M1] under different Skeleton Qualities



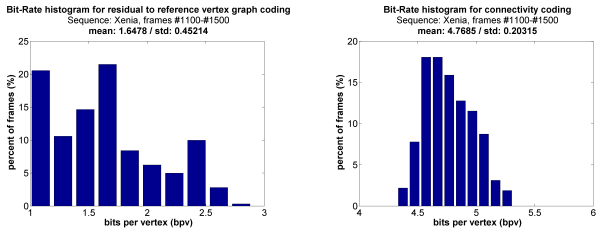
(a) Geometry rate. Mean: 3.3 bpv. (b) Color rate. Mean: 1.9 bpv.

Fig. 21: Histograms of geometry rate and color rate

discontinuity constrained) triangulation [45] and consists of 100 frames. The specific sequence was chosen for the sensitivity to skeleton quality evaluation due to its near ground-truth skeleton quality (as opposed to the rest of the sequences). White gaussian noise of equal powers was added per joint coordinate. Four different experiments were conducted, each one having a different additive skeletal noise power (measured in mm^2). The "ground truth" skeleton was corrupted using per joint coordinate noise powers of 20, 40, 60, 80. The curves of Fig. 20 were obtained by using the metric [M1] for the skeletons of different quality. The outcome of the experiment is reasonable since the more the skeleton corruption, the worse the proposed method performs in terms of compression rate. Despite the compression inefficiency, the method is robust to skeleton errors and does not compromise visual quality. Instead, this comes with a penalty in the compression rate. This can be concluded from the diagram since all the sampling points that correspond to the same distortion parameters, lie almost in the same ordinate.

VII. Conclusion and Future Work

In this paper a novel skeleton based approach for H-TVM compression was presented. Apart from efficiently encoding geometry and per vertex color by exploiting the temporal geometric redundancy, the proposed method is also shown to efficiently compress mesh connectivity. The paper introduces a modification to [1], named Modified-TFAN, that enables to



(a) Residual graph mapping rate. (b) Connectivity rate. Mean: 4.7 bpv. Mean: 1.6 bpv.

Fig. 22: Histograms of residual graph mapping rate and connectivity rate.

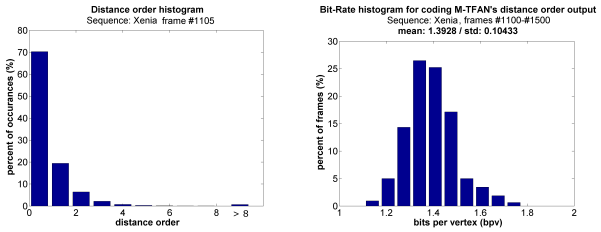


Fig. 23: Histograms of distance order values and M-TFAN's distance order rate.

encode connectivity at a small overhead when compared with the original static coder [1]. Evaluation was carried out over TVMs of MPEG-3DGC database [2]. The proposed method was evaluated against the state-of-the-art static mesh coder TFAN [1] that is currently a part of the MPEG-4 standard. The presented algorithm showed significant lower distortions at low bit-rates than its competitor, giving a practical proof of concept that efficient inter-frame redundancy exploitation in TVMs is possible and establishing ground for further improvements. Despite being complex, the execution time of the algorithm on a high-end PC is comparatively low when utilizing GPGPU computing and multi-core CPU multi-threaded programming, enabling for real-time transmission of H-TVMs. In addition, a novel approach to TVM compression using Motion-Based

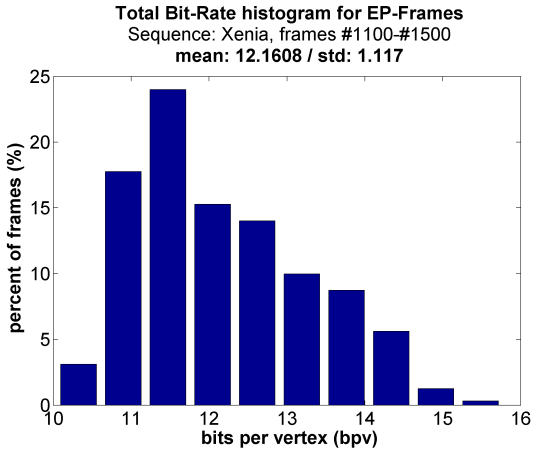


Fig. 24: Histogram of total rate for EP-Frames in a subsequence of "Xenia".

