Scalable Wavelet-based Coding of Irregular Meshes with Interactive Region-of-Interest Support

Jonas El Sayeh Khalil, Adrian Munteanu, *Member, IEEE,* and Peter Lambert, *Member, IEEE*

***Abstract*—This paper proposes a novel functionality in wavelet-based irregular mesh coding which is interactive region-of-interest (ROI) support. The proposed approach enables the user to define arbitrary ROIs at the decoder side and to prioritize and decode these regions at arbitrarily high granularity levels. In this context, a novel adaptive wavelet transform for irregular meshes is proposed which enables (i) varying the resolution across the surface *at arbitrarily fine granularity levels*, and (ii) dynamic tiling, which *adapts the tile sizes* to the local sampling densities at each resolution level. The proposed tiling approach enables a rate-distortion-optimal distribution of rate across spatial regions.**

When limiting the highest-resolution ROI to the visible regions, the fine granularity of the proposed adaptive wavelet transform reduces the required amount of graphics memory by up to 50%. Furthermore, the required graphics memory for an arbitrary small ROI becomes negligible compared to rendering without ROI support, independent of any tiling decisions. Random access is provided by a novel dynamic tiling approach, which proves to be particularly beneficial for large models of over $10^6 \sim 10^7$ vertices. The experiments show that dynamic tiling introduces a limited lossless rate penalty compared to an equivalent codec without ROI support. Additionally, rate savings up to 85% are observed while decoding ROIs of tens of thousands of vertices.

***Index Terms*—Irregular mesh coding, Region-of-Interest coding, wavelet coding, random access**

I. Introduction

3D acquisition devices generate increasingly larger and geometrically accurate models, delivering up to hundreds of millions of vertices with every scan. The main issues that have to be tackled when processing such models include storage capacity and memory requirements, bandwidth limitations, latency, and processing power. This resulted in advancements in single-rate and progressive compression techniques to handle storage and transmission problems, and out-of-core processing approaches and random accessibility to address memory and computational constraints.

These approaches are complementary. On the one hand, compression techniques allow for a small storage footprint and low transmission bandwidth, at the expense of larger processing power requirements for actually decoding the required mesh. Compression techniques should however offer resolution and quality scalability to cope with increasing sampling densities and acquisition accuracy offered by modern scanning systems, as 3D assets become too large to fit within hardware memory limits. Resolution-scalable coding solutions elegantly provide a scalable representation of high-resolution meshes. Starting from a base mesh, the resolution can progressively increase until the desired level is reached. On the other hand, out-of-core processing copes with memory and computational limitations by only loading the required parts of a 3D model in memory, though it does not reduce the total storage size. This form of scalability is conventionally termed *region-of-interest* (ROI) coding, and is well-suited for rendering detailed local views of large data sets.

A combination of both resolution scalability and ROI coding is required for a truly scalable mesh coding system, making optimal use of the available bandwidth and memory when given a specific camera viewpoint, or in other words, given requested *regions* and their appropriate *resolutions*. This combination is the main focus of this paper, and is achieved by making use of wavelet-based coding.

Wavelet representations have been used in several multimedia processing domains as an efficient way to represent data in a multi-resolution fashion. In this work we build upon the very nature of wavelets to construct multi-resolution mesh representations, hereby tackling challenges for ROI coding, as wavelet transforms are inherently global transforms.

The remainder of this paper is structured as follows. In Section II we will discuss related work. An overview of wavelet-based mesh coding is given in Section III. Section IV reviews our work on encoder-side ROI coding. In Section V we discuss the proposed adaptive inverse wavelet transform, followed by our dynamic tiling approach presented in Section VI. Details on how this approach can be exploited to improve the rate-distortion performance are given in Section VII. The experimental evaluation is presented in Section VIII, followed by our conclusions drawn in Section IX.

II. Related Work

Figure 1 depicts the architecture of a conventional mesh coding system. After an initial transform of mesh M to M^{TF}, entropy can be reduced in an encoding step resulting in a compressed representation M_{enc} of the input mesh. Scalable systems, which allow for progressively refining a mesh, are divided into fine-grained vertex-by-vertex schemes which add a single vertex per refinement step, and coarser-grained multi-resolution schemes which refine a mesh over its entire surface. Refinement operations can be localized in specific ROIs, which can be defined at either the encoder or at the decoder side. This corresponds to encoder-side and decoder-side ROI coding functionalities respectively.

J. El Sayeh Khalil and P. Lambert are with the Electronics and Information Systems Department (ELIS), Ghent University-imec, Belgium e-mail: jonas.elsayehkhalil@ugent.be.

A. Munteanu is with the Department of Electronics and Informatics, Vrije Universiteit Brussel, Belgium.

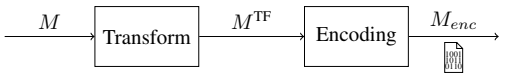


Fig. 1. Basic architecture of any mesh encoding system. A mesh M is transformed into M^{TF} with reduced entropy. Compression is achieved through the encoding block, which makes use of residual or subband encoding and entropy coding to obtain a compressed representation M_{enc} of the input mesh.

The encoding of an *encoder-side ROI* is symmetric: the decoded data is equivalent to the encoded data. Encoder-side ROIs are useful for prioritizing regions within M for coping with reduced bandwidths; e.g., prioritizing facial details of a virtual human character over details in his clothes. This codec functionality is further discussed in Section II-A.

Conversely, a *decoder-side* ROI adapts the resolution over the mesh surface to the needs of the decoder, requiring an encoding procedure which can no longer be symmetric. A decoder must be able to request specific parts of a model at specific resolutions, affecting either M_{enc}, M^{TF} or both. This type of functionality is further addressed in Section II-B.

A. Encoder-side Regions-of-Interest

The premise for encoder-side ROIs is easy to formulate: store data such that the prioritized regions are transmitted and decoded before the unprioritized background (BG) regions.

Vertex-by-vertex based systems such as [1] or [2] can trivially prioritize regions: vertices are prioritized individually, directly altering the refinement and encoding order.

Multi-resolution systems such as [3]–[7], where individual vertices are no longer accessible after the transform, call for specific designs. Few solutions have been proposed for encoder-side ROIs for multi-resolution systems. A solution which was inspired by the JPEG2000 maxshift and general scaling operations [8] was proposed for semi-regular codecs by Zheng et al. [9]. For wavelet-based *irregular* mesh codecs, we have recently proposed an encoder-side ROI coding approach in [10]. This method is similarly inspired by the maxshift operation and introduces ROI-steered upsampling for irregular wavelet transforms, which exploits the irregularity of meshes to reduce memory requirements. We briefly review our earlier encoder-side ROI coding method [10] in Section IV.

B. Decoder-side Regions-of-Interest

Contrary to encoder-side ROIs which are based on *predefined* prioritizations of specific spatial regions in the mesh, a decoder-side ROI is more involved as a single bit stream needs to provide for a dynamic, arbitrary ROI selection *at decoding time* in an efficient way.

A first 3D graphics domain where arbitrary ROI decoding was required was terrain visualization. Gobbetti et al. [11] proposed adaptive meshes for terrain data. The employed grid-based structure of [11] allowed for easy random access support as each two adjacent triangles form a root for further subdivision and can be processed individually. Similar tiled approaches have been suggested e.g. in [12], [13], and have proven to be successful for $2.5D$ surfaces, i.e. surfaces where each (x, y) couple corresponds to, at most, a single point (x, y, z) on the surface. However, such approaches are too restrictive for the general case of $3D$ surfaces.

In the following, we shortly overview the literature by focusing on *transform-based* ROI coding in Section II-B1 and *tile-based* ROI coding in Section II-B2. We list our contributions in Section II-C.

1) Transform-based ROI Coding: For *vertex-by-vertex* representations, decoder-side ROIs (e.g., a view-dependent selection) can be obtained by either carefully applying or skipping refinements. Such approaches are often termed as *selective refinement* methods. Hoppe [14] proposed such an ROI decoding approach for his progressive meshes method [1], which was implemented on graphics hardware by Hu et al. [15]. For *multi-resolution* representations, where each refinement step adds multiple vertices, the refinements need to be restricted based on the required ROIs. Gioia et al. [16] have proposed ROI support for wavelet representations of *semi-regular meshes*, whereby the resolution can be adapted over the geometric surfaces by suppressing wavelet coefficients. This corresponds to what we prove to be an adaptive inverse wavelet transform for *semi-regular meshes*. Roy et al. [17] have proposed a multi-resolution analysis for meshes with surface attributes, using the edge collapse operations of [1] and its selective refinement [14]. Similar to Gioia et al., the detail coefficients are also selectively filtered [17].

These solutions are termed *transform-based* ROI coding methods as the transform is enhanced to offer ROI scalability. This allows for adapting the resolution in a fine-grained manner over the surface. However, these solutions suffer from the fact that only the refinement steps are ROI-aware without providing for random access within the data stream; that is, the ROIs are only defined on the transformed mesh M^{TF} in Figure 1, requiring full decoding of M_{enc}.

2) Tile-based ROI Coding: The idea of tiling and encoding a mesh was proposed by [18] to allow for out-of-core processing. However, this approach did not offer random access due to the dependencies between encoded tiles. Most works decide a global tiling, after which each tile is individually encoded. These approaches are termed as *tile-based*, whereby ROI scalability is offered by the tiling process itself.

Initial tile-based approaches used a *single-rate codec operating on a global tiling* of a mesh. Choe et al. [19] proposed a tiling which resulted in a wire-net mesh representing the tile borders, and encoding each tile independently using a single-rate encoder. In [20], the authors improved upon [19], allowing for more control over random accessibility, better compression, and support for large meshes which require out-of-core processing. Yoon and Lindstrom [21] also group triangles in tiles, after which each of these tiles is then compressed using their streaming mesh compression approach [22]. In this method, each tile is also encoded at a single rate. Such approaches make binary decisions on tile granularity: either the tile is (partly) visible and is entirely decoded, or it is not visible and the decoding is skipped. The main downside is that this does not allow for scaling the resolution or quality of the visible regions, with issues similar to those of single-rate coding without tiling.

By employing a *progressive encoding per global tile*, each

tile can be decoded at a different level of detail, for instance depending on the distance to a virtual camera, or depending on some specified region of interest. Liu and Zhang describe a wavelet-based mesh compression scheme with random access [23], as one of the first works which combine random access with scalable coding, as opposed to utilizing a single-rate compression per tile. Making use of [4], their work operates on semi-regular meshes to obtain progressive encoding per tile, while random access is offered by exploiting the employed zerotree-based encoding and considering the base triangles as tiles. A binary decision is still made based on back-face culling: if an oriented triangle of the base mesh has its front side facing the virtual camera, the corresponding trees of coefficients are transmitted. This avoids tiling artefacts but results in a high sampling density, which is common for semi-regular mesh representations. Roudet et al. [24] similarly use [4], and define tiles by projecting high-resolution data onto the base triangles. In [25], Cheng et al. improve the progressive coder of [26] to use multi-granular quantization, and use this coder to encode tiles obtained after chartification. No details are given concerning approaches to avoid artefacts at the tile borders. Maglo et al. [27] continued on this approach, using the progressive codec of [28] combined with post-processing of the tile borders to ensure a valid topology. This is done by moving the border vertices of the higher-resolution tile towards the border vertices of the lower-resolution tile, and triangulating any remaining holes. This process results in visual blocking artefacts. This method was further continued in [29], allowing for smooth transitions between adjacent tiles without post-processing, with the restriction that neighbouring regions can differ by at most one resolution level. Du et al. [30] propose a coding method based on the codec of Gandoin and Devillers [31], which lowers the quantization of vertex locations iteratively. A two-level tree is proposed where the root represents the coarsely-quantized base mesh to be decoded completely, while the subtrees, i.e., the tiles, can be individually decoded each to its desired level. Again, care has to be taken for border triangles lying across subtrees. Finer granularity and random access for vertex-based selective-refinement coding was proposed by Kim et al. in [32] by partitioning the original vertex hierarchy into sub-blocks acting as tiles for random accessibility.

