The Visual Computer (2021) 37:973–991
https://doi.org/10.1007/s00371-020-01847-z

ORIGINAL ARTICLE

Compression of geometry videos by 3D-SPECK wavelet coder

Canan Gulbak Bahce[1] · **Ulug Bayazit**[1]

Published online: 11 May 2020

Abstract
A geometry video is formed as a sequence of geometry images where each frame is a remeshed form of a frame of an animated mesh sequence. For efficiently coding geometry videos by exploiting temporal as well spatial correlation at multiple scales, this paper proposes the 3D-SPECK algorithm which has been successfully applied to the coding of volumetric medical image data and hyperspectral image data in the past. The paper also puts forward several postprocessing operations on the reconstructed surfaces that compensate for the visual artifacts appearing in the form of undulations due to the loss of high-frequency wavelet coefficients, cracks near geometry image boundaries due to vertex coordinate quantization errors and serrations due to regular or quad splitting triangulation of local regions of large anisotropic geometric stretch. Experimental results on several animated mesh sequences demonstrate the superiority of the subjective and objective coding performances of the newly proposed approach to those of the commonly recognized animated mesh sequence coding approaches at low and medium coding rates.

Keywords Geometry image · Geometry video · Compression · Wavelet transform · Postprocessing

1 Introduction

Animated 3D mesh sequences have been widely used for synthetically representing the surface structures of moving 3D objects in movies and video games. In such applications, presentation of high detailed animated mesh sequences in high spatiotemporal resolution to the viewer typically requires very large data rates and file sizes for their transmission and storage, respectively. Under such data loads, in order to conserve storage space, or transmission time over bandwidth limited networks, compression techniques have been investigated for the past 20 years.

Resampling of the geometry on a regular structure has been considered since [36]. When properly performed, this approach benefits 3D mesh compression since regular connectivity can be inferred at the decoder and does not have to be stored or transmitted. Geometry images [13] have later been proposed as a remeshing technique that facilitates the existent 2D image processing and coding tools to be applied to the processing and coding of the geometry and related attributes of the 3D surface. The entire surface geometry is mapped to a geometry image by iterations of mesh cutting and parametrization steps followed by a sampling of the final parameter domain on a 2D regular rectangular grid. The steps are performed to minimize the geometric stretch from the parameter domain to the 3D physical space. The approach of geometry images has been extended to geometry videos [5] by jointly cutting and parametrizing multiple consecutive frames of an animated mesh sequence to yield a single set of cut paths and a single parametrization common to all these frames. Geometry video frames can be efficiently read into the GPU as textures and can be processed by commonly known video processing or coding tools.

Although commonly accepted video compression standards can be applied to code geometry video data [20,26,56], such approaches cannot fully exploit the statistical dependencies in the geometry video since the main coding tools in the existent video compression standards are very much optimized for natural image sequences and not for any form of synthetic video data. It is for this reason that specialized coding tools adapted to image sequences of screen content data

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s00371-020-01847-z) contains supplementary material, which is available to authorized users.

✉ Canan Gulbak Bahce
gulbakcanan@itu.edu.tr

Ulug Bayazit
ulugbayazit@itu.edu.tr

[1] Department of Computer Engineering, Istanbul Technical University, Istanbul, Turkey

[57] have been developed in the recent video coding standards. With this understanding, we have pursued an efficient solution for coding geometry video data.

This paper proposes the efficient 3D wavelet transform domain compression method called 3D-SPECK for coding geometry video data. 3D-SPECK has been employed in the past for coding hyperspectral and biomedical image sequences. It achieves its efficiency by exploiting spatial and temporal correlation at multiple scales. For low and medium coding rates, the distortion–rate performance of the proposed system surpasses the performances of the recognized animated mesh compression methods found in the literature and the video coding standards. The current paper also demonstrates the utility of several postprocessing operations applied to the decoded geometry video for mitigating the severity of the visual artifacts arising from the culling of high-frequency wavelet coefficients and the anisotropy of the geometric stretch associated with the geometry image remeshing.

The organization of this paper is as follows. Next section discusses the relevant work in the literature on animated mesh compression and geometry video coding. Section 3 elaborates on the components of the proposed geometry video coding system. After brief explanations of the processes for obtaining geometry images and geometry videos in Sect. 3.1, and the 3D Discrete Wavelet Transform in Sect. 3.2, Sect. 3.3 provides the details of the 3D-SPECK wavelet coding method. Specifically, Sect. 3.3.1 describes the type S sets of DWT coefficients and their hierarchical partitioning based on their significance with respect to exponentially decreasing thresholds and Sect. 3.3.2 presents the algorithmic flow. Sections 3.3.3 and 3.3.4 cover the processing order of type S sets, and the entropy coding of their significance decisions and the refinement decisions of significant coefficients, respectively. Section 4 describes in detail three postprocessing operations, namely the stitching of the reconstructed mesh boundaries (Sect. 4.1), adaptive smoothing of the high-frequency coding artifacts (Sect. 4.2), and adaptive stretch compensating triangulation of the reconstructed vertices (Sect. 4.3). Section 5 presents experimental results that compare the performance of the proposed compression system against the performances of several state-of-the-art compression methods. Concluding remarks and directions for extensions of the current work are presented in the last section.

2 Related work

A few algorithms rely on prediction to achieve the compression of dynamic mesh sequences. The vertex traversal-based compression algorithm of [23] exploits the local inter-frame and intra-frame dependencies among vertex locations by using two different extrapolating space-time predictors. Linear and nonlinear dependencies between layers and frames are exploited by layer-wise operating spatiotemporal predictors in [46]. Recently, a patch-based nonlinear prediction method [19] for motion estimation has been proposed for animated mesh sequence compression where the precision for quantizing the prediction error of each patch is based on a logarithmic model of its complexity.

Among the PCA-based methods, [1] applies PCA to key frames of an animation sequence and [24] extends this idea by linear predictive coding of significant PCA vectors in order to exploit temporal dependencies. A more advanced scheme by [44] applies multiple PCA's to the clusters of coherent rigid body parts with similar motion paths to perform the data compression. The recent approach of [52] applies the geometric Laplacian of an average (decoded) frame computed in edge space to the vectors describing the vertex trajectories in the reduced PCA trajectory space to get the delta trajectories. The delta trajectories are optionally predictively coded by applying local rigid transformations on the differential coordinates which are obtained by employing the geometric Laplacian of the average mesh for prediction.

For scalable, multiresolution compression, wavelet transform is preferred to PCA. The work of [17] uses a multiresolution wavelet representation in the spatial domain followed by progressive encoding of the wavelet detail information in the I and P frames. The subbands of temporal wavelet transform of frames of an animated mesh sequence are bit allocated by rate distortion optimization in [34]. A more elaborate scheme based on temporal wavelet transform is presented in [4], where the temporal wavelet transform is applied to the geometry compensated frames followed by R-D optimized rate allocation. The geometry compensation is facilitated by a different affine transform applied to each cluster of vertices obtained by motion-based clustering.

As with [44] and [4], [10] and [33] make use of segmentation. In [10], a base mesh is segmented into rigidly transforming parts to approximate the sequence of frames within a user specified distortion which is mitigated by spatiotemporal smoothing. In the motion compensated predictive coder of [33], spatial coherence of motion vectors is exploited by R-D optimization over all subdivisions of the motion vectors into cells on an octree structure and the prediction mode (one of direct coding, trilinear interpolation, mean replacement) for each cell.

Skinning-based compression [28,29] takes advantage of the rigid motion of the vertices to cluster the mesh into segments and model each segment's motion with a 3D affine transformation. A high compression ratio is achieved by combining this technique with the context-based adaptive binary arithmetic coding [30]; this method has been adopted in the MPEG-4 standard [37].

In geometry images [7,13,38], the geometry of an irregular surface mesh is parameterized on a rectangular 2D domain,

and the domain is sampled on a regular grid to form a geometry image. The parametrization in [13] aims to minimize the geometric-stretch measure of [42]. Briceno et al. [5] extend the geometry image method of [13] to animated mesh sequences by converting each frame of the animated mesh sequence to a frame of the geometry video (GV) by using the geometry image method. The compression scheme in [5] applies independent wavelet-based coding to the I frames of a geometry video and predicts the P frames from the previous frame by an affine transformation. The connectivity information for frames need not be encoded or transmitted since it can be taken as regular on the grid structure of the image.

The cut-path finding and parametrization steps of generating geometry video frames are enhanced in [18] to attain low texture density cut-paths and minimize a texture density weighted geometric stretch metric. Acquired, preprocessed motion data for 3D dynamic facial expressions is mapped to an expression invariant parametric domain in [55,56] by cutting the sequence frame to get two boundaries, mapping its outside boundary to the outside boundary of a genus-0 domain via arc length parametrization and mapping its interior to the interior of the domain using integral curves of harmonic functions of the frame and the domain. The resulting geometry video keeps exact correspondence of salient features (eye, mouth and nose) in different frames and is coded with a H.264/AVC encoder [53] having extra predictive modes suited to the spatial and temporal correlation characteristics of geometry videos. In [20,22], the expression invariant parameterization is used to generate geometry videos. While [20] directly applies model-based bit allocation by rate–distortion optimized quantization parameter selection to the coding of each coordinate (x, y, or z) video by H.264/AVC coder, [22] also decomposes each coordinate video by low-rank and sparse matrix decomposition before the bit allocation. Conformal geometry videos (CGV) [39], based on the global conformal parametrization approach of [14,15], exploits the isometric nature of 3D articulated motions. Due to isometry, the conformal factor images are approximately pose invariant and can be subsampled at key frames. Compression of CGV is performed in [39] by separately encoding this low frame rate sequence of conformal factor images and the full frame rate sequence of mean curvature images. Holovideos [25,26] extend the Holoimage technique of [16] to range frame sequences where each range frame is obtained by depth maps based on virtual fringe patterns formed by three waves of structured light with different phase shifts reflected off of model's surface. The holovideo is coded by a H.264/AVC-based coder after the height and 2D fringe are input to the Y and UV channels in [26]. The keyframe-based geometry video (KGV) approach [21] compresses geometry videos of animated human mesh sequences by iteratively extracting keyframes to minimize the linear interpolation error of the remaining frames, reordering them to maximize the sum of structural similarities between adjacent pairs and compressing them by a H.264/AVC coder whose parameters are selected by model-based rate–distortion optimization.

3 Animated mesh sequence coding system

3.1 Geometry image and geometry video

The geometry image generation scheme in [13] forms an initial cut on a 3D irregular mesh by first removing a seed triangle from the mesh and then iteratively removing an edge incident to one triangle and that triangle until all triangles are removed and finally iteratively removing a vertex incident to one edge and that edge until all such edge trees are trimmed. This initial cut is improved in iterations. In each iteration, the maximum geometric stretch point of Floater parametrization [12] is identified on a disk like domain, a hole is pinched through this point, the mesh is cut along shortest path that connects the hole to the current boundary and the cut path is merged with the boundary. The mesh boundary thus formed by cutting is exactly mapped onto the boundary of a 2D rectangular domain and geometric stretch parametrization is performed on this domain. Finally, the rectangular parametrization domain is uniformly sampled on a grid to get the (R,G,B) pixel coordinates of the image that represent the (x,y,z) coordinates of vertices on the surface of the 3D model. The surface can simply be triangulated by regularly splitting each rectangular quad into two triangles. Geometry images facilitate 2D image compression algorithms like those based on wavelet transforms to be applied to the problem of 3D mesh compression.

The parametrization in [13] directly minimizes geometric stretch by applying the hierarchical, progressive mesh refinement approach in [42]. The coarse-to-fine refinement is based on line search minimization along a randomly chosen direction [43] for optimizing vertex locations in the parameter domain. Although geometry images can also be obtained by sampling conformal parametrization domains, the scope of the current work is limited to the geometry images obtained by geometric stretch parametrization. Topologies of genus-0 complex models, like the ones used in the experiments of the current work, need to be nontrivially modified if area distortion of conformal mapping is to be mitigated.