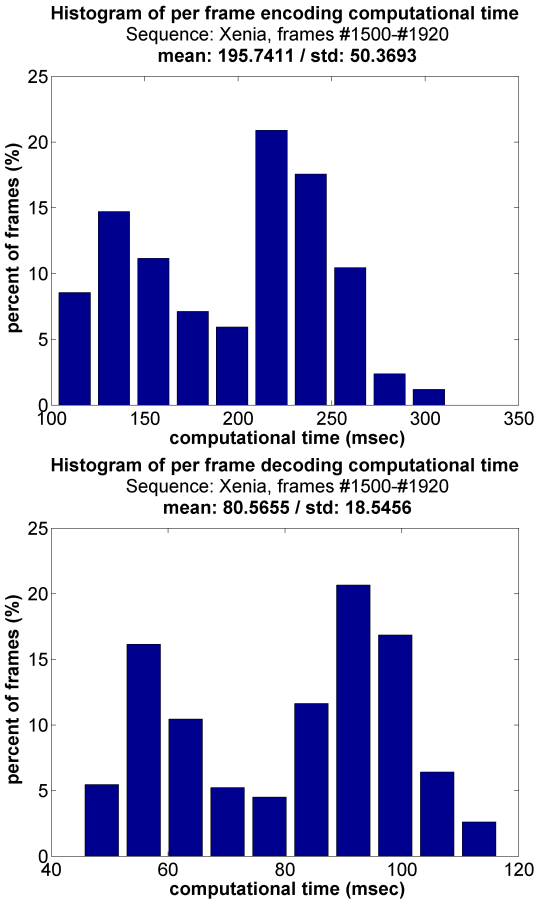


Fig. 25: Histogram of encoding and decoding computational time in a subsequence of "Xenia".

coding was presented. In the future, attempting to further reduce the size of the side information would probably allow for efficient compression at higher bit-rates as well. Moreover, there is no doubt that the proposed method will vastly benefit from future research in the area of skeleton tracking. Advances in skeleton tracking technology (e.g. Kinect 2) will provide an even more accurate and robust prediction scheme, potentially improving the rate-distortion performance of the proposed method, as well as the performance of motion-based coding. Finally, as a future work direction, we plan to study how the framework could be modified in order to be used for the compression of non-human TVMs, by replacing the skeleton-based predictor with a more generic one.

Acknowledgment

This work was supported by the EU funded project 3DLIVE, GA 318483. http://3dliveproject.eu/.

References

[1] K. Mamou, T. B. Zaharia, and F. J. Prêteux, "TFAN: A low complexity 3D mesh compression algorithm," *Journal of Visualization and Computer Animation*, vol. 20, no. 2-3, pp. 343–354, 2009.

[2] [Online]. Available: https://www.gti.ssr.upm.es/~mpeg/3dgc/3Dmodels/

[3] 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, 2007.

[4] S. Würmlin, E. Lamboray, and M. H. Gross, "3D video fragments: dynamic point samples for real-time free-viewpoint video," *Computers & Graphics*, vol. 28, no. 1, pp. 3–14, 2004.

[5] J.-M. Hasenfratz, M. Lapierre, and F. Sillion, "A real-time system for full body interaction with virtual worlds," in *Proceedings of the Tenth Eurographics conference on Virtual Environments*, ser. EGVE'04. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2004, pp. 147–156.

[6] A. Smolic, "3D video and free viewpoint video-from capture to display," *Pattern Recognition*, vol. 44, no. 9, pp. 1958–1968, Sep. 2011.

[7] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 1297–1304.

[8] M. Deering, "Geometry compression," in *SIGGRAPH*, New York, NY, USA, 1995, pp. 13–20.