The downside of these *tile-based* solutions is that they cannot adapt the resolution in a fine-granular way as offered by *transform-based* approaches; the granularity is limited by the tile granularity. Consequently, care has to be taken near tile borders to avoid artefacts, which can be challenging if the resolutions of neighbouring tiles differ by multiple levels.

Dynamic tiling: A solution where the tiles used for random access change dynamically is given by Courbet and Hudelot [33]. The authors propose a recursive mesh splitting approach to obtain hierarchical random access for polygonal meshes. The resulting representation allows to randomly access arbitrarily small portions of a mesh, but does not support resolution scalability: the portions of a mesh within the ROI have to be iteratively subdivided until the original triangle mesh is obtained for the ROI. The subsequently smaller tiles of [33] allow for more efficient processing.

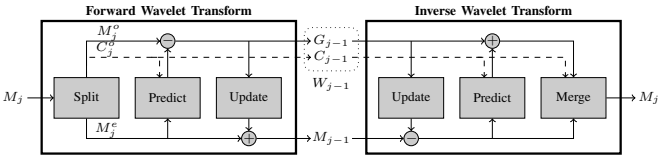


Fig. 2. The conventional wavelet transform. During the analysis step, a mesh M_j is decomposed into a lower-resolution mesh M_{j-1} and wavelet coefficients W_{j-1}. To allow for irregular mesh coding, additional connectivity information has to be provided, denoted as C_{j-1}.

C. Contributions

This work is the first to provide decoder-side ROI support for wavelet-based *irregular* mesh codecs. We bring the following contributions:

- *An adaptive inverse wavelet transform for irregular meshes.* The wavelet coefficients are filtered *and* the appropriate connectivity information is provided before performing the inverse wavelet transform. This offers transform-based ROI support and allows for reduced graphics memory requirements compared to wavelet transforms for semi-regular meshes.
- *Dynamic tiling per resolution in the wavelet domain.* By tiling in the wavelet domain, random access is provided for our transform-based ROI while avoiding tiling artefacts in the spatial domain. The tiling adapts to the sampling densities within each resolution, allowing for an optimal trade-off between coding performance and random access granularity *per resolution*.
- *Tile-based rate-distortion optimization.* An independent encoding of tiles allows for reordering tiles within a resolution level and even across resolutions, while preserving the benefits offered by our quality scalability. A rate-distortion-optimized bitplane transmission order is then obtained such that those bitplanes of the tiles which give the largest quality gain at the lowest rate are coded first.

III. Wavelet-based Mesh Coding and ROIs

The conventional wavelet transform used in mesh coding is given in Figure 2. This scheme is generic and stems from the classical synthesis of wavelet transforms based on lifting [34]. For semi-regular meshes, connectivity information is implicit, and needs not be transmitted. To accommodate for irregular meshes, the dashed lines in Figure 2 have been added to indicate that connectivity information can no longer be assumed to be implicitly known for this type of meshes.

For irregular meshes, each wavelet subband W_{j-1} (see Figure 2) contains the geometry information represented by the *conventional* wavelet coefficients $w_g \in G_{j-1}$ obtained via lifting, together with explicit connectivity information $w_c \in C_{j-1}$ to ensure the correct connectivity. For encoding, we employed a so-called template mesh, proposed originally in our previous works [7], [35], which was introduced in order to decouple the transform step from the encoding step and to allow for quality scalability. For each subband $j-1$, a template mesh M_{j-1}^T is maintained, representing all connectivity information in the original mesh. As both M_{j-1} and M_{j-1}^T share the same

connectivity, there exists a bijective mapping μ linking the vertices of both, that is:

$$\forall v \in M_{j-1} : \mu(v) \in M_{j-1}^T \quad (1)$$

$$\forall v^T \in M_{j-1}^T : \mu^{-1}(v^T) \in M_{j-1} \quad (2)$$

We will denote the reconstruction of the high-resolution mesh M_j given the low-resolution mesh M_{j-1} and wavelet subband W_{j-1} as:

$$M_j = WT^{-1}(M_{j-1}, W_{j-1}) \quad (3)$$

where WT^{-1} denotes the inverse wavelet transform. Let M_j^e and M_j^o represent the even and odd sets of vertices respectively in M_j as obtained by the splitting step in Figure 2. After downsampling, retriangulating and updating, the vertices $v_j^{e_i} \in M_j^e$ form the vertices $\gamma(v_j^{e_i}) = v_{j-1}^i$ of M_{j-1}, where γ is the bijective operator that maps $v_j \in M_j^e$ to $v_{j-1} \in M_{j-1}$. We define the wavelet coefficients corresponding to the odd vertices $v_j^{o_i} \in M_j^o$ as $\omega(v_j^{o_i}) = w_{j-1}^i \in W_{j-1}$, where ω is the bijective operator that maps $v_j \in M_j^o$ to $w_{j-1} \in W_{j-1}$.

Denote by w_c and w_g the connectivity and geometry information of wavelet coefficient w_{j-1}^i respectively. To simplify notations, the index i and subband $j-1$ are not explicitly given for these two variables. In the inverse wavelet transform, each $v_j^{o_i} = \omega^{-1}(w_{j-1}^i)$ is reconstructed as follows:

$$v_j^{o_i} = \Big(\sum_{k \in |M_{j-1}|} w_c^k v_{j-1}^k \Big) + w_g \quad (4)$$

where $|\ \cdot\ |$ is the cardinality of a set. In this equation, the first term represents the '*Predict*' step in the inverse wavelet transform in Figure 2, weighing all vertices v_{j-1}^k of the lower-resolution mesh by weights given by the connectivity information in w_{j-1}^i. The second term corrects this prediction. We now define the support $S(w_{j-1}^i)$ for a wavelet coefficient w_{j-1}^i as the set of vertices with a non-zero weight in (4):

$$S(w_{j-1}^i) = \{v_{j-1}^k \in M_{j-1} : w_c^k \neq 0\} \quad (5)$$

That is, $S(w_{j-1}^i)$ corresponds to the set of vertices $v_{j-1}^k \in M_{j-1}$ required for reconstructing $v_j^{o_i} = \omega^{-1}(w_{j-1}^i)$.

Given meshes M_j, an (encoder-side or decoder-side) application can now define subsets of vertices as *spatial* regions of interest $ROI_j^S \subset M_j$. We translate these ROI_j^S to regions of interest in the wavelet domain as follows:

$$ROI_j^W = \{w \in W_j : S(w) \cap ROI_j^S \neq \varnothing\} \quad (6)$$

i.e., ROI_j^W is the subset of those wavelet coefficients in W_j whose support overlaps with the given ROI_j^S. Additionally, we define the support of the set ROI_j^W as:

$$S(ROI_j^W) = \bigcup_{w \in ROI_j^W} S(w) \quad (7)$$

To accommodate for the varying resolution across a surface, we introduce the notations $M_{\alpha \leq j}, M_{\beta \leq j}, \ldots$ to identify several *partial* reconstructions of M_j, i.e., the resolution varies over the surface, reaching at most resolution j. Given such a partially reconstructed mesh $M_{\alpha \leq j}$ and ROI_j^W, we generalize (3) as follows:

$$M_{\alpha \leq j+1} = WT^{-1}(M_{\alpha \leq j}, ROI_j^W) \quad \text{with } S(ROI_j^W) \subset M_{\alpha \leq j} \quad (8)$$

The requirement in (8) states that if the support for each wavelet coefficient $w \in ROI_j^W$ is present in the *partially* reconstructed $M_{\alpha \leq j}$, then the inverse wavelet transform results in an upsampled mesh $M_{\alpha \leq j+1}$. Otherwise, if there exists at least one wavelet coefficient with a support which is not entirely present in $M_{\alpha \leq j}$, then the inverse transform is topologically ill-defined as the prediction term in (4) cannot be evaluated properly.

The codec design followed in this work follows the generic architecture of Figure 1. A number of key ingredients offer the required functionalities of quality and resolution scalability and region of interest coding. These include:

- successive approximation quantization (SAQ) of the wavelet coefficients [36] and bitplane coding, performing scalable quantization and coding of the wavelet coefficients and enabling quality and resolution scalability [7], [35] over entire models, without providing ROI support;
- wavelet coefficient boosting [10], discussed in Section IV, enabling encoder-side ROI support;
- the adaptive wavelet transform proposed in Section V, which is a key component to enable decoder-side ROI coding support;
- dynamic tile-based coding proposed in Section VI, enabling interactive ROI-support; and
- rate-distortion optimization, detailed in Section VII, offering optimized allocation of rate across wavelet subbands and bitplanes.

IV. Irregular Mesh Coding and Encoder-side ROIs

In [10] we proposed an encoder-side ROI coding method for wavelet-based irregular mesh codecs which allows for prioritizing specific geometric regions at the encoder side by defining an ROI_j^S at each resolution j. These regions can be encoded and transmitted without altering the coding step itself by taking advantage of the employed successive approximation quantization of the wavelet coefficients [10]. By scaling up the wavelet coefficients in each ROI_j^W, indicated as *wavelet boosting*, the ROI information is transmitted prior to the BG information. For the sake of completeness, our encoder-side ROI coding method [10] is shortly reviewed below. For a complete overview we refer the reader to [10].

ROI propagation: To ensure that ROI_j^S can be reconstructed after upsampling $ROI_k^S\ \forall k < j$, the propagation operator σ is introduced, propagating ROI_j^S at resolution j to the region $\sigma(ROI_j^S)$ at resolution $j-1$: $\sigma(ROI_j^S) =$

$$\{v \in M_{j-1}, \exists v_j^e \in (ROI_j^S \cap M_j^e) : v = \gamma(v_j^e)\} \quad \cup \quad \{v \in M_{j-1}, \exists v_j^o \in (ROI_j^S \cap M_j^o) : v \in S(\omega(v_j^o))\} \quad (9)$$

That is, all even vertices in ROI_j^S are preserved in $\sigma(ROI_j^S)$ while odd vertices require the support of the corresponding wavelet coefficients to be present as well. Requiring that $\sigma(ROI_{k+1}^S) \subseteq ROI_k^S\ \forall k < j$ ensures that any reconstruction

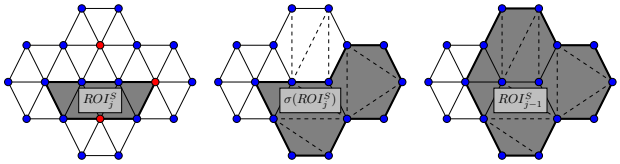


Fig. 3. ROI propagation. This figure shows M_j and ROI^j before downsampling, and M_{j-1} with $\sigma(ROI_j^S)$ and a possible ROI_{j-1}^S.

$M_{\alpha\leq j}$ can be obtained with ROI_j^S properly reconstructed. $ROI_j^S, \sigma(ROI_j^S)$ and ROI_{j-1}^S are illustrated in Figure 3.