The three stage process for obtaining a geometry image is illustrated in Fig. 1.

Geometry videos [5] extend geometry images by determining a single cut and a single parameterization common to a set of multiple consecutive frames of an animated mesh sequence. Since temporal correspondences between these frames are maintained by having a common cut and common parametrization, good compression performance can be

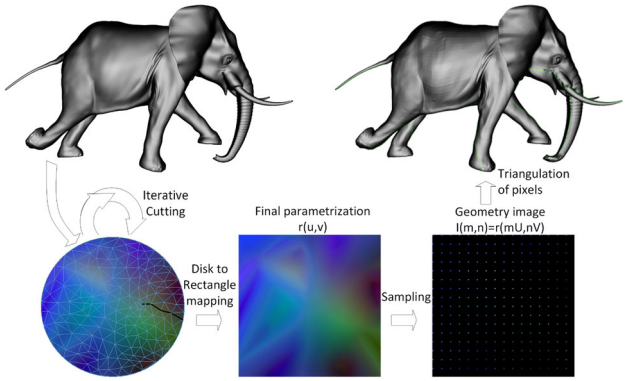


Fig. 1 Geometry image generation: iterative cutting, parametrization and sampling on a regular grid

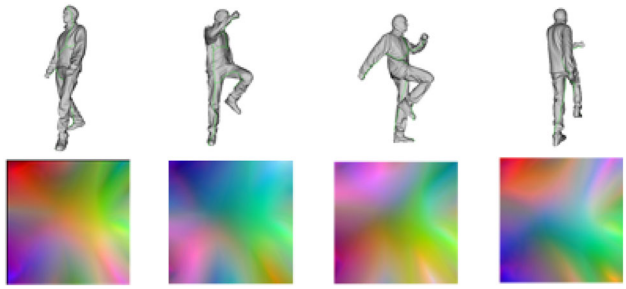


Fig. 2 Geometry video as a sequence of geometry images

attained by exploiting the temporal correlation in the geometry video domain.

The global cut is formed iteratively in a manner similar to the cut for a single geometry image. However, at each iteration, a hole is pinched at a vertex of a face having maximum average normalized stretch where the average is performed over the set of multiple consecutive frames. The cut path from the hole to the boundary is determined by applying Dijkstra's algorithm on a proxy mesh whose edge lengths are averages of lengths of corresponding edges of the frames in the set. The iterations are stopped when adding a cut would increase the average of the average distortion across all frames in the set.

As pointed out in [5], an arbitrary frame in the set is employed to compute a single geometric stretch parameterization for all of the frames in the set since this approach introduces only slightly larger parametrization error than a computation based on all the frames. Figure 2 illustrates a geometry video.

3.2 3D subband decomposition and coding

3D subband transform [facilitated by 3D DWT (Discrete Wavelet Transform)] coding is a highly efficient way of exploiting the spatial and temporal correlations among video pixels. Since the representation of an animated mesh sequence in the form of a geometry video is structurally similar to natural video, it is of interest to apply such coding techniques to geometry videos.

Several features of geometry videos are exploited by 3D wavelet-based coding in obtaining high coding efficiency. Firstly, the colors of adjacent geometry image pixels tend to be similar for a smooth surface, and 2D wavelets can represent smooth functions on a 2D domain in a very compact manner. Secondly, global cutting and parametrization yields time consistent geometry videos and the motion trajectory of each pixel (vertex) is a smoothly varying curve in 3D space over time. More importantly, the motions of neighboring vertices are not chaotic, but typically vary smoothly over the 3D surface. Application of a 3D DWT compacts data into lower dimensional subspaces by exploiting these temporal and spatial redundancies. A strong inter-frame correlation of corresponding geometry video pixel components (vertex coordinates) results in energy compaction into the temporal low-frequency subbands, whereas a strong intra-frame correlation of pixel (vertex) motion trajectories results in energy compaction into the spatiotemporal low-frequency subbands.

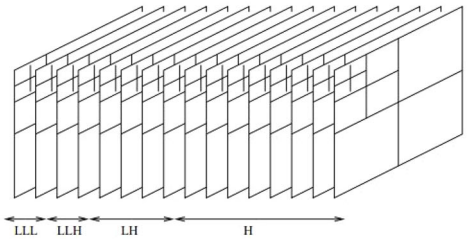
The 3D DWT is a straightforward extension of the 2D DWT to the spatiotemporal domain. Each group of (consecutive) frames (GOF) of the geometry video is first 1D DWT transformed along the temporal axis and each temporal slice of the resultant 3D data is then transformed by spatial 2D DWT. The application order of the temporal 1D DWT and the spatial 2D DWT may be interchanged. Figure 3a illustrates the decomposition of a GOF into four temporal and ten spatial frequency subbands. The lowest frequency subband LLL...LLL contains the coarsest spatiotemporal information. Higher frequency subbands contain detail information at various granularities.

Like most existent 3D DWT implementations, 1D CDF (Cohen-Daubechies-Feauveau) 9/7 filters and Haar transform are employed for the separable implementation of spatial 2D DWT and the in place implementation of the temporal 1D DWT, respectively. Both the spatial and temporal DWT's have 4 levels. Symmetric boundary extension is implemented at frame boundaries in order to apply filter kernels at boundaries. Proper scaling factors (l_2 norms) are used for the subbands of each level to ensure that the 3D DWT transform is energy preserving and corresponding bits of coefficients of different subbands have the same expected contribution to the signal approximation.

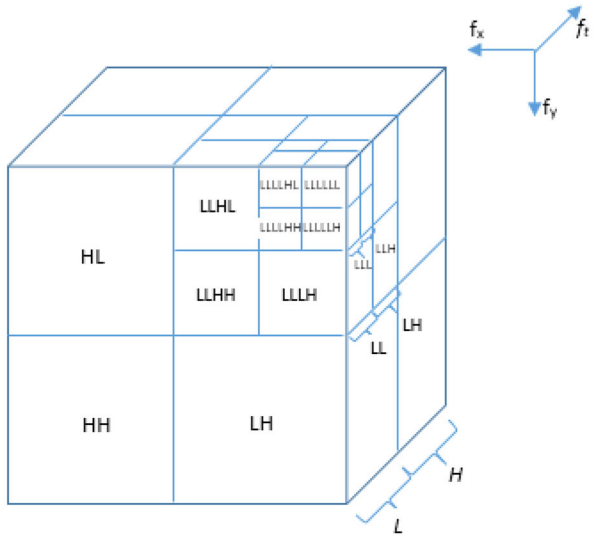
The block of 3D DWT coefficients of each GOF, shown in Fig. 3b, is processed by the 3D SPECK coder.

3.3 3D-SPECK algorithm

3D-SPECK [48] is an e xtension of the 2D-SPECK algorithm [35] for compressing 3D data with high coding efficiency.



(a) 1D temporal wavelet decomposition of a GOF and 2D spatial wavelet decomposition of each temporal slice



(b) The rectangular block of 3D DWT coefficients input to the 3D-SPECK coder

Fig. 3 .

The quadri-section partitioning in 2D-SPECK was extended to octo-section partitioning in 3D-SPECK in order to efficiently code the location of significant coefficients (that have magnitudes exceeding the threshold of the bitplane) with priority in the volumetric high-frequency bands.

3D-SPECK codes the 3D block of each GOF by indicating significant coefficients using partition sets and partitioning rules at each bitplane pass, and by successively refining them in later passes.

3.3.1 Type S sets and their partitioning

A 3D block of coefficients is termed a type S set. Let $c_{i,j,k}$ denote the DWT coefficient at coordinates i, j, k. The indicator function for the significance of a type S set with respect to the threshold 2^n for bitplane n is defined as

$$\Gamma_n(\mathcal{S}) = \begin{cases} 1 & \text{if } \max_{(i,j,k)\in\mathcal{S}} \|c_{i,j,k}\| \geq 2^n \\ 0 & \text{else} \end{cases} \quad (1)$$

If a type S set is deemed significant, quadtree or octree partitioning is recursively applied to locate and indicate all of the coefficients (single element type S sets) that are significant. This way the coefficients with large magnitude can be coded with priority. Two lists are maintained in 3D-SPECK:

- LIS (List of Insignificant Sets) contains type S sets of varying sizes that have not been found significant against a threshold yet.
- LSP (List of Significant Pixels) contains pixels found significant against a certain threshold.

Initially, the 3D block input to the coder is the only type S set in the LIS. The iterative significance testing for the type S sets in the LIS starts with the most significant bitplane $n = N$ and proceeds to the least significant bitplane $n = 1$ until the bit budget is exhausted. When a type S set in the LIS is deemed significant for bitplane n, it is removed from the LIS. Significant type S sets are recursively partitioned into offspring type S sets. Insignificant offspring type S sets are added to the LIS. Later in bitplane $n - 1$, all type S sets in the LIS are processed in the order of highest to lowest DWT level, lowest frequency to highest frequency subband within each level, and increasing size within each subband.

As illustrated in Fig. 4 and 5, the partitioning of a type S set $\mathcal{S}$ yields either 8 or 4 smaller offspring type S sets, $O(\mathcal{S})$. Each offspring $o \in O(\mathcal{S})$ has exactly or approximately half the size of the parent set in each dimension (with $|\mathcal{S}_t|$ being the size along the temporal axis of $\mathcal{S}$, if $|\mathcal{S}_t| = 1$, temporal partitioning is not applied). The successive partitionings can be represented as a tree structure with octal splitting at the high levels and quad splitting at the lower ones. For a 512x512x16 codeblock, the highest 4 levels have octal splitting (both spatial and temporal partitioning) and the lowest 5 levels have quad splitting (only spatial partitioning).

3D-SPECK transmits with priority the large magnitude coefficients of the small size subbands at the top-level of the wavelet decomposition that account for a significant portion of the energy of the 3D block. In return for the large reduction in distortion achieved by coding these coefficients, the expenditure in rate is minimal since the algorithm indicates these coefficients in a few partitioning steps. However, in later bitplane passes, when the algorithm processes large size significant type S sets at the lower levels of the wavelet decomposition, substantially more bits need to be expended to code the location of each significant coefficient in such sets. Hence, the algorithm exhibits a graceful distortion vs. rate trade-off.

3.3.2 The algorithm

The main body of 3D-SPECK is made up of an initialization pass and bitplane passes. Each bitplane pass consists of a sorting pass, a refinement pass and a quantization pass as detailed in Algorithm 1.