[9] G. Taubin and J. Rossignac, "Geometric compression through topological surgery," *ACM TRANSACTIONS ON GRAPHICS*, vol. 17, pp. 84–115, 1998.

[10] C. Touma and C. Gotsman, "Triangle mesh compression," in *Graphics Interface*, Vancouver, BC, Canada, 1998, pp. 26–34.

[11] P. Alliez and M. Desbrun, "Valence-driven connectivity encoding for 3D meshes," *Computer Graphics Forum*, vol. 20, no. 3, pp. 480–489, 2001.

[12] J. Rossignac, "Edgebreaker: Connectivity compression for triangle meshes," *IEEE Trans. Vis. Comput. Graph.*, vol. 5, no. 1, pp. 47–61, 1999.

[13] J. Rossignac and A. Szymczak, "Wrap&zip decompression of the connectivity of triangle meshes compressed with edgebreaker," *Journal of Computational Geometry, Theory and Applications*, vol. 14, pp. 119–135, 1999.

[14] D. King and J. Rossignac, "Guaranteed 3.67v bit encoding of planar triangle graphs," in *Canadian Conference on Computational Geometry*, Vancouver, British Columbia, Canada, 1999.

[15] V. Coors and J. Rossignac, "Delphi: geometry-based connectivity prediction in triangle mesh compression," *The Visual Computer*, vol. 20, no. 8-9, pp. 507–520, 2004.

[16] E.-S. Lee and H.-S. Ko, "Vertex data compression for triangular meshes," in *Proceedings of the 8th Pacific Conference on Computer Graphics and Applications*, ser. PG '00. Washington, DC, USA: IEEE Computer Society, 2000, pp. 225–234.

[17] Z. Karni and C. Gotsman, "Spectral compression of mesh geometry," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '00. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 279–286.

[18] S.-C. Cheng, C.-T. Kuo, and D.-C. Wu, "A novel 3d mesh compression using mesh segmentation with multiple principal plane analysis," *Pattern Recognition*, vol. 43, no. 1, pp. 267–279, 2010.

[19] H. Hoppe, "Progressive meshes," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '96. New York, NY, USA: ACM, 1996, pp. 99–108.

[20] H. Hoppe, "Efficient implementation of progressive meshes," *Computers & Graphics*, vol. 22, no. 1, pp. 27 – 36, 1998.

[21] G. Taubin, A. Guéziec, W. Horn, and F. Lazarus, "Progressive forest split compression," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '98. New York, NY, USA: ACM, 1998, pp. 123–132.

[22] R. Pajarola and J. Rossignac, "Compressed progressive meshes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, no. 1, pp. 79–93, Jan. 2000.

[23] P. Alliez and M. Desbrun, "Progressive compression for lossless transmission of triangle meshes," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '01. New York, NY, USA: ACM, 2001, pp. 195–202.

[24] S. Valette and R. Prost, "Wavelet-based progressive compression scheme for triangle meshes: Wavemesh," *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, no. 2, pp. 123–129, Mar. 2004.

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

[26] S. Valette, R. Chaine, and R. Prost, "Progressive lossless mesh compression via incremental parametric refinement," *Computer Graphics Forum*, vol. 28, no. 5, pp. 1301–1310, 2009.

[27] J. Peng and C.-C. J. Kuo, "Geometry-guided progressive lossless 3d mesh coding with octree (ot) decomposition," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 609–616, Jul. 2005.

[28] L. Ibarria and J. Rossignac, "Dynapack: Space-time compression of the 3d animations of triangle meshes with fixed connectivity," in *ACM Symp. Computer Animation*, Aire-la-Ville, Switzerland, Switzerland, 2003, pp. 126–135.

[29] Z. Karni and C. Gotsman, "Compression of soft-body animation sequences," *Computers & Graphics*, vol. 28, no. 1, pp. 25–34, 2004.

[30] L. Váša and V. Skala, "CODDYAC: Connectivity driven dynamic mesh compression," in *3DTV-CON 2007*. Piscataway: IEEE, 2007, pp. 1–4.