Boosted wavelet coefficients: Encoder-side ROI support is offered by using boosted wavelet coefficients: the quantized wavelet coefficients $w \in ROI_j^W$ are premultiplied by a scaling value $s_j > |w|, \forall w \in W_j \setminus ROI_j^W$. We construct the set of boosted wavelet coefficients W_j^B as:

$$\begin{cases} \forall w \in ROI_j^W : s_j w \in W_j^B \\ \forall w \in W_j \setminus ROI_j^W : w \in W_j^B \end{cases} \tag{10}$$

The original wavelet coefficients can be obtained as:

$$\begin{cases} \forall w \in W_j^B \wedge w \geq s_j : w/s_j \in W_j \\ \forall w \in W_j^B \wedge w < s_j : w \in W_j \end{cases} \tag{11}$$

Bitplane coding ensures that the wavelet coefficients in the ROI are encoded before those in the background, while quality scalability allows for storing the ROIs of *all* resolutions before any BG information.

ROI-steered upsampling: Limiting the decoder to the coding layers associated with the ROI results in deterioration in the BG, because the connectivity information is not ROI-aware. The set $\widetilde{M}_j^o$ of reconstructed odd vertices creates *all* vertices when upsampling:

$$\widetilde{M}_j^o = \{\omega^{-1}(w) : w \in W_{j-1}\} \tag{12}$$

Only the vertices in the ROI are accurately positioned, while prediction errors in BG-regions accumulate per resolution. Instead of low-pass filtering the BG vertices to ensure a smooth BG surface, irregular meshes allow for limiting the upsampling to only create vertices in the decoded ROI:

$$\widetilde{M}_j^o = \{\omega^{-1}(w) : w \in ROI_{j-1}^W \subset W_{j-1}\} \tag{13}$$

ROI_{j-1}^W is detected as the set of all *non-zero* wavelet coefficients when only decoding the ROIs; detecting zero-magnitude coefficients can be done by adding a pre-increment and post-decrement to (10) and (11) respectively.

V. Adaptive Inverse Wavelet Transform

The ROI-steered upsampling proposed in our prior work [10] (and discussed in Section IV) is suitable for *encoder-side* ROI approaches for which ROIs are predefined at the encoding side: per resolution j, a spatial-domain ROI_j^S will be accurately reconstructed in $M_{\alpha\leq j}$. In contrast, *decoder-side* ROI approaches, such as proposed in the current paper, have to accommodate *interactive* ROIs, which are arbitrarily defined while resolutions are being reconstructed. That is, at each resolution j, an ROI_j^S is specified by the user at the decoder side. During upsampling, these arbitrary ROIs cannot take into account (unknown) higher-resolution ROIs, as was done in our encoder-side ROI approach [10].

The proper reconstruction of $M_{\alpha\leq j+1}$ given ROI_j^S possibly requires modifying lower-resolution meshes $M_{\alpha\leq k}, k \leq j$ to obtain additional samples to satisfy the condition in (8). These lower-resolution $M_{\alpha\leq k}$ must be reconstructed with a larger ROI denoted by $ROI_{k|j}^S$, which is the *expansion* of ROI_k^S to yield a proper reconstruction at the higher-resolution j.

Consider for instance that $M_{\alpha\leq j}$ is accurately reconstructed given all lower-resolution regions of interest $ROI_k^S, k < j$. To reconstruct $M_{\alpha\leq j+1}$ given ROI_j^S, $M_{\alpha\leq j}$ needs to be modified to $M_{\beta\leq j}$, obtaining the additional samples to ensure $S(ROI_j^W) \subset M_{\beta\leq j}$. First, the given ROI_j^S needs to be expanded to $ROI_{j|j}^S$ in order to encompass the support $S(ROI_j^W)$ defined in (7), i.e., to include the supports of the wavelet coefficients $w \in W_j$ which overlap with ROI_j^S:

$$ROI_{j|j}^S = ROI_j^S \cup S(ROI_j^W) \tag{14}$$

To reconstruct $M_{\beta\leq j}$, the lower-resolution mesh $M_{\alpha\leq j-1}$ needs to be upsampled considering the interactively-specified ROI at resolution $j-1$, ROI_{j-1}^S, expanded to $ROI_{j-1|j}^S$, in order to ensure its accurate reconstruction at resolution j. One can write:

$$ROI_{j-1|j}^S = ROI_{j-1|j-1}^S \cup \sigma(ROI_{j|j}^S) \tag{15}$$

The new ROI encompasses both $ROI_{j-1|j-1}^S$, i.e., the expansion of ROI_{j-1} which yields a proper reconstruction at resolution $j-1$, and the propagation of $ROI_{j|j}^S$ to resolution $j-1$, as defined in (9).

Recursively, to reconstruct $M_{\beta\leq j}$ given this further expanded $ROI_{j-1|j}^S$, $M_{\alpha\leq j-1}$ needs to be modified to $M_{\beta\leq j-1}$, reconstructed from the subsequent lower-resolution mesh $M_{\alpha\leq j-2}$ by considering $ROI_{j-2|j}^S$, with:

$$ROI_{j-2|j}^S = ROI_{j-2|j-1}^S \cup \sigma(ROI_{j-1|j}^S) \tag{16}$$

(15) and (16) can be generalized, defining the expansions for $k < j$ recursively as:

$$ROI_{k|j}^S = ROI_{k|j-1}^S \cup \sigma(ROI_{k+1|j}^S) \tag{17}$$

The expansion $ROI_{k|j}^S$ of ROI_k^S given the *newly determined* ROI_j^S is the union of its expansion $ROI_{k|j-1}^S$ complying for the *already known* ROI_{j-1}^S, and the propagation $\sigma(ROI_{k+1|j}^S)$ of the higher-resolution expansion $ROI_{k+1|j}^S$.

Given the spatial-domain ROI_j^S with wavelet-domain ROI_j^W obtained via (6), and $S(ROI_j^W) \not\subset M_{\alpha\leq j}$, the adaptive inverse wavelet transform can now be recursively defined to obtain $M_{\beta\leq j}$:

$$M_{\beta\leq 1} = WT^{-1}(M_0, ROI_{0|j}^W) \tag{18}$$

$$M_{\beta\leq k} = WT^{-1}(M_{\beta\leq k-1}, ROI_{k-1|j}^W) \tag{19}$$

With this, $WT^{-1}(M_{\beta\leq j}, ROI_j^W)$ is properly defined.

The adaptive inverse wavelet transform is illustrated via the example of Figure 4. The figure shows a mesh M on the top line, and its template mesh M^T on the bottom line. The ROIs in Figure 4 are shaded as indicated in the figure. The

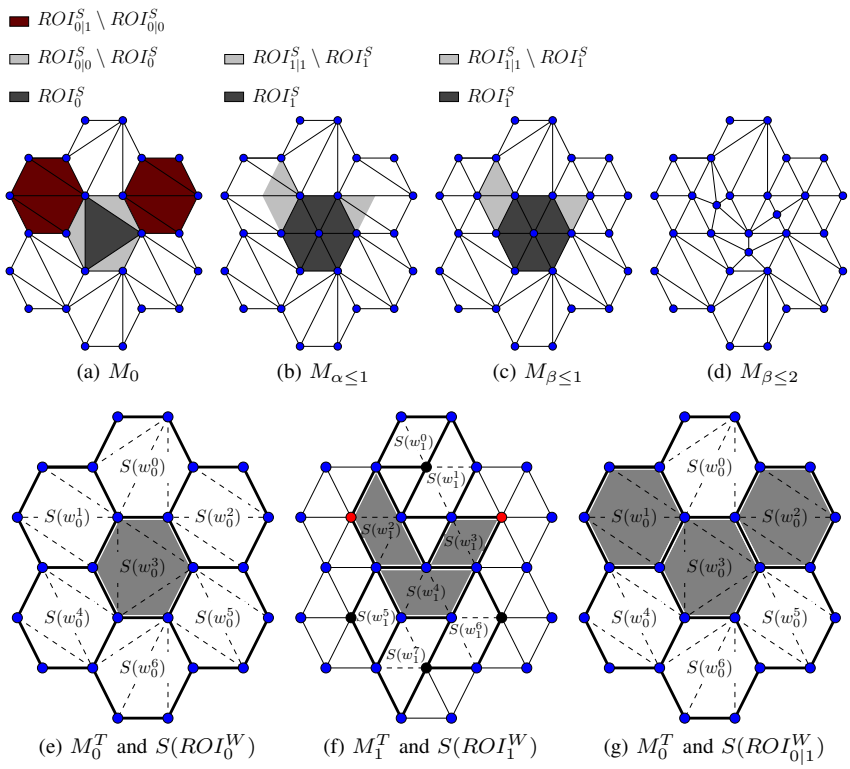


Fig. 4. Adaptive inverse wavelet transform. The figure shows the decoded mesh, with determined ROIs indicated in dark gray and expanded ROIs in light gray. The inverse transform operations subsequently transform M_0 (Fig. 4a) to $M_{\alpha\leq 1}$ (Fig. 4b) at resolution 1, a second version $M_{\beta\leq 1}$ (Fig. 4c) and finally $M_{\beta\leq 2}$ (Fig. 4d). The requested ROI^S are shaded in dark gray, and the corresponding $S(ROI^W)$ are depicted on the bottom line.

Listing 1 Example: Adaptive inverse wavelet transform

function UPSAMPLE(ROI_0^S)
  given M_0 and ROI_0^S ▷ (Fig. 4a)
  $ROI_0^W = \{w_0^3\}$
  $S(ROI_0^W) = S(w_0^3) \subset M_0$ ▷ (Fig. 4e)
  return $M_{\alpha\leq 1} = WT^{-1}(M_0, ROI_0^W)$ ▷ (Fig. 4b)
end function

function UPSAMPLE(ROI_1^S)
  given ROI_1^S ▷ (Fig. 4b)
  $ROI_1^W = \{w_1^2, w_1^3, w_1^4\}$
  $S(ROI_1^W) = S(w_1^2) \cup S(w_1^3) \cup S(w_1^4)$ ▷ (Fig. 4f)
  $\implies S(ROI_1^W) \not\subset M_{\alpha\leq 1}$ ▷ (Fig. 4b)
  function UPSAMPLE($ROI_{0|1}^S$)
    $ROI_{0|1}^S = ROI_{0|0}^S \cup \sigma(ROI_{1|1}^S)$ ▷ (Fig. 4a)
    $ROI_{0|1}^W = \{w_0^1, w_0^2, w_0^3\}$
    $S(ROI_{0|1}^W) = S(w_0^1) \cup S(w_0^2) \cup S(w_0^3)$ ▷ (Fig. 4g)
    return $M_{\beta\leq 1} = WT^{-1}(M_0, ROI_{0|1}^W)$ ▷ (Fig. 4c)
  end function
  return $M_{\beta\leq 2} = WT^{-1}(M_{\beta\leq 1}, ROI_1^W)$ ▷ (Fig. 4d)
end function

performed steps are described in Listing 1. Observe that the mapping μ is no longer surjective if $M_{\alpha\leq j} \neq M_j$:

$$v^T \in M_j^T \not\Rightarrow \exists v \in M_{\alpha\leq j} : \mu(v) = v^T \tag{20}$$

Although not *all* $v^T \in M_j^T$ have a corresponding $v \in M_{\alpha\leq j}$, the recursive expansion and propagation of ROIs (17) does ensure such a correspondence for the *required* vertices.