Algorithm 1 3D-SPECK Algorithm

1. **Initialization**
   - Output $n = \lfloor \log_2(\max |c_{i,j,k}|) \rfloor$
   - Set $LSP = \emptyset$
   - Set LIS = {all codeblocks of wavelet coefficients}
2. **Sorting Pass**
   - Order sets in LIS in increasing size
     for all set $\mathcal{S} \in LIS$ **do** ProcessS($\mathcal{S}$)
     end for
3. **Refinement Pass**
   - **for all** $c_{i,j,k} \in LSP : c_{i,j,k}$ not added in bitplane pass n **do**
     Output n'th MSB of $|c_{i,j,k}|$.
     end for
4. **Quantization Pass**
   - $n \leftarrow n - 1$
     Goto Sorting Pass

function ProcessS(S)
  Output $\Gamma_n(\mathcal{S})$

  if $\Gamma_n(\mathcal{S}) = 1$ **then**
    if $|\mathcal{S}| = 1$ **then**
      Output sign of $\mathcal{S}$
      $LSP = LSP \cup \mathcal{S}$
    else
      CodeS($\mathcal{S}$)
    end if
    $LIS \leftarrow LIS - \mathcal{S}$
  end if
end function

function CodeS(S)
  Partition $\mathcal{S} \rightarrow O(\mathcal{S})$
  (If $|\mathcal{S}_t| = 1$ then $|O(\mathcal{S})| = 4$
  else $|O(\mathcal{S})| = 8$)

  for all $o \in O(\mathcal{S})$ **do**
    Output $\Gamma_n(o)$
    if $\Gamma_n(o) = 1$ **then**
      if $|o| = 1$ **then**
        Output sign of o
        $LSP = LSP \cup o$
      else
        CodeS(o)
      end if
    else
      $LIS = LIS \cup o$
    end if
  end for
end function

In the sorting pass of bitplane n, function ProcessS first tests the significance of each type S set in the LIS by comparing the magnitudes of its coefficients against the threshold 2^n. Function CodeS recursively partitions a significant type S set having more than one coefficient into four or eight offspring subsets, $O(\mathcal{S})$, of approximately equal size (Figs. 4, 5) and immediately tests their significances until each significant type S set contains a single coefficient. Significant

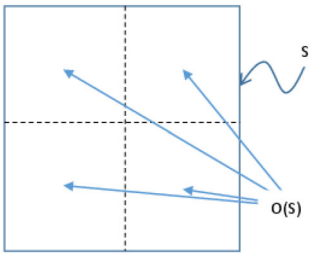


Fig. 4 A type S set and its partitioning into 4 smaller type S sets

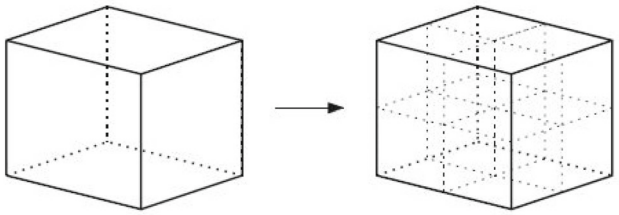


Fig. 5 A type S set and its partitioning into 8 smaller type S sets

single coefficient sets are added to the LSP. Insignificant sets are added to the LIS to be considered in the sorting pass of bitplane $n - 1$. The significance of each type S set and the sign of each significant single coefficient type S set are indicated by coding a single bit for each. ProcessS removes the significant type S set from the LIS after its processing is finished.

After all sets in the LIS are processed in the sorting pass, the refinement pass codes the n'th most significant bit of the magnitude of each coefficient in the LSP, excluding those included in the most recent sorting pass.

The decoder algorithm has the same flow path as the encoder algorithm. In the sorting pass of bitplane n, it reads the significance decision from the bitstream and reconstructs the magnitude of the significant coefficient value initially as

$$|\hat{c}_{(i,j,k)}| = (1.5 + b_n)2^n \tag{2}$$

where $b_n < 0$ is a bias value that accounts for the skewness of the magnitude density towards zero. In further refinement passes, the magnitude of the reconstructed coefficient value is refined as

$$|\hat{c}_{(i,j,k)}| = |\hat{c}_{(i,j,k)}| - b_{n+1}2^{n+1} + (0.5(2r_n - 1) + b_n)2^n \tag{3}$$

where r_n is the single bit coded for refinement at n'th bitplane pass. The coefficient value is formed by combining the reconstructed magnitude with the decoded sign bit.

3.3.3 Processing Order of Sets

Type S sets of varying sizes are added to the LIS during each bitplane sorting pass. Even though a coefficient being

significant in a small Type S set is somewhat more likely than one being significant in a large type S set, the bit expenditure to code the former is significantly lower than that to code the latter. Therefore, type S sets in LIS are tested for significance in increasing order of their sizes in sorting passes as per [2, 35]. This strategy resembles the prioritization of insignificant coefficients neighboring significant coefficients in fractional bitplane partitioning of coefficients for bitplane transmission in JPEG2000, known as significance propagation.

3.3.4 Entropy coding

The significance decisions for the four or eight offspring type S sets are jointly adaptive arithmetic coded [54] in the CodeS function. On the other hand, the single significance decisions of the type S sets in the ProcessS function as well as the refinement decisions are independently adaptive binary arithmetic coded.

4 Postprocessing of reconstructed surfaces

Even though the surfaces reconstructed after 3D-SPECK decoding and inverse 3D DWT approximate the original surfaces well, their close-up views reveal three kinds of visual artifacts caused by the geometry image transformation and subsequent 3D wavelet transformation.

The symmetric boundary extension used for spatial 2D-DWT results in substantial high-frequency energy content at the boundaries. When the 3D-DWT spatial high frequencies coefficients are culled with 3D-SPECK-based compression, sizable gaps are seen to separate the boundaries of the reconstructed surface. Secondly, the loss of mid-high-frequency subband coefficients results in undulations on the reconstructed surface. Finally, regular or quad splitting connectivity of local areas of the geometry image with large anisotropy of parametrization stretch appear as serrations on the originally smooth surfaces due to large dihedral angles between adjacent triangles. In the following, postprocessing solutions for each of these artifacts are presented.

4.1 Stitching of the boundary

The spatial 2D DWT employs boundary extension since the support of convolution kernels centered at or near boundary pixels extend outside the boundary. The mirroring boundary extension, typically used for natural RGB images during DWT analysis and the synthesis, yields a smooth boundary extension if the pixels near the boundary are correlated. However, for geometry images, at least one coordinate regularly changes with jumps in value as the pixels approach the boundary over a row or a column. At the boundary, mirroring of pixel coordinate values artificially introduces large magnitude high-frequency coefficients. These high-frequency coefficients do not pose a problem if their compression is lossless, but for lossy compression with an embedded wavelet coder like SPECK, they are the first ones to get culled. Small openings appear on the surface due to the separation of the two boundaries' segments that originally matched in 3D space.

This problem is addressed by stitching the entire boundary. In [13], the boundary vertices are one-to-one mapped to other boundary vertices using the topological sideband information. When this information is used to perform the stitching on the geometry images of the original, closed mesh, the result is a watertight geometry image representation of the mesh. However, in this work, we make provision for the case where such information is not available at the decoder and cannot be easily derived from the decoded coordinate values of the boundary pixels due to quantization noise.

We first map each boundary vertex to its nearest neighbor boundary vertex in 3D space. We allow for a boundary pixel to map to itself if its two neighbors map to each other. Then, we traverse the geometry image boundary one pixel at a time and determine the segments of consecutive pixels that map to segments of consecutive pixels. If two successive boundary pixels map to boundary pixels that differ by more than 3 pixel positions on the boundary, a segment is ended and the next segment is begun. Otherwise, if the difference is 0 pixels, two consecutive boundary pixels map to the same boundary pixel. One of the two images of the two pixels is shifted by one pixel so that the two images are consecutive pixels after the shift is applied. Of the two possible one pixel shifts (image of either boundary pixel is moved away from the image of its neighbor by one pixel), we prefer the one that results in the largest Euclidean distance distortion reduction for the mapping. Finally, if the difference is 2 pixels (two consecutive pixels map to two pixels with a pixel between them), the image of either pixel is shifted by one pixel (moved toward the image of its neighbor to cover the single pixel gap) so that, after the shift, the two images are consecutive. Again, the shift that results in the largest reduction in Euclidean distance distortion for the mapping is preferred. Figure 6 illustrates the cases for the two-to-one mapping (0-pixel difference) and the gap (2-pixel difference).

The segments of consecutive pixels are coded in two parts. In the first part, the number of segments followed by the minimum and maximum segment lengths are first fixed length coded with a sufficient number of bits. Then, for each segment, the difference of its length from the minimum segment length as well as the image of the first pixel of the segment are fixed length coded. In the second part, exceptions to consecutiveness of the mapping within each segment are coded. Exceptions occur since the above-mentioned one pixel shift corrections may inadvertently result in other overlaps of images of consecutive pixels or gaps between them.

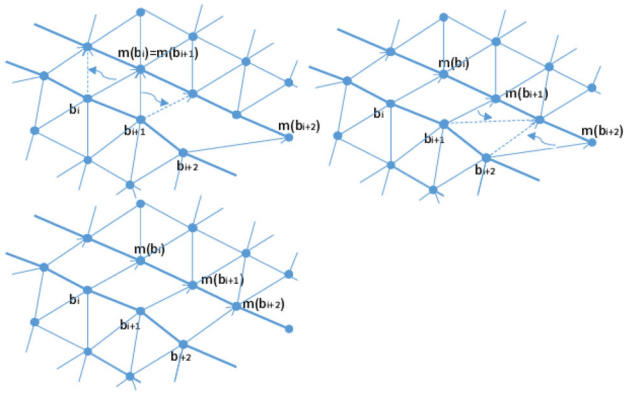


Fig. 6 Top left: The boundary pixels b_i mapping to boundary pixels $m(b_i)$ exhibits both a two-to-one map ($m(b_i) = m(b_{i+1})$) and a gap ($m(b_i)$ and $m(b_{i+1})$ have 3 pixel difference). Top-right: The two-to-one-map is resolved and gap is reduced to 2 pixel difference after $m(b_{i+1})$ is shifted by one pixel. Bottom-left: A second shift of $m(b_{i+2})$ resolves the gap completely

An exception occurs if map(b_i) and map(b_{i+1}) are not consecutive where b_i is the i'th boundary pixel. We indicate the position of each exception relative to the previous one to conserve bits. For each exception, we also code a single bit to indicate whether map(b_i) and map(b_{i+1}) are the same or have more than one pixel difference. In the latter case, we further indicate the difference over 1 by a Golomb code for $m = 1$.

The decoding end uses this coded side information to stitch the boundary by replacing the value of each coordinate of each boundary pixel by the arithmetic average of the value of the coordinate and the corresponding coordinate of the image pixel under the mapping. Since the final mapping is not one-to-one in general, the stitched boundary might still exhibit very tiny holes which are imperceptible for typical viewing distances.

4.2 Adaptive smoothing

4.2.1 Conditional smoothing

At low coding rates, the loss of mid-high-frequency DWT coefficients appears as undulations on the mesh surface. Smoothing can successfully improve visual quality by reducing such artifacts if it is selectively applied on the local regions free of significant surface features. To manage this problem, we divide the geometry image into uniform size blocks and indicate whether each block has significant features by coding a single bit.

Let $v_i \in \mathcal{N}(v)$ be the i'th neighbor of vertex v in the original irregular mesh. The normal curvature estimated at vertex v along edge $v\bar{v}_i$ may be expressed as $\kappa_i^N = \frac{2\overrightarrow{N}_v \cdot \overrightarrow{v_i v}}{\|v\bar{v}_i\|^2}$ where $\overrightarrow{N}_v$ is the per vertex normal at vertex v estimated as a normalized area weighted average over normals of faces incident to v. At each vertex v, we use the mean curvature estimate formed as the weighted average of the normal curvatures, $\sum_i w_i \kappa_i^N$, as a measure of the feature content at vertex v where the weights are $w_i = \frac{1}{8}(\cot\alpha_i + \cot\beta_i)\|v - v_i\|^2$ as in [32].