[31] L. Váša and V. Skala, "COBRA: Compression of the basis for pca represented animations," *Computer Graphics Forum*, vol. 28, no. 6, pp. 1529–1540, 2009.

[32] F. Payan and M. Antonini, "Wavelet-based compression of 3d mesh sequences," in *In Proceedings of IEEE ACIDCA-ICMI2005, Tozeur*, 2005.

[33] K. Mamou, T. Zaharia, and F. Prêteux, "A skinning approach for dynamic 3d mesh compression: Research articles," *Comput. Animat. Virtual Worlds*, vol. 17, no. 3-4, pp. 337–346, Jul. 2006.

[34] K. Mamou, T. B. Zaharia, and F. J. Prêteux, "FAMC: The MPEG-4 standard for animated mesh compression," in *ICIP*, San Diego, California, USA, 2008, pp. 2676–2679.

[35] X. Gu, S. Gortler, and H. Hoppe, "Geometry images," *ACM Trans. Graphics (SIGGRAPH)*, vol. 21, no. 3, 2002.

[36] H. Briceno, P. Sander, L. McMillan, S. Gortler, and H. Hoppe., "Geometry videos: A new representation for 3D animations," in *Symposium on Computer Animation*, 2003, pp. 136–146.

[37] D. Quynh, Y. He, X. Chen, J. Xia, Q. Sun, and S. Hoi, "Modeling 3D articulated motions with conformal geometry videos (CGVs)," in *Proceedings of the 19th ACM International Conference on Multimedia*. NY, USA: ACM, 2011, pp. 383–392.

[38] S. Gupta, K. Sengupta, and A. A. Kassim, "Compression of dynamic 3D geometry data using iterative closest point algorithm," *Computer Vision and Image Understanding*, vol. 87, no. 13, pp. 116 – 130, 2002.

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

[40] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, 1992.

[41] S.-R. Han, T. Yamasaki, and K. Aizawa, "Geometry compression for time-varying meshes using coarse and fine levels of quantization and run-length encoding," in *ICIP*, San Diego, California, USA, 2008, pp. 1045–1048.

[42] T. Yamasaki and K. Aizawa, "Patch-based compression for time-varying meshes," in *ICIP*, Hong Kong, China, 2010, pp. 3433–3436.

[43] J.-M. Lien, G. Kurillo, and R. Bajcsy, "Multi-camera tele-immersion system with real-time model driven data compression: A new model-based compression method for massive dynamic point data," *Vis. Comput.*, vol. 26, no. 1, pp. 3–15, Nov. 2009.

[44] D. S. Alexiadis, D. Zarpalas, and P. Daras, "Real-time, realistic full-body 3D reconstruction and texture mapping from multiple kinects," in *11th IEEE IVMSP Workshop: 3D Image/Video Technologies and Applications, Yonsei University, Seoul, Korea*, 2013.

[45] D. S. 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, no. 2, pp. 339–358, 2013.

[46] T. Maeda, T. Yamasaki, and K. Aizawa, "Model-based analysis and synthesis of time-varying mesh," in *Proceedings of the 5th international conference on Articulated Motion and Deformable Objects*, ser. AMDO '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 112–121.

[47] E. S. Jang, S. Lee, B. Koo, D. Kim, and K. Son, "Fast 3d mesh compression using shared vertex analysis," *ETRI*, vol. 32, no. 1, pp. 163–165, Feb. 2010.

[48] P. Chou, T. Lookabaugh, and R. Gray, "Entropy-constrained vector quantization," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 1, pp. 31–42, Jan. 1989.

[49] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.

[50] G. Turk and M. Levoy, "Zippered polygon meshes from range images," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '94. New York, NY, USA: ACM, 1994, pp. 311–318.

[51] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: Measuring error on simplified surfaces," Paris, France, Tech. Rep., 1996.

[52] M. Corsini, M. C. Larabi, G. Lavoué, O. Petřík, L. Váša, and K. Wang, "Perceptual metrics for static and dynamic triangle meshes," *Computer Graphics Forum*, vol. 32, no. 1, pp. 101–125, 2013.