The main advantage of such an adaptive inverse wavelet transform is that a rendering system is allowed to select its desired ROI, reducing the processing power spent for the inverse wavelet transform, and reducing the data transmission to, and the memory usage on, the graphics hardware. The main disadvantage is that this approach still requires decoding all data in M_{enc}. Additionally, one notes that the size of the transformed data M^{TF}, i.e., before any arithmetic or entropy coding, is often proportional to the size of M itself. Hence, while graphics memory requirements are optimized, the memory required to obtain this ROI-adapted model is not.

VI. Dynamic Tile-based Coding

To reduce memory and bandwidth requirements when reconstructing $M_{\alpha\leq j+1}$ given ROI_j^S, only the wavelet coefficients $w_j \in ROI_j^W$ and $w_k \in ROI_{k|j}^W$ for $k < j$ need to be decoded. This reveals the conventional trade-off in randomly-accessible coding: at one end of the spectrum, all samples are individually decodable which means there is *no entropy coding* but perfect random accessibility; at the other end of the spectrum, all samples are encoded simultaneously resulting in optimal coding performance but *no random accessibility*. To solve this trade-off, the proposed coding paradigm makes use of dynamic tile-based coding, detailed in this section.

We employed the connectivity and geometry coder of [7]. For each wavelet coefficient $w \in W_j$, the connectivity infor-

mation $w_c \in C_j$ is represented by assigning a binary value $\beta(e_j^k)$ to each edge e_j^k, indicating whether or not the edge is preserved when upsampling. This is shown in Figure 4e for M_0^T using full lines for edges that are preserved, and dashed lines across which triangles are merged. Merging triangles across such edges results in non-triangular faces which are immediately recognized as *patches*. An odd vertex $v_{j+1}^{o_i}$ is added per patch, and the patch is retriangulated (depicted in Figure 4f). The geometry information $w_g \in G_j$ allows for an accurate reconstruction of the vertex positions.

The connectivity samples, i.e. a sample per edge for C_j, and the geometry samples, i.e. a single sample per wavelet coefficient for G_j, are encoded using a connectivity coder and a geometry coder respectively. Both make use of octree coding, by embedding the samples through the template mesh. Let $v_j^{k_a}$ and $v_j^{k_b}$ denote the two vertices which define e_j^k. Given the embedding operator ϵ which embeds a vertex v at position $\epsilon(v) = \check{\mathbf{v}} \in \mathbb{R}^3$, the binary values $\beta(e_j^k)$ are embedded at

$$\epsilon(e_j^k) = \check{\mathbf{e}}_j^k = \frac{1}{2}\epsilon\Big(\mu(v_j^{k_a})\Big) + \frac{1}{2}\epsilon\Big(\mu(v_j^{k_b})\Big) \tag{21}$$

and the wavelet coefficients $\omega(v_{j+1}^{o_i})$ are embedded at

$$\epsilon(v_{j+1}^{o_i}) = \check{\mathbf{v}}_{j+1}^{o_i} = \epsilon\Big(\mu(v_{j+1}^{o_i})\Big) \tag{22}$$

In [7], [35], we used the template mesh for embedding both the connectivity and geometry samples in $\mathbb{R}^3$ as described above. In Section V, the same template mesh is additionally used to map the wavelet coefficient supports $S(w)$ from M_j^T where they are ensured to be accurately represented, to $M_{\alpha \le j}$ where these supports are not necessarily fully reconstructed. The next sections discuss approaches to avoid decoding an *entire* subband before performing an adaptive inverse transform. Section VI-A discusses tiling of the geometry samples $\check{\mathbf{v}}_{j+1}^{o_i}$; Section VI-B goes on by discussing how the connectivity samples $\check{\mathbf{e}}_j^k$ can be tiled.

A. Tiled Geometry Information

For an ROI-based reconstruction of $M_{\alpha \le j+1}$, the decoding of the geometric samples can be limited to $w \in ROI_{k|j}^W$ for each $k \le j$. A partial, ROI-based decoding of these geometric samples does not hinder subsequent decoding steps as each M_j^T, which embeds the samples, is reconstructed using only connectivity information.

At each resolution, C_j is fully decoded. To allow for random access into the geometry data G_j, the geometry samples can be partitioned into tiles based on any criterion that can be mirrored by a decoder; for instance, based on the topology of M_{j+1}^T or based on the sampling locations $\check{\mathbf{v}}_{j+1}^{o_i}$ within M_{j+1}^T.

Each odd vertex corresponds to a single geometry sample, and can consequently be mapped to a single tile after partitioning. Denote the mapping of odd vertices v_j^o to tile $T_j^{\mathbf{x}}$ as $T(v_j^o) = T_j^{\mathbf{x}}$. The set of tiles required for ROI_j^W is:

$$\mathcal{T}_j^{\text{req}} = \bigcup_{w \in ROI_j^W} T\big(\omega^{-1}(w)\big) \tag{23}$$

Except for signalling the tiles within a bit stream, the only lossless rate penalty is caused by the trade-off between fine-granular random access (requiring smaller tiles) and high coding performance (requiring larger tiles). As the portion of geometry information vastly surpasses the connectivity information (see, for instance, [2], [5], [37]), large speed-ups and rate savings for ROI decoding can be obtained. However, using a template mesh M^T having the same number of vertices as the original mesh has memory limitations: for instance, decoding only a fraction of a multi-million vertex model M still requires a multi-million vertex M^T in memory. One can solve this problem by tiling also the connectivity information, as proposed next.

B. Tiled Connectivity Information

Tiling geometry samples is possible in a straightforward manner because the entire template mesh M_j^T at each resolution j is available for embedding the samples before (partial) decoding, *and* each wavelet coefficient corresponds to a single encoded sample. Connectivity information cannot be handled similarly as these two assumptions are no longer true.

Firstly, partial decoding of the connectivity information results in only partly upsampling M_j^T to $M_{\alpha \le j+1}^T$, breaking the symmetry with the encoder side. Denote the tiles obtained after tiling using any criterion as $\widetilde{T}_j^{\mathbf{x}}$. Contrary to the tiling of geometry samples in Section VI-A where M_j^T can be used *entirely* by both the encoder and decoder to create identical tiles, the tiling of connectivity samples which are only partly decoded can no longer depend on all template mesh information.

To tackle this, one approach is to tile the samples only once, as is done in literature and which results in *fixed tiles*. While this tiling operation is often performed on the base mesh, it can in general be performed at any resolution j. As the samples are not tiled at resolutions $k < j$, M_j^T will be entirely reconstructed by the decoder and the tiling operation can still use all template mesh information. Alternatively, we suggest a *dynamic tiling* approach. By allowing the amount of tiles to change per resolution level, the tiling can be adapted to the global average sampling density. Additionally, by allowing for non-uniform tiling, the tiling can be adapted to local sampling densities. In this approach, only the first tiling operation can use *all* template mesh information. Subsequent tiling operations can only consider information *local* to each tile in order to preserve the symmetry with the encoder. Consequently, either additional signalling in the data stream allows for uniform tiling, or decoder-side tiling decisions result in non-uniform tiling as only local information can be considered.

Secondly, whereas the geometry is decoded given a *single* sample $\omega(v_j^{o_i})$ per wavelet coefficient, properly upsampling the connectivity requires multiple samples $\beta(e_j^k)$ to define the wavelet coefficient supports. As these supports are only known to a decoder *after* decoding connectivity samples, a tiling of the connectivity samples *before* decoding them necessarily needs to duplicate vertices of neighbouring tiles to avoid patches being encoded only partially within a tile.

We discuss two approaches for extending the obtained tiles $\widetilde{T}_j^{\mathbf{x}}$ to account for patches across tile borders:

- extend tile samples with the minimal amount of samples such that each patch is fully represented (Section VI-B1),

- or extend tile samples such that all subsequent resolutions are decodable given the current tile (Section VI-B2).

The following two sections discuss these two approaches for extending the tiles $\widetilde{T}_j^{\mathbf{x}}$ to form the tiles $T_j^{\mathbf{x}}$ which are encoded.

Due to only partially reconstructing $M^T_{\alpha\leq j}$, the mapping of vertices to tiles $T(v_j) = T_j^{\mathbf{x}}$ is no longer straightforward. The vertices at resolutions where (fixed or dynamical) tiling is performed, are trivially mapped. However, for vertices v^o reconstructed at *higher* resolutions, $S(\omega(v^o))$ possibly lies across a tile border and is consequently duplicated across multiple tiles. In our implementation, higher-resolution even vertices v_j^e and odd vertices v_j^o are mapped as follows:

$$T(v_j^e) = T\big(\gamma(v_j^e)\big) \tag{24}$$

$$T(v_j^o) = \bigcup_{v\in S(\omega(v_j^o))} T(v) \tag{25}$$

That is, reconstructed vertices inherit the (possibly different) tiles of the vertices in their support. The set of required tiles $\mathcal{T}_j^{\text{req}}$ per resolution j can then be determined via:

$$\mathcal{T}_j^{\text{req}} = \bigcup_{v\in ROI_j^S} T(v) \tag{26}$$

where a single tile of each $T(v)$ suffices.

Contrary to (23) the required tiles can no longer be determined directly in the wavelet-domain ROI_j^W. As connectivity information is only partially decoded, not all $w_c \in C_j$, needed for determining $S(w)$ using (5), are known. Hence, ROI_j^W cannot be determined using (6). We decode the extended tiles corresponding to the vertices in the *spatial-domain* ROI_j^S. Hence, the extension of the tiles must ensure that sufficient samples are being decoded such that ROI_j^W can accurately be obtained for reconstructing $M_{\alpha\leq j+1}$.

1) Minimal connectivity information per tile: After partitioning, the vertices in patches which lie across tile borders are scattered over multiple tiles. Hence, decoding only a specific tile $\widetilde{T}_j^{\mathbf{x}}$ can result in inaccurate connectivity near the tile borders, which in turn results in drift when decoding geometry samples for tile $\widetilde{T}_j^{\mathbf{x}}$. The most straightforward approach encodes the tiles $T_j^{\mathbf{x}}$ which extend $\widetilde{T}_j^{\mathbf{x}}$ as follows:

$$\begin{aligned}&\widetilde{T}_j^{\mathbf{x}} \subset T_j^{\mathbf{x}} \text{ and}\\ &\forall w_j \in W_j : S(w_j) \cap \widetilde{T}_j^{\mathbf{x}} \neq \varnothing \implies S(w_j) \subset T_j^{\mathbf{x}}\end{aligned} \tag{27}$$

That is, if one vertex of the wavelet coefficient support $S(w_j)$ was partitioned into $\widetilde{T}_j^{\mathbf{x}}$, then the entire support needs to be encoded in $T_j^{\mathbf{x}}$.

Observe that this tiling cannot be mirrored by a decoder: $\widetilde{T}_j^{\mathbf{x}}$ can be determined in the same way as done at the encoder side, but without knowledge of W_j, $T_j^{\mathbf{x}}$ cannot be found. We propose to encode additional vertex rings per tile $\widetilde{T}_j^{\mathbf{x}}$ until $T_j^{\mathbf{x}}$ is entirely taken into account; the number of additional vertex rings is a parameter which needs to be encoded in the bitstream. Each encoded sample which is not found in $T_j^{\mathbf{x}}$ is encoded as a null-value to ensure no patches are determined outside of $T_j^{\mathbf{x}}$. The actual values for these samples are irrelevant for decoding $\widetilde{T}_j^{\mathbf{x}}$; if a decoder needs these values, the appropriate tile will be decoded.