Next, the measure of feature content at each vertex (pixel) p of the original geometry image is determined by mapping p to the nearest point on the original irregular mesh. Towards this end, the original irregular mesh vertex c nearest to p is determined and one of the triangles incident to vertex c, $T = T(v_1, v_2, v_3)$ with $v_i = c$, is arbitrarily chosen. If the projection of the vector $\overrightarrow{cp}$ to the plane of T, $P(T)$, points inside T ($c + \text{proj}_{P(T)}(\overrightarrow{cp})$ lies on the correct side of each of the three edges of T), the measure of feature content at p is interpolated from the measures of the feature contents at v_1, v_2, v_3 where the interpolation weights are $w_i = \frac{\|\overrightarrow{pv_i}\|^{-1}}{\sum_{j=1}^{3}\|\overrightarrow{pv_j}\|^{-1}}$. We refrain from using barycentric interpolation here, since the correlation coefficient of the feature contents at p and v_i is commensurate with $\|\overrightarrow{pv_i}\|^{-1}$ and the best first order (linear) estimate employs the correlation coefficient as the coefficient of estimation. If a check for one of the three edges, say $v_k\bar{v}_l$, returns false, the triangle on the other side of $v_k\bar{v}_l$ is taken as new T, c is updated as one of its vertices and inclusion of $c + \text{proj}_{P(T)}(\overrightarrow{cp})$ in T is checked. This procedure is iterated until either a triangle includes the projection of point p as described above or the projection of p falls on the wrong side of an edge $E = v_k\bar{v}_l$ for both triangles sharing E. In the latter case, by letting c be one of the two vertices of E, we determine whether $c + \text{proj}_{L(E)}(\overrightarrow{cp})$ falls on E or outside E where $L(E)$ is the line that contains E. If E contains $c + \text{proj}_{L(E)}(\overrightarrow{cp})$, we determine the measure of the feature content at p by interpolating the measures of the feature contents at the vertices v_k, v_l where the interpolation weights are $w_\gamma = \frac{\|\overrightarrow{pv_\gamma}\|^{-1}}{\|\overrightarrow{pv_k}\|^{-1}+\|\overrightarrow{pv_l}\|^{-1}}$ for $\gamma \in \{k, l\}$. If $c + \text{proj}_{L(E)}(\overrightarrow{cp})$ falls outside E, the measure of the feature content at pixel p is set equal to that at the nearest vertex of edge E (v_k or v_l). In case the original irregular mesh has boundaries and the edge check returns false for an edge that belongs to the boundary, a similar interpolation using the measures of feature contents at the two vertices of the edge is applied.

The significance of feature content within each block of geometry image pixels is decided by thresholding the total feature content within the block and conveyed from encoder to decoder by coding a single bit. For 512×512 geometry images, a block size of 4×4 gives a good trade-off between the side information rate due to block significance coding and the manifestation of blockiness artifact between the regions with and without significant feature content.

Directional Taubin smoothing with coefficients of $\lambda = 0.6307$ and $\mu = 0.6347$ is applied for 25 iterations to the

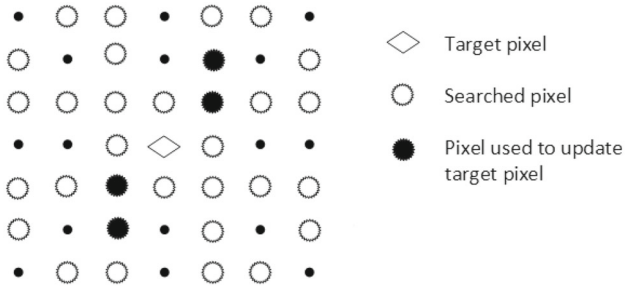


Fig. 7 Illustration of an example directional smoothing filter support. Only the distances of certain pixels (shown as hollow and filled circles) in the local neighborhood of the target pixel (shown as hollow diamond) are calculated, and the two best pair of pixels (shown as filled circles) symmetrically opposite with respect to the target pixel are used to update the given pixel

non-boundary geometry pixels in the blocks that have below threshold feature content.

4.2.2 Directional Taubin smoothing

Smoothing typically relies on a low-pass filtering operation that incurs the least loss of feature detail when the filter support covers the pixels that are most correlated with the target pixel. Unlike natural imagery, the pixels most correlated with a target pixel of a geometry image are not necessarily its closest neighbors in the parameter domain, but are those at the shortest 3D distance from it.

Therefore, the filter support for a target pixel is varied according to the local neighborhood of the target pixel in 3D space. Specifically, among the pixels in a 7×7 subgrid centered at the target pixel, the pair of pixels that are symmetrically opposite with respect to the target pixel and has the smallest total 3D Euclidean distance to the vertex represented by the target pixel is identified. To keep the search complexity low, among all the pixels lying on the same line in the parameter domain, only the two nearest the target pixel in the parameter domain are considered. For example, among the pixels with offset $(-3, -3)$, $(-2, -2)$, $(-1, -1)$, $(1, 1)$, $(2, 2)$ and $(3, 3)$ from the target pixel in the 7×7 subgrid, only the ones with offset $(1, 1)$ and $(-1, -1)$ are considered.

A second pair is also similarly determined such that it has the smallest total 3D distance to the target pixel among the pixels remaining in the 7×7 subgrid. The four pixels are then used to form the update term in Taubin smoothing [49].The example illustrated in Fig. 7 shows pixels with offsets $(1, 1)$, $(-1, -1)$, $(1, 2)$ and $(-1, -2)$ from the target pixel as forming the update term.

4.3 Stretch adaptive triangulation

The simple connectivity approach of [13] for a geometry image that triangulates the grid of pixels by splitting each quad of 2×2 pixels adaptively along the shorter diagonal is prone to generate skinny triangles in 3D space when the eigenvalues of the local surface metric tensor resulting from the 2D parametrization of the surface differ significantly [3]. Two such skinny and adjacent triangles can have a large dihedral angle between them and a succession of such triangles formed by the quad splitting connectivity can appear as serration (alternating valleys and ridges) on originally smooth surfaces if surface curves have torsion components [3].

As a solution, [3] proposes a greedy region growing algorithm in the parameter domain for triangulating the pixels to achieve more optimal connectivity of the pixels by compensating for the anisotropy of the underlying local stretch and variations in the first principal direction of the metric tensor. The region is iteratively grown in such a way that the maximum length of each newly added triangle along the first principal direction is small. Specifically, at each iteration, the triangulated 2D region is grown by the addition of the triangle that is the best (minimizing a cost function) among the topologically valid candidate triangles sharing an edge with the boundary of the grown region from the previous iteration. Following the addition of this triangle, the best (minimizing the same cost function) topologically valid candidate triangles bordering all new edges of the added triangle are determined and added to the list of candidate triangles to be considered in the next iteration. The algorithm resembles the advancing front surface reconstruction scheme of [9] for point clouds. However, not only does it triangulate a preexistent set of vertices, but also these vertices are ordered in the form of a grid that guides the triangulation. The resulting reconstructions are both smoother compared to reconstructions based on regular or quad splitting connectivity, and also free of the artifacts seen with unguided triangulation in [9] due to the incorrect joining of the vertices on different sides of thin or narrow parts of the model.

The pseudocode of the greedy region growing algorithm of the triangulation approach applied in the last postprocessing step is given in Algorithm 2 where $\mathbf{r}(v)$ denotes the vector of coordinates of vertex v, $T_{v1,v2,v3}$ denotes a triangle with vertices $v1$, $v2$, $v3$, $he(v0, v1)$ denotes the half edge directed from $v0$ to $v1$, $C(T)$ denotes the cost of adding triangle T to the grown region. We use $C(T_{v0,v2,v1}) = \|\mathbf{r}(v2) - \mathbf{r}(v0)\|^2 + \|\mathbf{r}(v1) - \mathbf{r}(v2)\|^2$. After a triangle is added to the grown region, up to three possible triangles (options) are considered at each of the new half edges of the region's boundary before the best is selected as the candidate triangle at that half edge. The candidate triangle is checked for topological validity (in the parameter domain, its half edges not intersecting the edges of the triangles in the region) as soon as it is formed as well as right before it is added to the grown region at a later time.

Algorithm 2 Triangulation of geometry image $\mathcal{G}$ with boundary $B(\mathcal{G})$ by growing region $\mathcal{R}$ ([3])

Split quads of $\mathcal{G}$ diagonally into right triangles $\{T_{\alpha,\beta,\gamma}\}$
$\mathcal{R} = \arg_{T_{\alpha,\beta,\gamma}} \min\{\|\mathbf{r}(\alpha) - \mathbf{r}(\beta)\|$
$+\|\mathbf{r}(\beta) - \mathbf{r}(\gamma)\| + \|\mathbf{r}(\gamma) - \mathbf{r}(\alpha)\|\}$
repeat
  if first iteration **then**
    $T_{v^*,v^{***},v^{**}} = \mathcal{R}$
  else
    $T_{v^*,v^{***},v^{**}} = \arg_{T_{v0,v2,v1}:he(v0,v1)\in\breve{B}} \min \tilde{C}(v0, v1)$
    where $T_{v0,v2,v1} = \tilde{T}(v0, v1)$ is the candidate on
    $he(v0, v1)$ and $\breve{B} = B(\mathcal{R}) \cap B(\bar{\mathcal{G}})$
  end if
  if $\min \tilde{C}(v^*, v^{**}) < \infty$ **and** EDGESVALID($T_{v^*,v^{***},v^{**}}$) **then**
    $\mathcal{R} \leftarrow \mathcal{R} \cup T_{v^*,v^{***},v^{**}}$, $v0 = v^*$,$v1 = v^{**}$,$v2 = v^{***}$
    for all $he(v^p, v^q) \in \{he(v0, v2), he(v2, v1)\}$ **do**
      if $he(v^q, v^p)$ exists **then** ▷ Ear/Hole filling
        $he(v^p, v^q) \cup he(v^q, v^p) \rightarrow e(v^p, v^q)$
      else ▷ Propagation/Gluing
        $C_a = C_b = C_c = C_d = \infty$
        if $|v_y^p - v_y^q| == 1$ **then**
          $v_y^a = v_y^q$, $v_x^a = v_x^q + S_x$ ▷ Option a
          if EDGESVALID(T_{v^p,v^a,v^q}) **then**
            $C_a = C(T_{v^p,v^a,v^q})$
          end if
          $v_y^b = v_y^p$, $v_x^b = v_x^p + S_x$ ▷ Option b
          if EDGESVALID(T_{v^p,v^b,v^q}) **then**
            $C_b = C(T_{v^p,v^b,v^q})$
          end if
        end if
        if $|v_x^p - v_x^q| == 1$ **then**
          $v_x^c = v_x^q$, $v_y^c = v_y^q + S_y$ ▷ Option c
          if EDGESVALID(T_{v^p,v^c,v^q}) **then**
            $C_c = C(T_{v^p,v^c,v^q})$
          end if
          $v_x^d = v_x^p$, $v_y^d = v_y^p + S_y$ ▷ Option d
          if EDGESVALID(T_{v^p,v^d,v^q}) **then**
            $C_d = C(T_{v^p,v^d,v^q})$
          end if
        end if
        Best option $n = \arg_{m\in\{a,b,c,d\}} \min C_m$
        $\tilde{T}(v^p, v^q) = T_{v^p,v^n,v^q}$, $\tilde{C}(v^p, v^q) = C_n$
      end if
    end for
    $he(v0, v1) \cup he(v1, v0) \rightarrow e(v0, v1)$
    $\tilde{C}(v0, v1) = \infty$
  end if
until $T_{v^*,v^{***},v^{**}}$ does not exist ($\min \tilde{C}(v^*, v^{**}) = \infty$))
▷ $S_x \in \{-1, 1\}$ and $S_y \in \{-1, 1\}$ so that $v^* \rightarrow v^{**} \rightarrow v^{***}$ has CCW order

Algorithm 3 Validity check for new triangle.

function EDGESVALID($T_{v^*,v^{***},v^{**}}$)
  if $he(., v^*) \cap he(v^{***}, v^{**}) \neq \emptyset$ **then return** False
  end if
  if $he(v^{**}, .) \cap he(v^*, v^{***}) \neq \emptyset$ **then return** False
  end if
  for all T_{v^0,v^{***},v^1} **do**
    if $T_{v^0,v^{***},v^1} = T_{v^*,v^{***},v^{**}}$ **then return** False
    end if
    if $he(v^1, v^0) \cap he(v^*, v^{***}) \neq \emptyset$ or
      $he(v^1, v^0) \cap he(v^{***}, v^{**}) \neq \emptyset$ **then return** False
    end if
  end for
  return True
end function

5 Experimental results

In this section, the performance of the proposed animated mesh sequence compression system is evaluated. Experiments have been conducted on five animated mesh sequences representing human motion, namely March1 (240 frames), Jumping (144 frames), March2 (240 frames), Handstand (112 frames), Squat1 (112 frames) taken from [47] and two animated mesh sequences representing animal motion, namely Elephant (48 frames) and Snake (128 frames) taken from [51]. All animated sequences have been broken up into Groups of Frames (GOF) of size of 16. The geometry video has been obtained on a 512×512 grid as described in Sect. 3.1 by employing the OpenGI platform [40] that employs the geometric stretch parametrization of [58]. The distortion between the original animated mesh sequence and the reconstructed geometry video sequence is reported in terms of the average of the RMS (Root-Mean-Squared) distances between the corresponding frames of the original mesh sequence and reconstructed geometry video. Coding rate is reported as bits per vertex (bpv) for each frame.