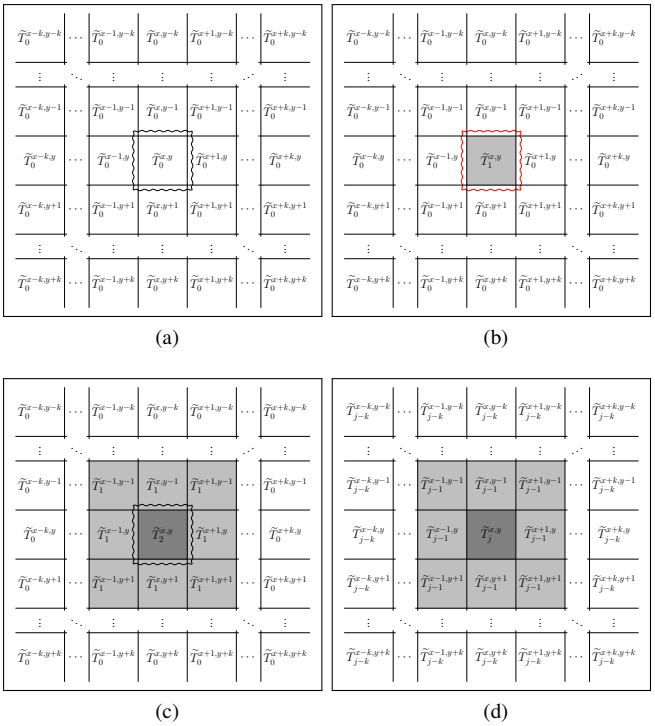


Fig. 5. Connectivity tiling with minimal duplication. In Fig. 5a, an adaptive inverse wavelet transform requires the data of subband 0, limited to tile $\widetilde{T}^{x,y}$. The black wave line indicates data outside of this tile required to properly reconstruct the template mesh $M^T_{\alpha\leq 1}$ which perfectly reconstructs M_1^T *within tile* $\widetilde{T}^{x,y}$. As M_0^T is at resolution 0 over all tiles, all required data is available. In Fig. 5b, data of subband 1 is required to reconstruct $M^T_{\alpha\leq 2}$, again limited to tile $\widetilde{T}^{x,y}$. Additional data outside of this tile (indicated by the red wave line) is no longer guaranteed to be available, as $M^T_{\alpha\leq 1}$ is not necessarily at resolution 1 in the tiles neighbouring $\widetilde{T}^{x,y}$. To ensure that all required data is available, the neighbouring tiles need to be at resolution 1 before reconstructing $M^T_{\alpha\leq 2}$ (Fig. 5c). In general, if data of subband j is required in a tile $\widetilde{T}^{x,y}$, tiles minimally need to be decoded as depicted in Fig. 5d.

The construction of the tiles $T_j^{\mathbf{x}}$ considering wavelet coefficient supports, as given in (27), ensures that decoding the tiles given in (26) provides the necessary wavelet coefficients of W_j for accurately determining ROI_j^W defined in (6), which are, in turn, required for reconstructing $M_{\alpha\leq j+1}$. However, as the encoder only takes into account W_j without considering higher resolutions, there is no guarantee that a subsequent decoding step of a higher resolution will be possible without drift.

Figure 5 illustrates this effect. Figure 5a shows all tiles of M_0^T. The wave line indicates the samples added to $\widetilde{T}_0^{x,y}$ to obtain $T_0^{x,y}$. For the next upsampling step, such additional samples are not necessarily available in Figure 5b as the neighbouring tiles $T_0^{x\pm1,y}$, $T_0^{x,y\pm1}$ and $T_0^{x\pm1,y\pm1}$ are not decoded, nor were these additional samples taken into account when constructing $T_0^{x,y}$ using (27). Consequently, these neighbouring tiles need to be decoded before $T_1^{x,y}$ can be decoded (Figure 5c). In general, if an ROI_j^S requires $\widetilde{T}_j^{x,y}$, the minimal resolutions are shown in Figure 5d: neighbouring tiles can differ by, at most, one resolution level.

This tiling approach has the disadvantage that much of the decoding effort is spent to support neighbouring tiles instead of the highest resolution *within* the tile. Furthermore, while dynamic tiling is possible, ensuring the appropriate

resolutions per tile becomes even more involved if tiles are non-uniformly distributed. To solve these issues, an alternative tiling methodology is presented next.

2) Sufficient connectivity information for independent tiles: Alternatively, given tiles $\widetilde{T}_j^{\mathbf{x}}$, we propose a tiling $T_j^{\mathbf{x}}$ which takes into account *all* higher resolutions. In a first step, we traverse all higher resolutions to find wavelet coefficients affected by vertices in $\widetilde{T}_j$, defining temporary tiles $\widehat{T}_k^{\mathbf{x}}$ for $k \geq j$. Let $\widehat{T}_j^{\mathbf{x}} = \widetilde{T}_j^{\mathbf{x}}$; $\widehat{T}_k^{\mathbf{x}}$ is recursively determined as follows:

$$\begin{aligned} &\forall v_k^e \in M_k^e : \gamma(v_k^e) \in \widehat{T}_{k-1}^{\mathbf{x}} \implies v_k^e \in \widehat{T}_k^{\mathbf{x}} \\ &\forall v_k^o \in M_k^o : S(\omega(v_k^o)) \cap \widehat{T}_{k-1}^{\mathbf{x}} \neq \varnothing \implies v_k^o \in \widehat{T}_k^{\mathbf{x}} \end{aligned} \tag{28}$$

for each $k \in [j{+}1, r_{\max}]$, with $r_{\max}$ the highest resolution. $\widehat{T}_{r_{\max}}^{\mathbf{x}}$ encompasses all highest-resolution vertices which are, directly or indirectly, affected by the vertices in $\widetilde{T}_j^{\mathbf{x}}$. To ensure that these highest-resolution vertices are accurately reconstructed, we traverse back to resolution j to ensure that all wavelet coefficient supports are taken into account. Define temporary tiles $\check{T}_k^{\mathbf{x}}$ for the traversal back to resolution j. Let $\check{T}_{r_{\max}}^{\mathbf{x}} = \widehat{T}_{r_{\max}}^{\mathbf{x}}$; we have:

$$\begin{aligned} &\forall v_{k+1}^e \in (\check{T}_{k+1}^{\mathbf{x}} \cap M_{k+1}^e) : \gamma(v_{k+1}^e) \in \check{T}_k^{\mathbf{x}} \\ &\forall v_{k+1}^o \in (\check{T}_{k+1}^{\mathbf{x}} \cap M_{k+1}^o) : S(\omega(v_{k+1}^o)) \in \check{T}_k^{\mathbf{x}} \end{aligned} \tag{29}$$

Using (29) we eventually find $\check{T}_j^{\mathbf{x}}$. The minimally required tile data which needs to be encoded is $T_j^{\mathbf{x}} = \check{T}_j^{\mathbf{x}}$. This ensures that, given an initial $\widetilde{T}_j$, sufficient additional samples are provided to decode this resolution j as in Section VI-B1, *and* to decode future resolutions $j + k$ as long as ROI_{j+k}^S is composed of vertices which are (directly or indirectly) affected by vertices in the original $\widetilde{T}_j^{\mathbf{x}}$. Additionally, as tiles can now be treated independently, tiling can be easily adapted to the decoded sample densities per resolution.

Similar to the approach in Section VI-B1, the decoder has no knowledge about W_k with $k \geq j$ and cannot reconstruct $T_j^{\mathbf{x}}$. We again consider the vertex rings around $\widetilde{T}_j^{\mathbf{x}}$ which need to be added until $T_j^{\mathbf{x}}$ is entirely taken into account. This number of vertex rings is communicated to ensure that the same tiles are used at the encoder and decoder sides. Finally, observe that we obtain the same $T_j^{\mathbf{x}}$ as in (27) if we only look one resolution higher, i.e., if we consider $r_{\max} = j + 1$ and thus only apply (28) and (29) once.

This approach lends itself perfectly to dynamic tiling, producing tiles of various sizes, adapted on the local density of the tessellation. In our implementation, the partitioning is based on the recursive octree decomposition of the bounding box of template mesh vertices. Let τ^{TS} be the given tile-split threshold which controls the amount of vertices within any given tile. The dynamic tiling algorithm initially considers all vertices to be contained within a single tile $\widetilde{T}_0^0 = T_0^0$. The tile partitioning approach considers the axis-aligned bounding box of all vertices within a tile, i.e., the box containing the points (x, y, z) with $x \in [x_{\min}, x_{\max}], y \in [y_{\min}, y_{\max}], z \in [z_{\min}, z_{\max}]$ and

$$x_{\min} = \min_{v \in T_j^0} \epsilon(v)_0 \text{ and } x_{\max} = \max_{v \in T_j^0} \epsilon(v)_0 \tag{30}$$

$$y_{\min} = \min_{v \in T_j^0} \epsilon(v)_1 \text{ and } y_{\max} = \max_{v \in T_j^0} \epsilon(v)_1 \tag{31}$$

Listing 2 Algorithm: Dynamic tile decoding

```
1:  T := {T_0^0}
2:  for all j ∈ [0, r_max − 1] do
3:    for all T_j ∈ T : T_j ∈ T_j^req do
4:      T := T \ {T_j}
5:      T_{j+1} = DECODEANDUPSAMPLE(T_j)
6:      if |T_j| < τ^TS then                          ▷ keep T_{j+1}
7:        T := T ∪ {T_{j+1}}
8:      else                                          ▷ split T_{j+1}
9:        v_C = ((x_min+x_max)/2, (y_min+y_max)/2, (z_min+z_max)/2)
10:       for all x ∈ {0,1}^3 do
11:         T̃_{j+1} = {v ∈ T_{j+1} : (−1)^{x_i}(v_{C,i} − ε(v)_i) > 0}
12:         T_{j+1}^x: obtained using (28) and (29)
13:         T := T ∪ {T_{j+1}^x}
14:       end for
15:     end if
16:   end for
17: end for
```

$$z_{\min} = \min_{v \in T_j^0} \epsilon(v)_2 \text{ and } z_{\max} = \max_{v \in T_j^0} \epsilon(v)_2 \tag{32}$$

In these notations $\epsilon(v)_i$ indicates the i^{th} component of the embedding $\epsilon(v) \in \mathbb{R}^3$.

The vertices are partitioned by considering eight octants around the bounding box center $\mathbf{v}_C$. The index of each of the octants is given by a triple $\mathbf{x} = (x_0, x_1, x_2)$ with $x_i \in \{0, 1\}$, where $x_i = 0$ indicates that the i^{th} component of each vertex position in the octant is smaller than $\mathbf{v}_{C,i}$, i.e., the i^{th} component of $\mathbf{v}_C$. Conversely, $x_i = 1$ indicates that the vertices have an embedded position with the i^{th} component larger than $\mathbf{v}_{C,i}$. This condition can be represented compactly:

$$\forall v \in \widetilde{T}_j^{x_0,x_1,x_2} : (-1)^{x_i}(\mathbf{v}_{C,i} - \epsilon(v)_i) > 0, i \in [0, 2] \tag{33}$$

The dynamic tile decoding algorithm is described in Listing 2. $\mathcal{T}$ represents the set of decodable tiles. At each resolution, ROI_j^S determines the required tiles $\mathcal{T}_j^{\text{req}}$ using (26). Each required tile that still needs to be decoded, denoted by T_j in Listing 2, is removed from the decodable tiles (line 4), is decoded and upsampled (line 5), and either the upsampled tile T_{j+1} is added to the decodable tiles as such (line 7) or T_{j+1} is first split and each of the subtiles are added to the decodable tiles (line 13).