Let M_i and M_i^r be the i'th frames of the original and reconstructed animated mesh sequences, respectively, and v and v^r be their generic vertices. The average forward RMS

$$\bar{\text{RMS}}^f = 1/N \sum_{i=1}^{N} \text{RMS}^f(M_i, M_i^r) \tag{4}$$

is based on the forward RMS between M_i and M_i^r which is defined as

$$\text{RMS}^f(M_i, M_i^r) = \left(\frac{1}{\text{area}(M_i)} \int_{M_i} \min_{v_r \in M_i^r} \|v - v_r\|^2 dv \right)^{1/2}$$

The average backward RMS, $\bar{\text{RMS}}^b$, is similarly defined so that $\text{RMS}^b(M_i, M_i^r) = \text{RMS}^f(M_i^r, M_i)$. Both $\text{RMS}^f(M_i, M_i^r)$ and $\text{RMS}^b(M_i, M_i^r)$ are computed using the METRO tool [8]. The average RMS, $\bar{\text{RMS}} = 0.5(\bar{\text{RMS}}^f + \bar{\text{RMS}}^b)$, is reported.

Performance comparison of the proposed compression method has been made with several other prominent methods: MPEG-4 Frame Animated Mesh Coding method (FAMC) [29], Geometric Laplacian Coding method (GLCoder) [52], Keyframe Geometry Videos (KGV) method [21], MPEG Video-based Point Cloud Coding (MPEG-VPCC) method [45] and x265 implementation of H.265/HEVC coding (Main

4:4:4 12, preset = veryslow) of geometric stretch geometry videos. The operational distortion–rate curves for the seven test sequences coded with these methods are displayed in Fig. 8.

Implementations of FAMC [31] and GLCoder [50] have been employed with their default settings. The keyframes of FAMC coder have been coded by the TFAN [27] based O3DGC coder. For the KGV method, the graphical results on some of the sequences published in [21] have been reproduced.

Comparisons have also been made with a recent implementation [41,45] of the recently developed MPEG-VPCC (MPEG Video-based Point Cloud Compression) standard. In MPEG-VPCC, the point cloud sequence is converted to a geometry video by segmenting each frame's point cloud into patches and tightly packing the patches into a geometry image. Padding is applied to each frame to increase the coding efficiency before the geometry video is coded by a standards-based video encoder like HEVC. An occupancy map as well as patch info are transmitted as side-information.

In the experiments on MPEG-VPCC, point clouds with point numbers in the range 240,000–260,000 have been generated by randomly sampling [6] the faces of each frame of a mesh sequence to generate the point cloud sequence. As with the geometry video of the proposed method, the grid resolution of 512×512 is employed in the geometry videos of MPEG-VPCC. These parameter settings have yielded the best distortion–rate performance for the MPEG-VPCC coder on wide ranges of tested values. The CGAL (Computational Geometry Algorithms Library) surface reconstruction implementation [11] with the greatest fidelity has been employed for reconstructing the animated mesh surfaces from decoded point cloud sequence. Only the sizes of the geometry and occupancy bitstreams for each GOF are used to compute the bitrate for the overall sequence.

It can be concluded that the proposed SPECK coder achieves the best performance in the low-to-middle rate range when compared to the FAMC [29], GLCoder [52] and KGV method [21]. At high rates, the proposed compression method reduces the error in the geometry video down to zero, but cannot reduce the remeshing error (between the original mesh sequence and original geometry video). Remeshing errors have been computed in terms of $\bar{RMS}$ for the sequences Handstand, Squat and Elephant as 0.00017, 0.000165 and 0.000168, respectively, to give an idea of the limiting performance of the proposed system at high rates.

The proposed method is superior to the MPEG-VPCC method in distortion–rate performance as well visual quality at all rates. The inferiority of MPEG-VPCC can be attributed to the gaps among the patches in the geometry images formed by this method as well as the application of the H.265/HEVC standards-based video coder that has been designed for natural imagery, but not surface geometry.

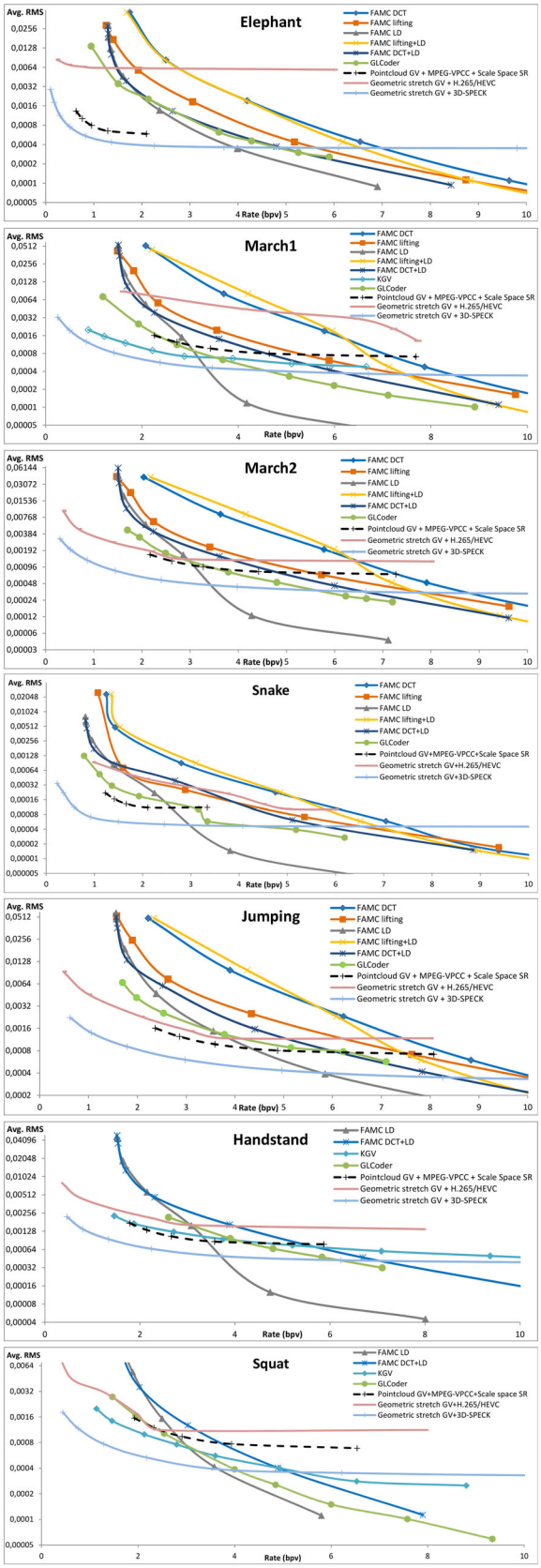


Fig. 8 Distortion ($\bar{RMS}$) versus Rate (bpv) for the test sequences

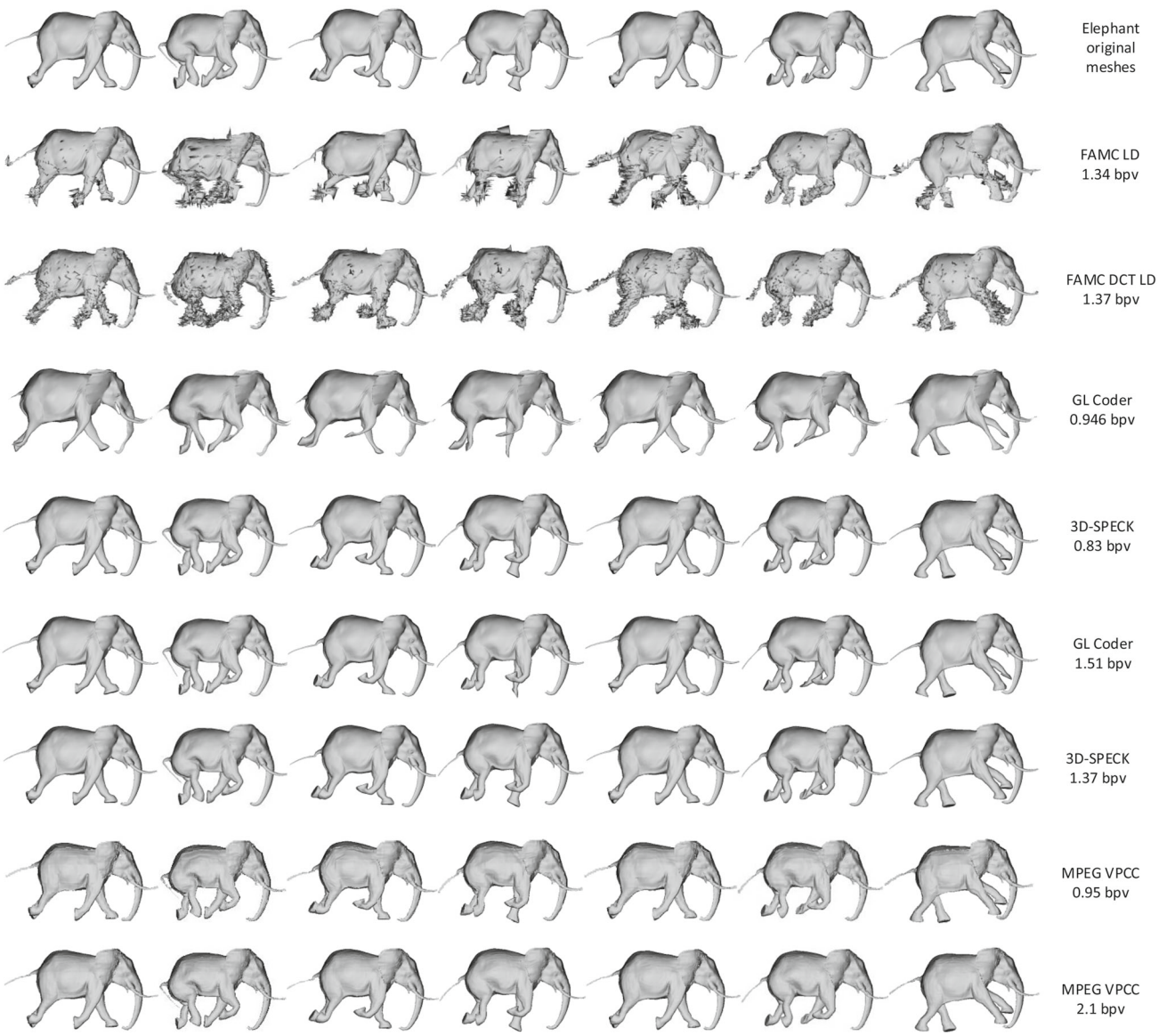


Fig. 9 Elephant

From each coded sequence, snapshots of several frames are displayed in Figs. 9, 10, 11a, b, 12. It can be seen that both the GLCoder and the proposed SPECK coder achieves much better visual quality in reconstruction than the FAMC coder at all bitrates shown. The visual distortions for the SPECK coder and GLCoder are only distinguishable at the lowest bit rates. At those rates, it is typical that the SPECK coder yields higher fidelity than the GLCoder for the macroscale features of the models that translates to a visual quality difference for almost all viewing distances of the reconstructed models. On the other hand, the GLCoder appears to perform better for small scale features of the models that are only perceptible when the reconstructed models are viewed close up.

Figure 13 shows the reconstructions of the head section of the Elephant model after the geometry video remeshing, 3D-SPECK coding and the postprocessing stages. In Fig. 14, close up snapshots of the hind legs of the model before and after boundary stitching postprocessing stage shows the closing of the gaps at the boundary.

The postprocessing operations do not improve the distortion-rate performance of the coding algorithm and the second one even reduces distortion-rate performance at times. However, the improvement in visual quality is noticeable.

The coding efficiency of 3D-SPECK is reduced when less energy is compacted into the low frequency temporal subbands as is typically the case with lower frame rates.

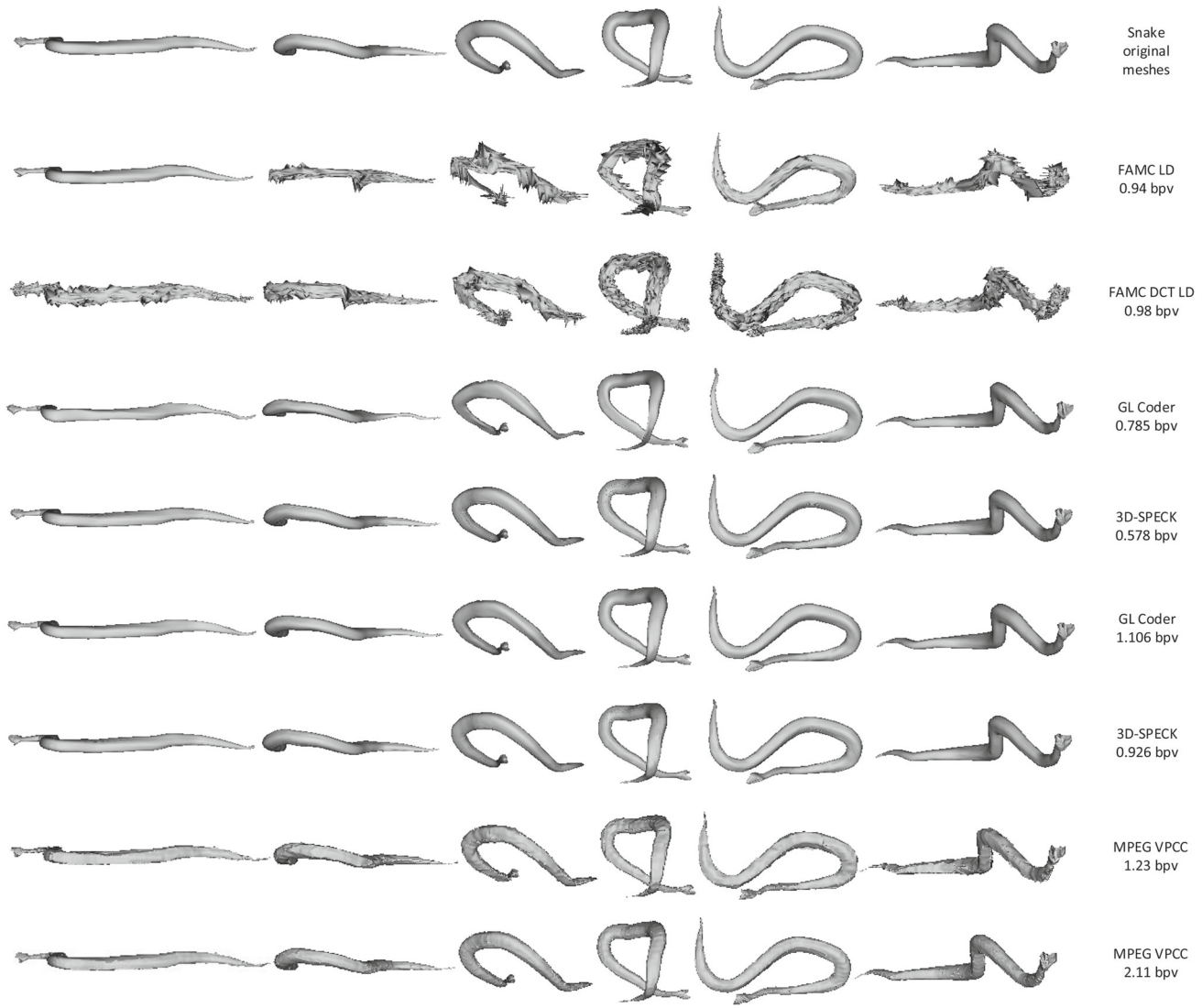


Fig. 10 Snake

Figure 15 shows the performance comparisons of the proposed method and the GLCoder at low frame rates achieved by 4:1 and 3:1 temporal subsampling of the Snake and Elephant sequences, respectively. The proposed method has less distortion–rate performance advantage in the low frame rate scenario. This can be attributed to the increase in energy in the temporal high-frequency subbands. Nevertheless, a low frame rate is not desirable in most applications since it results in jerky visualization of moving parts of the model.

3D-SPECK also reproduces sharp features (having large curvature) with less priority and fidelity compared to the other smooth parts of the model at low rates. Figure 16 shows the colorization of the distances of the vertex points of the Elephant0000 mesh reconstructed by the proposed method and GLCoder from the original mesh at two different rates. It can be seen that the proposed method reconstructs smooth regions with more fidelity than regions with large curvature features like creases on the skin of the model. On the other hand, while the compared GLCoder reproduces these features with higher fidelity, it yields large errors in the reconstruction of large smooth regions. This can be attributed to the GLCoder relying on the correlation of rigid motion between neighboring vertices which need not be valid for regions undergoing deformation. Since such low frequency errors are not consistent over frames, they appear as sudden and jerky motion during animation.

Finally, Table 1 reports the run time complexity figures of the different components of the proposed method along with the corresponding figures for the GLCoder. The figures have been obtained for the March sequence on an Intel Core i7-3520M 2.9GHz CPU with 8GB RAM. We note that no software optimization or acceleration by parallelization