An example illustrating an encoding and decoding of the proposed dynamic tiling approach is shown in Figure 6, where tiles are split binary; tile indices $\mathbf{x}$ are now scalar indices which stay constant between the resolution at which the tile is created and the resolution where the tile is further split. A base mesh is encoded using a single tile T^0. Tile T_0^0 encodes the first subband W_0, i.e., G_0 and C_0. Tiles T_1^0 and T_2^0 encode the next two subbands. However, the amount of samples after upsampling T_2^0 surpasses τ^{TS} so instead of encoding a single tile T_3^0, the tile is split and T_3^1 and T_3^2 are encoded instead (see Figure 6). Information is duplicated, such that each subtree is now processed independently. Because tiles are processed

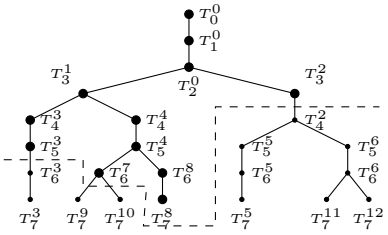


Fig. 6. Dynamic tiling. This figure visualizes tiles hierarchically. The dashed line represents a possible front of tiles which are decoded at a particular time.

independently they need not be split at the same resolution, allowing for the tile splitting to adapt to the sampling densities within the model.

The decoding process follows the same steps. The given ROI_j^S per resolution j determines the tiles $\mathcal{T}_j^{\text{req}}$ which need to be decoded, while no dependencies occur between neighbouring tiles. For instance, for a given ROI, tile T^8 can be decoded up to resolution 7 while the neighbouring tile T^2 can remain at resolution 3, without any blocking artefacts showing up due to our construction of the tiles. At each step of the decoding process, the state is represented by a *tile front* which is formed as illustrated by the dashed line in Figure 6, and corresponds with $\mathcal{T}$ in Listing 2. In this example, $\mathcal{T} = \{T_5^3, T_6^7, T_7^8, T_3^2\}$ represents the four tiles and their respective resolutions for decoding a specific mesh $M_{\alpha\leq 7}$. T_3^2 spans approximately half of the bounding box, T_5^3 spans a quarter and T_6^7 and T_7^8 span $1/8$, showing that the decoded tiles appropriately adapt to the sampling densities required for the requested ROIs.

Contrary to our approach in Section VI-B1, tiles are only decoded in order to supply wavelet coefficients for the mesh reconstruction within the tile and not as support for decoding neighbouring tiles located in the ROI. That is, given an ROI_j^S, only the tiles given by (26) need to be decoded, as sufficient samples are provided for decoding up to resolution j without drift. Compared to regular tiling of Section VI-B1, this reduces the amount of tiles being decoded for a given resolution level and improves random accessibility. This is illustrated in Figure 7 which compares the proposed dynamic tiling method (Figure 7b) with the regular tiling of Section VI-B1 (Figure 7a). One notices that providing any given resolution level requires less tiles when following dynamic tiling compared to the regular tiling of VI-B1.

Although the proposed method with independent tiles (Figure 7b) improves random accessibility compared to the method with minimally-sized tiles (Figure 7a), it also introduces an additional rate penalty. On the one hand, additional samples are taken into account further outside of the tile borders; this is illustrated by the overshoots of the dashed lines over the tile marks (indicated by the vertical lines) in Figure 7b. If tiles are not decoded until the highest resolution, some of these additional samples are irrelevant. On the other hand, information will be duplicated over several neighbouring tiles; this is indicated by the gray areas in Figure 7b which illustrate data near the tile borders that has been decoded twice.

Nonetheless, while the lossless coding rate increases due to duplicate information being encoded, the required rate when decoding specific ROIs is vastly lowered due to the tile resolutions scaling down as fast as the inverse wavelet transform does.

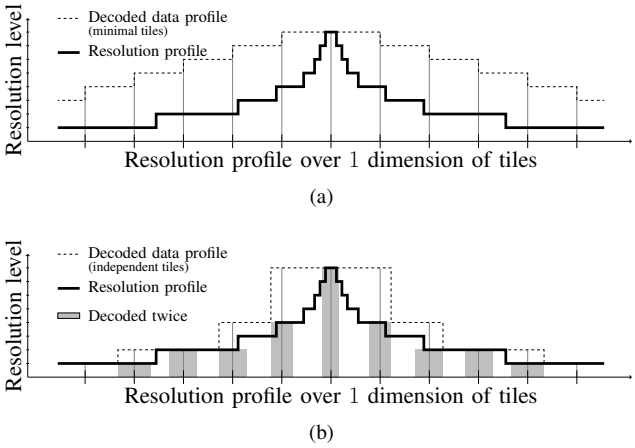


Fig. 7. Tile granularity. These two plots qualitatively compare the methods discussed in Sections VI-B1 and VI-B2 respectively. Ideally, the dashed line which represents the *decoded data* (at tile granularity) perfectly follows the full line which represents the amount of *data used* for the wavelet transform (at triangle granularity). The gray areas indicate information which is duplicated in order to provide random accessibility while avoiding tile dependencies.

VII. Rate-Distortion Optimized Encoding

With this tiling available, we propose a rate-distortion optimization of the coding system which aims at optimally allocating rate across different tiles. We have investigated rate-distortion optimization for *untiled* wavelet-based irregular mesh encoding in [35]. In this paper, we generalize this approach for dynamic tiling, as detailed next.

Without tiling, at each moment during encoding, either a new resolution is decoded up to a specific number of bitplanes, or the quality of an existing resolution is improved by encoding an additional number of bitplanes. This is formalized as follows. Consider all wavelet coefficients being represented using $p_{\max}$ bitplanes, and the connectivity information using an additional data layer. The data layers are labeled decreasingly as $p_{\max}$ for the connectivity information, and $p_{\max}-1, \ldots, 0$ for the geometry information from most significant to least significant bitplanes. This allows for constructing rate-distortion curves per resolution, where each curve has $p_{\max}+2$ ratepoints, one for each encoded data layer (including the case when no information is sent for the given resolution). Each ratepoint is denoted by $(R_p^{(j)}, D_p^{(j)})$, with $R_p^{(j)}$ the bitrate required to obtain the ratepoint with label p of resolution j, and $D_p^{(j)}$ the accompanying distortion.

Let P_j be the last encoded layer of resolution j, and let L be the first unencoded resolution (hence, $P_L = p_{\max} + 1$, the ratepoint *before* encoding any connectivity information); in general, when performing rate-distortion optimization, we want to encode k' layers of resolution j' by maximizing the distortion-rate slopes given by:

$$(j', k') = \underset{j\in[0,L], k\in[1,P_j]}{\arg\max} \frac{D_{P_j}^{(j)} - D_{P_j-k}^{(j)}}{R_{P_j-k}^{(j)} - R_{P_j}^{(j)}} \tag{34}$$

Per resolution j, the amount of data layers $k \in [1, P_j]$ which yields the largest slope is determined. We encode these data layers for the resolution for which this largest slope is maximal.

With the availability of tiles, rate-distortion optimization can be further improved as the tiles can be present at different resolutions and quality levels. Given that each resolution j counts T_j tiles, we can now find the rate-distortion curves per resolution *and* per tile. Each such curve has again $p_{\max} + 2$ ratepoints, denoted by $(R_p^{(j,t)}, D_p^{(j,t)})$. $P_{j,t}$ now signifies the last encoded layer *of tile* t in resolution j, and $R_p^{(j,t)}$ and $D_p^{(j,t)}$ respectively give the bitrate and distortion after encoding layer p *of tile* t at resolution j. We now want to encode k' layers of tile t' of the j'^{th} resolution by determining:

$$(j', t', k') = \underset{\substack{j\in[0,L]\\ t\in[0,T_j-1]\\ k\in[1,P_{j,t}]}}{\arg\max} \frac{D_{P_{j,t}}^{(j,t)} - D_{P_{j,t}-k}^{(j,t)}}{R_{P_{j,t}-k}^{(j,t)} - R_{P_{j,t}}^{(j,t)}} \tag{35}$$

In these equations, we make the following conventions (similar to the conventions in [35]):

$$R_{p_{\max}+1}^{(j,t)} = 0; \tag{36}$$

$$R_{\text{conn}}^{(j,t)} = R_{p_{\max}}^{(j,t)} - R_{p_{\max}+1}^{(j,t)} = R_{p_{\max}}^{(j,t)} \tag{37}$$

$$R_{\text{geom}}^{(j,t)} = R_0^{(j,t)} - R_{p_{\max}}^{(j,t)} \tag{38}$$

$$D_{p_{\max}+1}^{(j,t)} = D_{p_{\max}}^{(j)} \tag{39}$$

These conventions signify that the rate for each tile at each resolution starts at 0 as given by (36); the rate for each tile t of each resolution j can be attributed to connectivity information by the first data layer (37) and geometry information in the remaining layers (38). We do not consider any distortion decrease by decoding the connectivity information (39).

VIII. Experimental Results

For the evaluation of the proposed methods, we have experimented with models up to the *Asian Dragon* model of 3 609 600 vertices. In the literature, no proper evaluation criteria have been proposed for comparing different approaches w.r.t. the quality of ROI decoding or the accuracy offered by random accessibility. Comparative studies are usually limited to comparing the *lossless rates*, and *visual results* without rate indications. These visual results are obtained for instance using a click-and-drag approach in a virtual environment.

In this section we provide both lossless rates and visual results, but we also complement them by providing experimental results for two ROI decoding scenarios: *front-view ROI* and *point-based ROI*.

The *front-view ROI* selects all "front-facing" triangles. Let $\mathbf{n}$ be the surface normal of a triangle, and $\mathbf{v}$ the direction to the camera. A triangle is front-facing if $\mathbf{n}$ and $\mathbf{v}$ form an angle smaller than $90°$, signifying that the front of the triangle is visible when looking from the direction of the camera. For the visible triangles, the proposed ROI coding methodology should guarantee *visually* lossless results. Figure 8 shows examples for the *heptoroid*, *fertility* and *golfball* models. Depending on the model, front-facing triangles can either be found mainly in the front half of the model (e.g., *golfball*) or over the entire surface (e.g., *heptoroid*). To avoid selecting ROIs over the entire surface, we add experimental results where the front-view results are limited to those ROIs in the front half, indicated as "*front-view (half)*". The result for the fertility model is shown in Figure 8d.

The *point-based ROI* selects a random vertex at the base (lowest) resolution; at each resolution j those triangles surrounding this selected vertex are selected as part of ROI_j^S, while each higher resolution either keeps the same vertex or one of the newly created vertices to continue this process. Visual results with this method are shown in Figure 9.

Such experiments are sensitive to the orientation, geometry and topology of the models, but still give insights on how the proposed ROI decoding methodology performs. The proposed evaluation scenarios are considered as *soft extrema*. On the one hand, the *front-view* results give an estimation on the maximally useful ROIs, considering the camera position but disregarding occlusion and lower resolution requirements due to the distance to the camera. On the other hand, *point-based* decoding can be considered as an estimation on the minimally useful ROIs: only a single vertex is considered as ROI and the resolution over the reconstructed mesh surface is minimal in order to provide for a valid mesh topology.