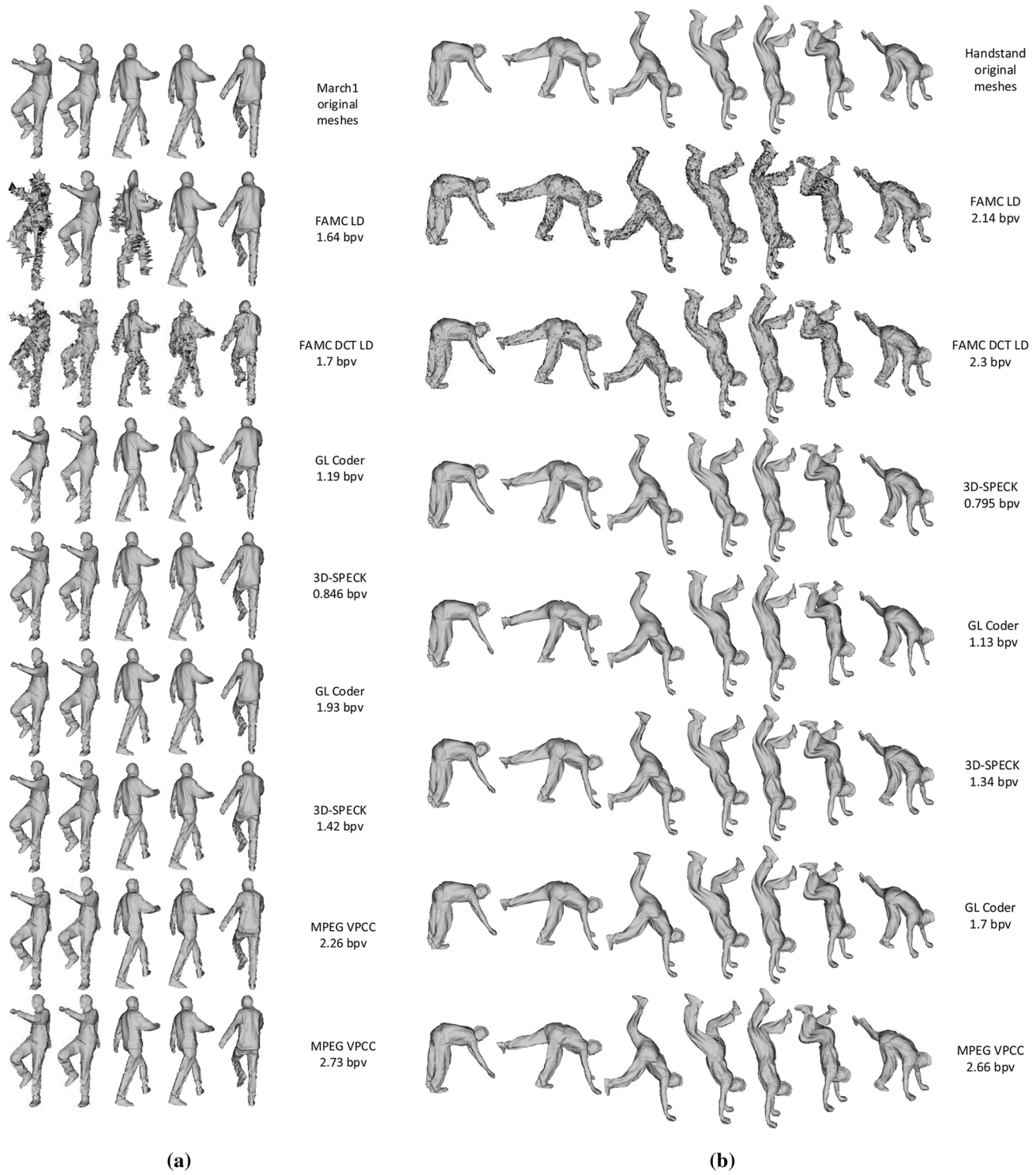


Fig. 11 **a** March, **b** handstand

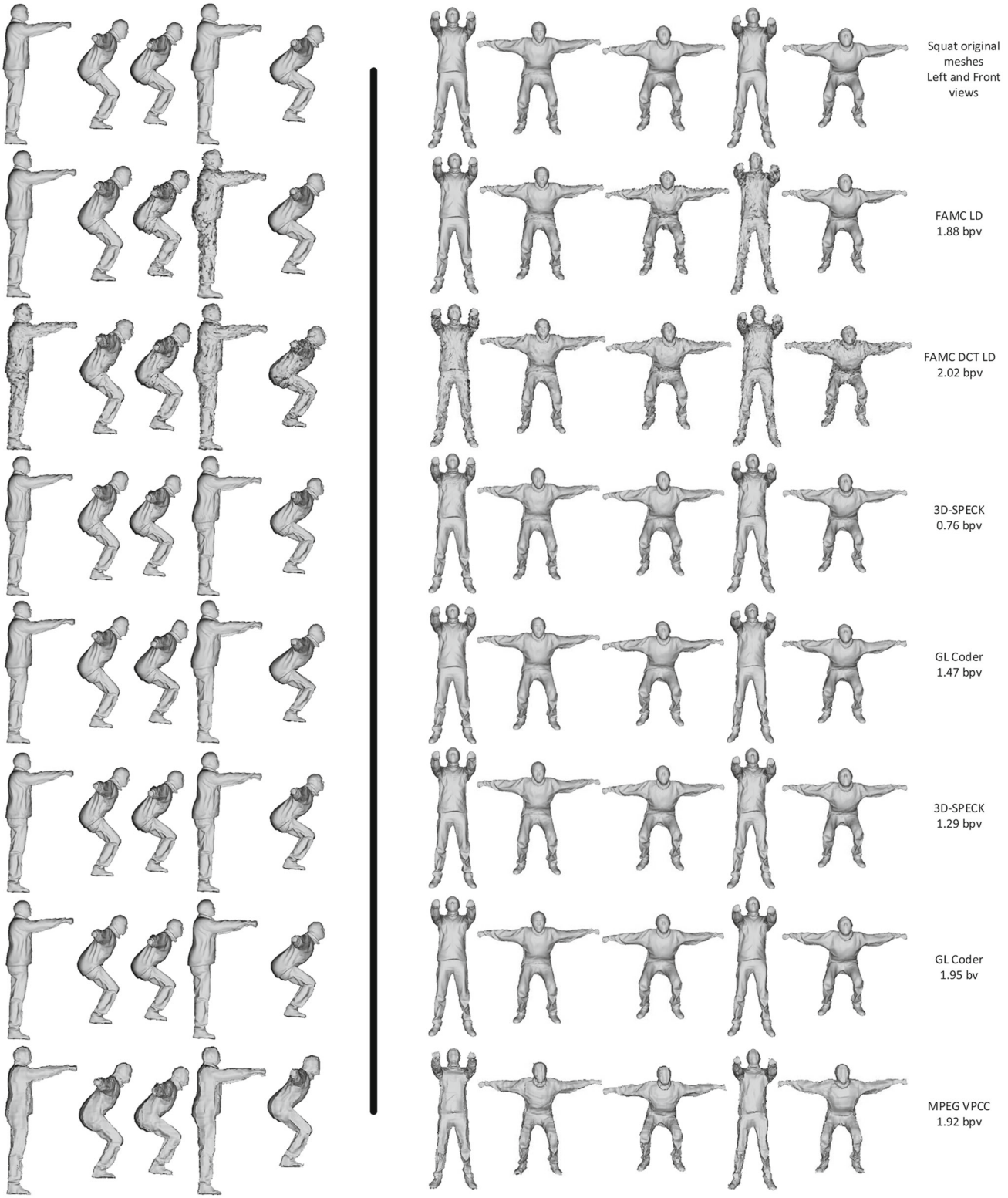


Fig. 12 Squat (front and left views)

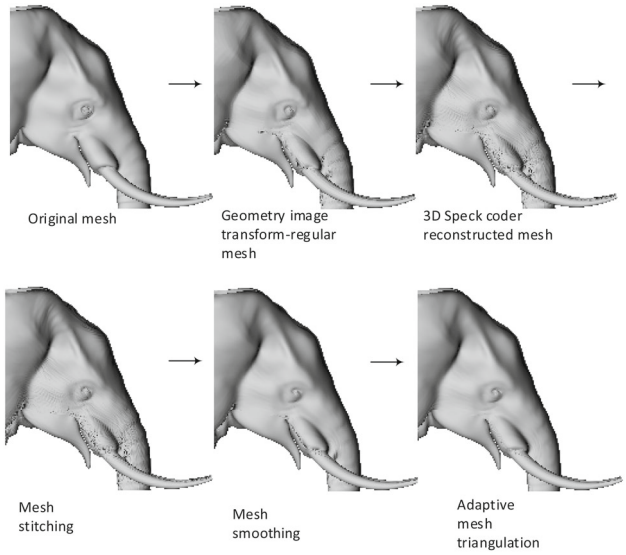


Fig. 13 Head of Elephant after each postprocessing stage

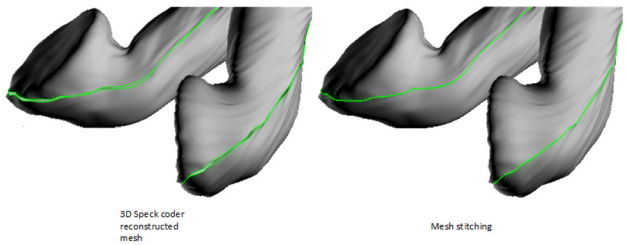


Fig. 14 Closing of hole type artifacts at geometry image boundaries by stitching

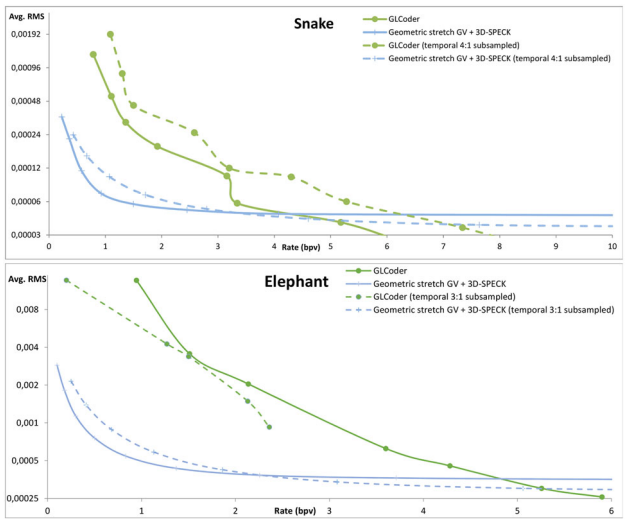


Fig. 15 Effect of temporal subsampling on distortion (RMS) versus rate (bpv) performance for the test sequences

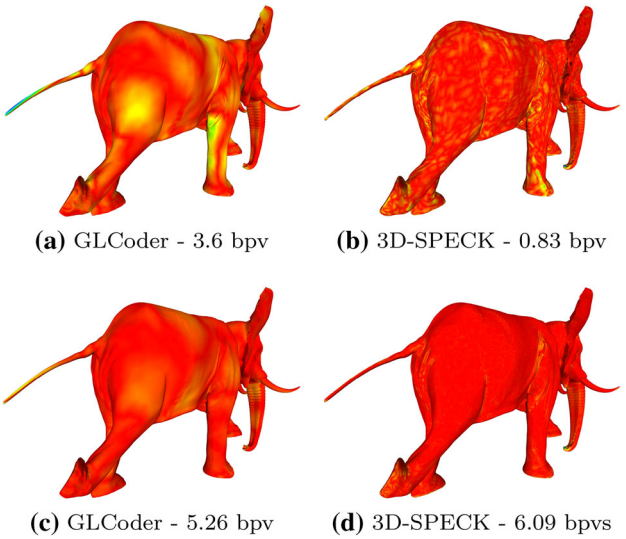


Fig. 16 Colorization of distances of the reconstructed vertices from original Elephant0000 mesh for the two methods. Distance values are color coded from small to large as red → yellow → green → cyan → blue

has been performed on the current implementation of the proposed method. The figures show that the smoothing and triangulation stages of postprocessing need to be optimized for speed in the future.

6 Conclusions

As with sequences of natural images, animated sequences converted to geometry videos have energy compacted into their low frequency spatiotemporal wavelet coefficients. This paper proposed the application of 3D-SPECK coder, a distortion vs. rate performance efficient 3D wavelet transform coder that codes wavelet coefficients by employing set partitioning rules in order to exploit this property. Several postprocessing techniques needed to be applied at the decoding end to mitigate the visual artifacts in the forms of undulations on smooth surfaces, opening up of gaps on originally matching boundary pixels as well as jaggedness due to regular or quad-splitting triangulation of geometry images having locally anisotropic surface metric tensors.

In this work, we have not exploited the motion along vertex trajectories to increase the compression performance. In the future, we plan on developing a coarse to fine scale local coordinate transform that can be substituted in place of a motion compensation scheme to exploit rigid body motion. We are also currently in the process of devising a multithread implementation of the adaptive triangulation postprocessing step.

Table 1 Average per frame run time complexities (s) of different components of the proposed method (and GLCoder for reference)

	3D-SPECK encoder			3D-SPECK decoder			P1	P2	P3	GLCoder			GLDecoder		
bpv	0.5	2.38	6.70	0.5	2.38	6.70				0.5	2.0	7.0	0.5	2.0	7.0
Run time	6.05	6.22	6.83	3.69	3.92	4.27	3.70	311.42	1774.01	1.88	3.69	5.19	3.0	3.0	3.0

P1: stitching, P2: smoothing, P3: adaptive triangulation

Acknowledgements Mesh data used in this work was made available by Robert Sumner and Jovan Popovic, and by Daniel Vlasic, Ilya Baran, Wojciech Matusik, Jovan Popović from the Computer Graphics Group at MIT.

References

1. Alexa, M., Müller, W.: Representing animations by principal components. Comput. Graph. Forum **19**, 411–418 (2000)
2. Islam, A., Pearlman, W.A.: Embedded and efficient lowcomplexity hierarchical image coder. In: Visual Communications and Image Processing 1999. SPIE (1998). https://doi.org/10.1117/12.334677
3. Bayazit, U.: A greedy region growing algorithm for anisotropic stretch adaptive triangulation of geometry images. Graph. Models (2019). https://doi.org/10.1016/j.gmod.2019.101045
4. Boulfani Cuisinaud, Y., Antonini, M.: Motion-based geometry compensation for dwt compression of 3d mesh sequences. In: ICIP (1), pp. 217–220. IEEE (2007). http://dblp.uni-trier.de/db/conf/icip/icip2007-1.html#Boulfani-CuisinaudA07
5. Briceño, H.M., Sander, P.V., McMillan, L., Gortler, S.J., Hoppe, H.: Geometry videos: a new representation for 3d animations. In: Parent, R., Singh, K., Breen, D.E., Lin, M.C. (eds.) Symposium on Computer Animation, pp. 136–146. The Eurographics Association (2003). http://dblp.uni-trier.de/db/conf/sca/sca2003.html#BricenoSMGH03
6. Castro, D.d.l.I.: Pyntcloud (2019). https://pypi.org/project/pyntcloud/
7. Chew, B.S., Chau, L.P., He, Y., Wang, D., Hoi, S.C.H.: Spectral geometry image: image based 3d models for digital broadcasting applications. TBC **57**(3), 636–645 (2011)
8. Cignoni, P., Rocchini, C., Scopigno, R.: Metro: measuring error on simplified surfaces. Comput. Graph. Forum **17**(2), 167–174 (1998)
9. Cohen-Steiner, D., Da, F.: A greedy delaunay-based surface reconstruction algorithm. Vis. Comput. **20**(1), 4–16 (2004)
10. Collins, G., Hilton, A.: A rigid transform basis for animation compression and level of detail. In: Chantler, M. (ed.) Vision, Video, and Graphics. The Eurographics Association, Aire-la-Ville (2005). https://doi.org/10.2312/vvg.20051003
11. Digne, J., Morel, J.M., Mehdi-Souzani, C., Lartigue, C.: Scale space meshing of raw data point sets. Comput. Graph. Forum **30**, 1630–1642 (2011). https://doi.org/10.1111/j.1467-8659.2011.01848.x
12. Floater, M.S.: Parametrization and smooth approximation of surface triangulations. Comput. Aided Geom. Des. **14**(3), 231–250 (1997). https://doi.org/10.1016/S0167-8396(96)00031-3
13. Gu, X., Gortler, S.J., Hoppe, H.: Geometry images. ACM Trans. Graph. **21**(3), 355–361 (2002)
14. Gu, X., Wang, Y., Yau, S.T.: Geometric compression using Riemann surface structure. Commun. Inf. Syst. **3**(3), 171–182 (2004)
15. Gu, X., Yau, S.T.: Global conformal parameterization. In: Kobbelt, L., Schröder, P., Hoppe, H. (eds.) Symposium on Geometry Processing, ACM International Conference Proceeding Series, vol. 43, pp. 127–137. Eurographics Association (2003). http://dblp.uni-trier.de/db/conf/sgp/sgp2003.html#GuY03
16. Gu, X., Zhang, S., Huang, P., Zhang, L., Yau, S.T., Martin, R.: Holoimages. In: Proceedings of the 2006 ACM Symposium on Solid and Physical Modeling, SPM '06, pp. 129–138. ACM, New York, NY, USA (2006). https://doi.org/10.1145/1128888.1128906
17. Guskov, I., Khodakovsky, A.: Wavelet compression of parametrically coherent mesh sequences. In: Badler, N.I., Desbrun, M., Boulic, R., Pai, D.K. (eds.) Symposium on Computer Animation, pp. 183–192. The Eurographics Association (2004). http://dblp.uni-trier.de/db/conf/sca/sca2004.html#GuskovK04
18. Habe, H., Katsura, Y., Matsuyama, T.: Skin-off: representation and compression scheme for 3d video. In: in Proceedings of Picture Coding Symposium (PCS '04), pp. 301–306 (2004)
19. Hajizadeh, M., Ebrahimnezhad, H.: NLME: a nonlinear motion estimation-based compression method for animated mesh sequence. Vis. Comput. **36**(3), 649–665 (2020)
20. Hou, J., Chau, L.P., He, Y., Zhang, M., Magnenat-Thalmann, N.: Rate-distortion model based bit allocation for 3-d facial compression using geometry video. IEEE Trans. Circuits Syst. Video Techol. **23**(9), 1537–1541 (2013)
21. Hou, J., Chau, L.P., Magnenat-Thalmann, N., He, Y.: Compressing 3-d human motions via keyframe-based geometry videos. IEEE Trans. Circuits Syst. Video Technol. **25**(1), 51–62 (2015)
22. Hou, J., Chau, L.P., Zhang, Magnenat-Thalmann, N., He, Y.: A highly efficient compression framework for time-varying 3-d facial expressions. IEEE Trans. Circuits Syst. Video Technol. **24**(9), 1541–1553 (2014). https://doi.org/10.1109/TCSVT.2014.2313890
23. Ibarria, L., Rossignac, J.: Dynapack: space-time compression of the 3d animations of triangle meshes with fixed connectivity. In: Symposium on Computer Animation, pp. 126–135. The Eurographics Association (2003)
24. Karni, Z., Gotsman, C.: Compression of soft-body animation sequences. Comput. Graph. **28**(1), 25–34 (2004)
25. Karpinsky, N., Zhang, S.: Holovideo: real-time 3d range video encoding and decoding on GPU. Opt. Lasers Eng. **50**(2), 280–286 (2012). https://doi.org/10.1016/j.optlaseng.2011.08.002
26. Karpinsky, N., Zhang, S.: 3D range geometry video compression with the H. 264 codec. Opt. Lasers Eng. **51**, 620–625 (2013). https://doi.org/10.1016/j.optlaseng.2012.12.021
27. Mamou, K., Zaharia, T., Prêteux, F.: TFAN: a low complexity 3d mesh compression algorithm. J. Vis. Comput. Anim. **20**, 343–354 (2009). https://doi.org/10.1002/cav.319
28. Mamou, K., Zaharia, T.B., Prêteux, F.J.: A skinning approach for dynamic 3d mesh compression. J. Vis. Comput. Anim. **17**(3–4), 337–346 (2006)
29. Mamou, K., Zaharia, T.B., Prêteux, F.J.: FAMC: The MPEG-4 standard for animated mesh compression. In: ICIP, pp. 2676–2679. IEEE (2008). http://dblp.uni-trier.de/db/conf/icip/icip2008.html#MamouZP08
30. Marpe, D., Wiegand, T., Schwarz, H.: Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard. IEEE Trans. Circuits Syst. Video Technol. **13**(7), 620–636 (2003)
31. Mekuria, R.: MPEG reference software with OpenCTM for benchmarking MPEG graphics codecs and datasets from live reconstruction (2015). https://github.com/kmammou/openFAMC

32. Meyer, M., Desbrun, M., Schröder, P., Barr, A.H.: Discrete differential-geometry operators for triangulated 2-manifolds. Vis. Math. **3**(7), 34–57 (2002)
33. Müller, K., Smolic, A., Kautzner, M., Wiegand, T.: Rate-distortion optimization in dynamic mesh compression. In: ICIP, pp. 533–536. IEEE (2006). http://dblp.uni-trier.de/db/conf/icip/icip2006.html#MullerSKW06
34. Payan, F., Antonini, M.: Temporal wavelet-based geometry coder for 3D animated models. Comput. Graph. **31**(1), 77–88 (2007). https://doi.org/10.1016/j.cag.2006.09.009
35. Pearlman, W.A., Islam, A., Nagaraj, N., Said, A.: Efficient, low-complexity image coding with a set-partitioning embedded block coder. IEEE Trans. Circuits Syst. Video Technol. **14**(11), 1219–1235 (2004). https://doi.org/10.1109/TCSVT.2004.835150
36. Peercy, M.S., Airey, J., Cabral, B.: Efficient bump mapping hardware. In: Owen, G.S., Whitted, T., Mones-Hattal, B. (eds.) SIGGRAPH, pp. 303–306. ACM (1997). http://dblp.uni-trier.de/db/conf/siggraph/siggraph1997.html#PeercyA-C97
37. Pereira, F.C., Ebrahimi, T.: The MPEG-4 Book. Prentice Hall, Upper Saddle River (2002)
38. Praun, E., Hoppe, H.: Spherical parametrization and remeshing. ACM Trans. Graph. **22**(3), 340–349 (2003)
39. Quynh, D.T., He, Y., Chen, X., Xia, J., Sun, Q., Hoi, S.C.: Modeling 3d articulated motions with conformal geometry videos (CGVS). In: Proceedings of the 19th ACM International Conference on Multimedia, MM '11, pp. 383–392. ACM, New York, NY, USA (2011). https://doi.org/10.1145/2072298.2072349
40. Rau, C.: OpenGI—easy parameterization and geometry image creation. http://opengi.sourceforge.net/ (2011)
41. Ricard, J.: Video codec based point cloud compression (V-PCC) test model (2019). https://github.com/MPEGGroup/mpeg-pcc-tmc2
42. Sander, P.V., Gortler, S.J., Snyder, J., Hoppe, H.: Signal-specialized parametrization. In: Proceedings of the 13th Eurographics Workshop on Rendering, EGRW '02, pp. 87–98. Eurographics Association, Aire-la-Ville (2002). http://dl.acm.org/citation.cfm?id=581896.581909
43. Sander, P.V., Snyder, J., Gortler, S.J., Hoppe, H.: Texture mapping progressive meshes. In: Pocock, L. (ed.) SIGGRAPH, pp. 409–416. ACM (2001). http://dblp.uni-trier.de/db/conf/siggraph/siggraph2001.html#SanderS-GH01
44. Sattler, M., Sarlette, R., Klein, R.: Simple and efficient compression of animation sequences. In: Terzopoulos, D., Zordan, V.B., Anjyo, K., Faloutsos, P. (eds.) Symposium on Computer Animation, pp. 209–217. ACM (2005). http://dblp.uni-trier.de/db/conf/sca/sca2005.html#SattlerSK05
45. Schwarz, S., Preda, M., Baroncini, V., Budagavi, M., Cesar, P., Chou, P.A., Cohen, R.A., Krivokuća, M., Lasserre, S., Li, Z., Llach, J., Mammou, K., Mekuria, R., Nakagami, O., Siahaan, E., Tabatabai, A., Tourapis, A.M., Zakharchenko, V.: Emerging MPEG standards for point cloud compression. IEEE J. Emerging Sel. Top. Circuits Syst. **9**(1), 133–148 (2019). https://doi.org/10.1109/JETCAS.2018.2885981
46. Stefanoski, N., Klie, P., Liu, X., Ostermann, J.: Layered predictive coding of time-consistent dynamic 3d meshes using a non-linear predictor. In: ICIP (5), pp. 109–112. IEEE (2007). http://dblp.uni-trier.de/db/conf/icip/icip2007-5.html#StefanoskiKLO07
47. Sumner, R.W., Po-povic, J.: Mesh data from deformation transfer for triangle meshes (2004). http://people.csail.mit.edu/sumner/research/deftransfer-/data.html
48. Tang, X., Pearlman, W.A.: Three-Dimensional Wavelet-Based Compression of Hyperspectral Images, pp. 273–308. Springer, Boston (2006). https://doi.org/10.1007/0-387-28600-4_10
49. Taubin, G.: Curve and surface smoothing without shrinkage. In: Proceedings of IEEE International Conference on Computer Vision, pp. 852–857 (1995)
50. Vasa, L.: Geometric Laplacian dynamic mesh encoder (1.03, with kgcomparer) (2015). http://meshcompression.org/software-tools
51. Vlasic, D., Baran, I., Matusik, W.: Articulated mesh animation from multi-view silhouettes. http://people.csail.mit.edu/drdaniel/mesh_animation/in-dex.html (2008)
52. Vása, L., Marras, S., Hormann, K., Brunnett, G.: Compressing dynamic meshes with geometric Laplacians. Comput. Graph. Forum **33**(2), 145–154 (2014)
53. Wiegand, T., Sullivan, G.J., Bjontegaard, G., Luthra, A.: Overview of the H.264/AVC video coding standard. IEEE Trans. Circuits Syst. Video Technol. **13**(7), 560–576 (2003). https://doi.org/10.1109/TCSVT.2003.815165
54. Witten, I.H., Neal, R.M., Cleary, J.G.: Arithmetic coding for data compression. Commun. ACM **30**(6), 520–540 (1987). https://doi.org/10.1145/214762.214771
55. Xia, J., He, Y., Quynh, D.P., Chen, X., Hoi, S.C.: Modeling 3D facial expressions using geometry videos. In: Proceedings of the 18th ACM International Conference on Multimedia, MM '10, pp. 591–600. ACM, New York, NY, USA (2010). https://doi.org/10.1145/1873951.1874010
56. Xia, J., Quynh, D.T.P., He, Y., Chen, X., Hoi, S.C.H.: Modeling and compressing 3-d facial expressions using geometry videos. IEEE Trans. Circuits Syst. Video Technol. **22**(1), 77–90 (2012)
57. Xu, J., Joshi, R.L., Cohen, R.A.: Overview of the emerging HEVC screen content coding extension. IEEE Trans. Circuits Syst. Video Technol. **26**(1), 50–62 (2016). https://doi.org/10.1109/TCSVT.2015.2478706
58. Yoshizawa, S., Belyaev, A., Seidel, H.P.: A fast and simple stretch-minimizing mesh parameterization. In: Proceedings of the Shape Modeling International 2004, pp. 200–208. IEEE Computer Society, Washington, DC, USA (2004). https://doi.org/10.1109/SMI.2004.2

Canan Gulbak Bahce received the B.S. degree from Galatasaray University, Istanbul, Turkey, in 2005, and the M.S. degree from Université d'Artois, Arras, France in 2007, in software engineering. She is currently pursuing the Ph.D. degree at the Department of Computer Engineering, Istanbul Teknik University. Her current research interests include data compression and image, video, and computer graphics coding.

Ulug Bayazit received the B.S. degree from Bogazici University, Istanbul, Turkey, in 1991, and the M.S. and Ph.D. degrees from Rensselaer Polytechnic Institute, Troy, NY, in 1993 and 1996, respectively, all in electrical engineering. From 1996 to 2000, he was employed with the Toshiba America Consumer Products, Princeton, NJ and Toshiba America Electronic Components, San Jose, CA, as Research Scientist and Systems Architect, respectively. In 2000, he was appointed as an Assistant Professor at the Electrical and Electronics Engineering Department in Isik University, Istanbul Turkey where he became an Associate Professor in 2004. Between 2007 and 2013, he was employed as an Associate Professor with the Computer Engineering Department of Istanbul Technical University. Since 2013, he is a Professor of the same department. His research interests include coding and processing of visual data, computational geometry and machine learning.