A. Adaptive Inverse Wavelet Transform

This section evaluates the inverse wavelet transform proposed in Section V, which is independent of any tiling decisions. We investigate the amount of decoded vertices for the three ROI decoding scenarios, i.e. *front-view*, *front-view (half)* and *point-based*, indicating the ratio of ROI vertices w.r.t. the total amount of vertices as:

$$\rho = \frac{v_{ROI}}{v_{total}} \tag{40}$$

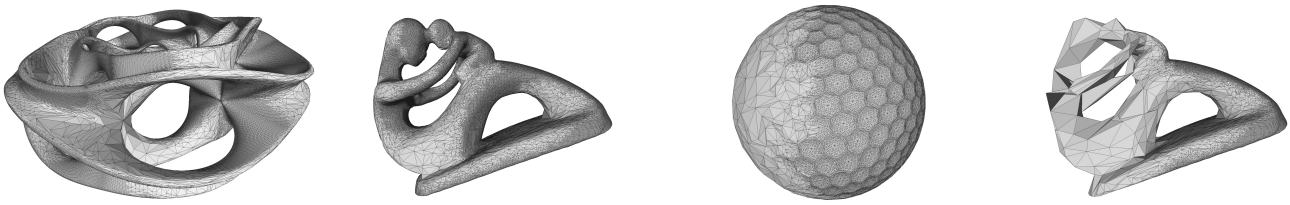
The results are shown in Figure 10. The lines connecting the samples have no significance but where added for clarity.

For front-view decoding, ρ converges to 50%, which was expected assuming that approximately half of the triangles are facing any camera in general. The overhead caused by the ROI propagation (see (17) and Figure 4c) is only significant for smaller models, where ρ goes up to 80%. For point-based decoding, ρ converges to 0% as the ROI, i.e., a single vertex, reduces relative to the full-resolution sizes.

Such accurate representations are possible because our wavelet transform does not depend on any tiling decisions. For tile-based solutions, the smallest ρ depends on the tiling granularity. These results are valuable from a rendering perspective: a reduction in the amount of vertices directly relates to the reduction in memory usage for real-time rendering. However, as mentioned at the end of Section V, without tiling, this still requires lossless decoding and substantial amounts of memory that linearly scale with the mesh sizes.

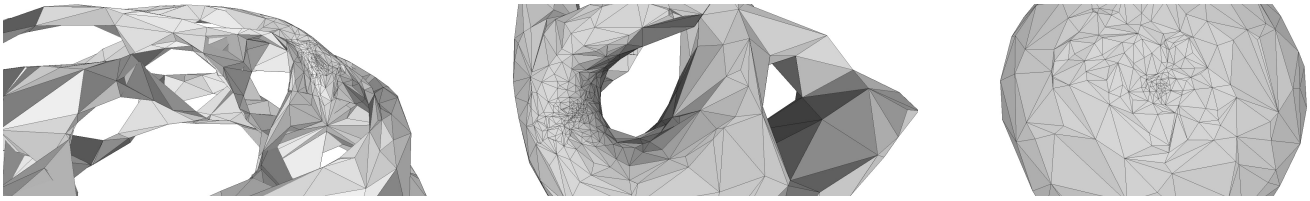
B. Dynamic Tile-based Coding

In the next set of experiments, we investigate the dynamic tiling discussed in Section VI, introduced to reduce the coding



(a) heptoroid (b) fertility (c) golfball (d) fertility (half)

Fig. 8. "*Front-view ROI*" (8a, 8b, 8c) and "*Front-view (half) ROI*" (8d).



(a) heptoroid (b) fertility (c) golfball

Fig. 9. Point-based ROI.

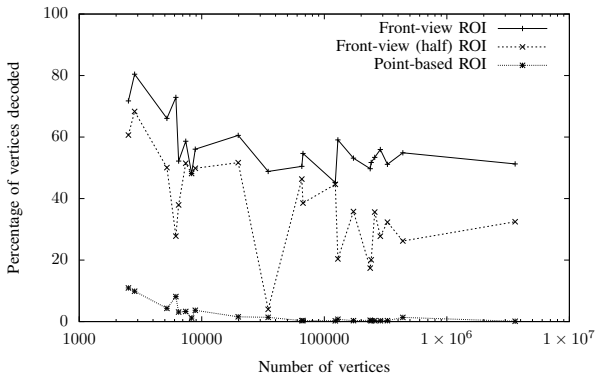


Fig. 10. Percentage of vertices of ROI-decoded models. This plot shows the percentage ρ of decoded triangles when (i) considering front-facing regions (Front-view ROI), converging to $\rho = 50\%$, (ii) front half regions (Front-view (half) ROI), and (iii) only considering the region surrounding a random point (Point-based ROI), converging to $\rho = 0$.

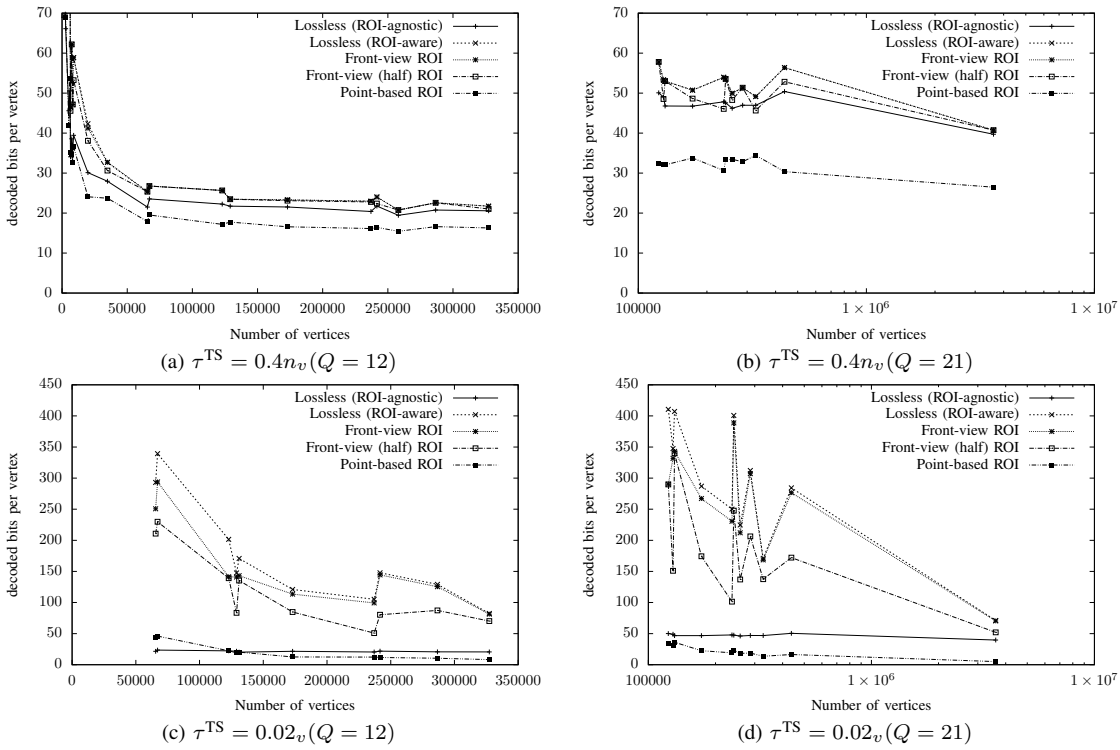
rate required by the ROI-aware inverse transform. We again consider the *front-view*, *front-view (half)* and *point-based* ROI coding scenarios. Additionally, we present the bitrates for lossless decoding, both with and without tiling, i.e., *ROI-aware* and *ROI-agnostic*. These figures will reveal the actual rate penalty introduced by tiling.

Given a fraction p, we evaluate the effect of splitting tiles by setting $\tau^{\text{TS}} = p.n_v$, with n_v the amount of vertices in a model. This results in similar tiling for all models, independent of the model sizes. Results are given for models up to $350\,000$ vertices at 12 bit quantization, and models up to over 3.6 million vertices at 21 bit quantization.

High values of p result in only splitting the tiles once. When splitting at $p = 40\%$ (see plots in Figures 11a and 11b), the single base tile (see Figure 6) will only be split at a high resolution. In general, none of the eight new tiles will surpass this τ^{TS} again, resulting in a highest resolution with eight tiles. We observe that the lossless penalty is minimal, which reduces with increasing model sizes. We also observe rate gains when decoding a single vertex. Finally, when increasing the amount of quantization bits, the obtainable rate gains also increase, showing that our approach is valuable for high-accuracy models. The gains are limited due to the large amount of data that is still encoded in a single tile.

For low values of p, tiles are split at a lower resolution in Figure 6. Figures 11c and 11d show the results when splitting at $p = 2\%$. Although we observe higher lossless rate penalties, we again observe that these penalties reduce with increasing model sizes: using smaller tile sizes (resulting in more tiles) requires larger models to be efficient. For ROI-decoding, however, much larger gains can be obtained. For smaller models the gains are reduced due to the ROI propagation (as was also mentioned in Section VIII-A); for larger models we observe significant gains. The *dragon* model of $437\,645$ vertices decodes the 21 bit quantized ROI around a single vertex at 16.4bpv (while lossless decoding requires 50.4bpv), which saves 67.5% of the lossless bitrate; given that $\tau^{\text{TS}} = 8\,753$ for this model, any ROI consisting of several thousands of vertices will be decoded at a similar bitrate. Similarly, the *Asian dragon* model of $3\,609\,600$ vertices decodes the ROI around a single vertex at 4.96bpv (of the 39.7 lossless bpv); as $\tau^{\text{TS}} = 72\,192$, ROIs consisting of tens of thousands of vertices will be decoded at a similar rate, saving 87.5% of the lossless bitrate. Finally, we remark that, with increasing model sizes, the expected ROI will decrease w.r.t. the total model size, which signifies that the actual rates will move closer towards the *point-based* results.

Notice that decoding the front-view ROI mostly coincides with lossless decoding. This confirms the fact that triangles which face the camera, i.e., front-facing triangles, are not necessarily restricted to the front half of a model, as was seen in Figures 8a and 8b. Consequently, depending on the tiling granularity, nearly all of the tiles can be required,



(a) $\tau^{\mathrm{TS}} = 0.4n_v (Q = 12)$ (b) $\tau^{\mathrm{TS}} = 0.4n_v (Q = 21)$

(c) $\tau^{\mathrm{TS}} = 0.02_v (Q = 12)$ (d) $\tau^{\mathrm{TS}} = 0.02_v (Q = 21)$

Fig. 11. Bitrates for decoding ROIs, for models encoded with relative tile sizes.

such that requiring all front-facing triangles at the highest resolution is detrimental for random access. Allowing only front-facing triangles *in the front half* for the ROIs, as depicted by the *front-view (half)* results, still more than half of the tiles are required due to the ROI propagation discussed in Section V; these will require lower-resolution triangles in the back half of the model to be upsampled, to ensure proper reconstruction of the front half without blocking artefacts. To allow for efficient streaming, the distance to the camera (i.e., using lower resolutions for far away regions) and the actual visibility (considering, for instance, the view frustum and self-occlusions) need to be taken into account.

Parallel decoding: An additional advantage of the proposed tiling approach is the unlocked parallel decoding opportunity. Let us consider an example where a single tile is split into eight tiles at some resolution $r_{\max} - k$. For instance, this is the case for $p = 40\%$ depicted in Figure 11a. The results show an average lossless rate penalty of 14.8 bpv when considering all models, or, by ignoring models with less than $20\,000$ vertices, an average rate penalty as low as 2.5 bpv.

Earlier, this rate penalty was already justified considering ROI decoding: for large models a relatively small ROI will be required. Consequently, for reconstructing ROI-aware versions of large models, the low value of ρ will result in reduced bitrates compared to ROI-agnostic coding.

In addition, if sufficient memory, processing power and bandwidth are available and the full model needs to be decoded losslessly, the k highest resolutions (i.e., the most computationally expensive resolutions) will benefit from a potential eightfold speedup if all eight new tiles are equally filled. This is made possible because the individual tiles can be decoded completely independently, even across resolutions. This is left as topic of further investigations.

IX. Conclusions

This work proposed a region-of-interest (ROI) based coding approach for irregular triangle meshes which allows for varying the resolution of a model over its surface at the finest granularity level. To allow for randomly accessing parts of the data, we have proposed a dynamic tiling in the wavelet domain where each tile can be independently processed. This allows for adapting the tiling to the sampling densities and the requested ROIs, while also permitting decoding speedups in the lossless case by allowing for parallel decoding. Despite rate penalties which are unavoidable when adding ROI support, we observe that decoding ROIs can be done at the fraction of the lossless rate, for increasingly larger models.

The results show that we are at the turning point where ROI support proves its value. In future work, a more efficient implementation will be able to process models which are several orders of magnitude larger; the main challenge will become the encoder which still needs to process the entire model, not just a selected ROI.

Acknowledgments

The research activities as described in this paper were funded by Ghent University, imec, Innoviris (3DLicornea), Flanders Innovation & Entrepreneurship (VLAIO), the Fund for Scientific Research-Flanders (FWO-Flanders), and the European Union.

Datasets are courtesy of the Stanford University, Georgia Institute of Technology, Princeton Graphic Group, INRIA, Cyberware and Aim@Shape.

REFERENCES

[1] H. Hoppe, "Progressive meshes," in *Proc. 23rd Annu. Conf. on Computer Graphics (SIGGRAPH)*, 1996, pp. 99–108.

[2] S. Valette, R. Chaine, and R. Prost, "Progressive lossless mesh compression via incremental parametric refinement," in *Computer Graphics Forum - Proc. Eurographics Symp. on Geometry Processing (SGP)*, 2009, pp. 1301–1310.

[3] M. Lounsbery, T. D. DeRose, and J. D. Warren, "Multiresolution analysis for surfaces of arbitrary topological type," *ACM Transactions on Graphics (TOG)*, vol. 16, no. 1, 1997.

[4] A. Khodakovsky, P. Schröder, and W. Sweldens, "Progressive geometry compression," in *Proc. 27th Annu. Conf. on Computer Graphics (SIGGRAPH)*, 2000, pp. 271–278.

[5] S. Valette and R. Prost, "Wavelet-based progressive compression scheme for triangle meshes: Wavemesh," *IEEE Trans. Vis. Comput. Graphics*, vol. 10, no. 2, 2004.

[6] L. Denis, S. M. Satti, A. Munteanu, J. Cornelis, and P. Schelkens, "Scalable intraband and composite wavelet-based coding of semiregular meshes," *IEEE Trans. Multimedia*, vol. 12, no. 8, 2010.

[7] J. El Sayeh Khalil, A. Munteanu, L. Denis, P. Lambert, and R. Van de Walle, "Scalable feature-preserving irregular mesh coding," *Computer Graphics Forum*, vol. 36, no. 6, 2017.

[8] J. Askelöf, M. L. Carlander, and C. Christopoulos, "Region of interest coding in JPEG 2000," *Signal Processing: Image Communication*, vol. 17, no. 1, 2002.

[9] H. Zheng, B. Liu, and H. Zhang, "Region-of-interest coding of 3D mesh based on wavelet transform," in *Image and Graphics (ICIG), 3rd Int. Conf. on*, 2004, pp. 438–441.

[10] J. El Sayeh Khalil, A. Munteanu, and P. Lambert, "3d mesh coding with predefined region-of-interest," in *Proc. 24th IEEE Int. Conf. on Image Processing (ICIP)*, 2017, pp. 1422–1426.

[11] E. Gobbetti, F. Marton, P. Cignoni, M. Di Benedetto, and F. Ganovelli, "C-BDAM — compressed batched dynamic adaptive meshes for terrain rendering," *Computer Graphics Forum*, vol. 25, no. 3, 2006.

[12] L. M. Hwa, M. A. Duchaineau, and K. I. Joy, "Adaptive 4-8 texture hierarchies," in *IEEE Visualization 2004*, 2004, pp. 219–226.

[13] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, and R. Scopigno, "Planet-sized batched dynamic adaptive meshes (P-BDAM)," in *Proc. of the IEEE Conf. on Visualization (VIS)*, 2003, pp. 147–154.

[14] H. Hoppe, "View-dependent refinement of progressive meshes," in *Proc. 24th Annu. Conf. on Computer Graphics (SIGGRAPH)*, 1997, pp. 189–198.

[15] L. Hu, P. V. Sander, and H. Hoppe, "Parallel view-dependent level-of-detail control," *IEEE Trans. Vis. Comput. Graphics*, vol. 16, no. 5, 2010.

[16] P. Gioia, O. Aubault, and C. Bouville, "Real-time reconstruction of wavelet-encoded meshes for view-dependent transmission and visualization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 7, 2004.

[17] M. Roy, S. Foufou, and F. Truchetet, *Multiresolution Analysis for Irregular Meshes with Appearance Attributes*. Springer Netherlands, 2006.

[18] J. Ho, K.-C. Lee, and D. Kriegman, "Compressing large polygonal models," in *Proc. of the IEEE Conf. on Visualization (VIS)*, 2001, pp. 357–573.

[19] S. Choe, J. Kim, H. Lee, S. Lee, and H.-P. Seidel, "Mesh compression with random accessibility," in *Israel-Korea Bi-National Conf*, 2004, pp. 81–86.

[20] S. Choe, J. Kim, H. Lee, and S. Lee, "Random accessible mesh compression using mesh chartification," *IEEE Trans. Vis. Comput. Graphics*, vol. 15, no. 1, 2009.

[21] S.-E. Yoon and P. Lindstrom, "Random-accessible compressed triangle meshes," *IEEE Trans. Vis. Comput. Graphics*, vol. 13, no. 6, 2007.

[22] M. Isenburg, P. Lindstrom, and J. Snoeyink, "Streaming compression of triangle meshes," in *Proc. 3rd Eurographics Symp. on Geometry Processing (SGP)*, 2005.

[23] B. Liu and H.-B. Zhang, "Wavelet based progressive mesh compression with random accessibility," in *Proc. 2004 Int. Symp. on Intelligent Multimedia, Video and Speech Processing, 2004.*, 2004, pp. 615–618.

[24] C. Roudet, F. Dupont, and A. Baskurt, "Semi-regular 3D mesh progressive compression and transmission based on an adaptive wavelet decomposition," in *Wavelet Applications in Industrial Processing VI - Electronic Imaging Science and Technology Symp.*, 2009, pp. 724 807–724 807–12, vol. 7248.

[25] Z.-Q. Cheng, H.-F. Liu, and S.-Y. Jin, "The progressive mesh compression based on meaningful segmentation," *The Visual Computer*, vol. 23, no. 9, 2007.

[26] P. Alliez and M. Desbrun, "Progressive compression for lossless transmission of triangle meshes," in *Proc. 28th Annu. Conf. on Computer graphics and interactive techniques (SIGGRAPH)*, 2001, pp. 195–202.

[27] A. Maglo, I. Grimstead, and C. Hudelot, "Cluster-based random accessible and progressive lossless compression of colored triangular meshes for interactive visualization," in *Proc. Computer Graphics Int. 2011 Conf.*, 2011, pp. 1–8.

[28] H. Lee, G. Lavoué, and F. Dupont, "New methods for progressive compression of colored 3D mesh," in *Proc. 18th Int. Conf. on Computer Graphics, Visualization and Computer Vision (WSCG)*, 2010, pp. 199–206.

[29] A. Maglo, I. Grimstead, and C. Hudelot, "POMAR: Compression of progressive oriented meshes accessible randomly," *Computers & Graphics*, vol. 37, no. 6, 2013.

[30] Z. Du, P. Jaromersky, Y.-J. Chiang, and N. Memon, "Out-of-core progressive lossless compression and selective decompression of large triangle meshes," in *2009 Data Compression Conf.*, 2009, pp. 420–429.

[31] P.-M. Gandoin and O. Devillers, "Progressive lossless compression of arbitrary simplicial complexes," in *Proc. 29th Annu. Conf. on Computer graphics and interactive techniques (SIGGRAPH)*, 2002, pp. 372–379.

[32] J. Kim, S. Choe, and S. Lee, "Multiresolution random accessible mesh compression," *Computer Graphics Forum*, vol. 25, no. 3, 2006.

[33] C. Courbet and C. Hudelot, "Random accessible hierarchical mesh compression for interactive visualization," in *Computer Graphics Forum - Proc. Eurographics Symp. on Geometry Processing (SGP)*, 2009, pp. 1311–1318.

[34] I. Daubechies and W. Sweldens, "Factoring wavelet transform into lifting steps," *The Journal of Fourier Analysis and Applications*, vol. 4, pp. 247–269, 1998.

[35] J. El Sayeh Khalil, A. Munteanu, and P. Lambert, "Rate-distortion optimized wavelet-based irregular mesh coding," in *Proc. 12th Int. Joint Conf. on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 1: GRAPP (VISIGRAPP)*, 2017, pp. 212–219.

[36] J. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Process.*, vol. 41, no. 12, 1993.

[37] A. Maglo, C. Courbet, P. Alliez, and C. Hudelot, "Progressive compression of manifold polygon meshes," in *Proc. 15th Shape Modeling Int. Conf. (SMI)*, 2012, pp. 349–359.

Jonas El Sayeh Khalil received the M.Sc. degree in computer science engineering from Ghent University, Ghent, Belgium, in 2012. Afterwards, he started working towards the Ph.D. degree at the Internet Technology and Data-Science Lab (IDLab) of Ghent University-imec, obtaining his Ph.D. degree in 2018. His research topics and interests include 3D mesh representations, coding and transmission, computer graphics and real-time rendering.

Adrian Munteanu (M07) is professor at the Electronics and Informatics department of the Vrije Universiteit Brussel, Belgium. His research interests include image, video and 3D graphics coding, distributed visual processing, 3D graphics, multimedia transmission over networks, and statistical modeling.

Adrian Munteanu is the author of more than 300 journal and conference publications, book chapters, and contributions to standards and holds 7 patents in image and video coding. He is the recipient of the 2004 BARCO-FWO prize for his PhD work, and of several prizes and scientific awards in international journals and conferences. Adrian Munteanu served as Associate Editor for IEEE Transactions on Multimedia.

Peter Lambert currently is a full-time Associate Professor at the Internet Technology and Data Science Lab (IDLab) of Ghent University-imec (Belgium) since 2013. He received his Master's degree in science (mathematics) and in applied informatics from Ghent University in 2001 and 2002, respectively, and he obtained the Ph.D. degree in computer science in 2007 at the same university. In 2009 he became a Technology Developer at Ghent University, which he combined with a part-time Assistant Professorship since 2010. His research interests include (mobile) multimedia applications, multimedia coding and compression, and 3D graphics.