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.
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 adapt 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 \( M_{\text{enc}} \), \( M^\text{TF} \) 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 \((x, y, z)\) couple corresponds to, at most, a single point 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 \( M^\text{TF} \) in Figure 1, requiring full decoding of \( M_{\text{enc}} \).

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 quantization. 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 neighboring 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.

### 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 $w_e \in C_{j-1}$ 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 $M_{j-1}$ is maintained, representing all connectivity information in the original mesh. As both $M_{j-1}$ and $M_{j-1}'$ share the same
connectivity, there exists a bijective mapping \( \mu \) linking the vertices of both, that is:

\[
\forall v \in M_{j-1} : \mu(v) \in M^T_{j-1} \quad (1)
\]

\[
\forall v^T \in M^T_{j-1} : \mu^{-1}(v^T) \in M_{j-1} \quad (2)
\]

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 \( M^e_i \) and \( M^o_i \) represent the even and odd sets of vertices respectively in \( M_i \) as obtained by the splitting step in Figure 2. After downsampling, retriangulating and updating, the vertices \( v^e_i \in M^e_i \) form the vertices \( \gamma(v^e_i) = v^i \) of \( M_{j-1} \), where \( \gamma \) is the bijective operator that maps \( v^i_j \in M^e_i \) to \( v^i_{j-1} \in M_{j-1} \). We define the wavelet coefficients corresponding to the odd vertices \( v^o_j \in M^o_j \) as \( \omega(v^o_j) = w^i_{j-1} \in W_{j-1} \), where \( \omega \) is the bijective operator that maps \( v^i_j \in M^o_j \) to \( v^i_{j-1} \in W_{j-1} \).

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

\[
v^o_i = \left( \sum_{k \in |M_{j-1}|} w^e_k v^k_{j-1} \right) + w_o \quad (4)
\]

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 \( v^k_{j-1} \) of the lower-resolution mesh by weights given by the connectivity information in \( w^i_{j-1} \). The second term corrects this prediction.

We now define the support \( S(w_{j-1}) \) for a wavelet coefficient \( w^i_{j-1} \) as the set of vertices with a non-zero weight in (4):

\[
S(w_{j-1}) = \{ v^k_{j-1} \in M_{j-1} : w^e_k \neq 0 \} \quad (5)
\]

That is, \( S(w_{j-1}) \) corresponds to the set of vertices \( v^k_{j-1} \in M_{j-1} \) required for reconstructing \( v^o_i = \omega^{-1}(w^i_{j-1}) \).

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

\[
ROI^{W}_j = \{ w \in W_j : S(w) \cap ROI^S_j \neq \emptyset \} \quad (6)
\]

i.e., \( ROI^{W}_j \) is the subset of those wavelet coefficients in \( W_j \) whose support overlaps with the given \( ROI^S_j \). Additionally, we define the support of the set \( ROI^{W}_j \) as:

\[
S(ROI^{W}_j) = \bigcup_{w \in ROI^{W}_j} S(w) \quad (7)
\]

To accommodate for the varying resolution across a surface, we introduce the notations \( M_{\alpha \leq j} \), \( M_{\beta \leq j} \),... 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 \( ROI^{W}_j \), we generalize (3) as follows:

\[
M_{\alpha \leq j+1} = WT^{-1}(M_{\alpha \leq j}, ROI^{W}_j) \quad (8)
\]

with \( S(ROI^{W}_j) \subset M_{\alpha \leq j} \).

The requirement in (8) states that if the support for each wavelet coefficient \( w \in ROI^{W}_j \) 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 \( ROI^S_k \) 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 \( ROI^{W}_j \), 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 \( ROI^S_j \) can be reconstructed after upsampling \( ROI^S_k \) \( \forall k < j \), the propagation operator \( \sigma \) is introduced, propagating \( ROI^S_j \) at resolution \( j \) to the region \( \sigma(ROI^S_j) \) at resolution \( j-1 \): \( \sigma(ROI^S_j) = \{ v \in M_{j-1}, \exists \forall v^i \in (ROI^S_j \cap M^i_j) : v = \gamma(v^i) \} \cup \{ v \in M_{j-1}, \exists \forall v^o \in (ROI^S_j \cap M^o_j) : v = S(\omega(v^o)) \} \quad (9) \)

That is, all even vertices in \( ROI^S_j \) are preserved in \( \sigma(ROI^S_j) \) while odd vertices require the support of the corresponding wavelet coefficients to be present as well. Requiring that \( \sigma(ROI^S_{k+1}) \subseteq ROI^S_k \forall k < j \) ensures that any reconstruction
\(M_{\alpha \leq j}\) can be obtained with ROIs properly reconstructed. ROIs, \(\sigma(ROI_j^S)\) and \(ROI_j^{S-1}\) 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:

\[
\begin{align*}
\forall w &\in ROI_j^W : s_jw \in W_j^B \\
\forall w &\in W_j \setminus ROI_j^W : w \in W_j^B
\end{align*}
\]  
(10)

The original wavelet coefficients can be obtained as:

\[
\begin{align*}
\forall w &\in W_j^B \land w \geq s_j : w/s_j \in W_j \\
\forall w &\in W_j^B \land w < s_j : w \in W_j
\end{align*}
\]  
(11)

Biplane 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 \(\tilde{M}_j^o\) of reconstructed odd vertices creates all vertices when upsampling:

\[
\tilde{M}_j^o = \{w^{-1}(w) : w \in W_{j-1}\}
\]  
(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:

\[
\tilde{M}_j^o = \{w^{-1}(w) : w \in ROI_{j-1}^W \subset W_{j-1}\}
\]  
(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\(_{k}^{S}\), 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\(_{j\beta}^{S}\) 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\(_j^S\):

\[
ROI_{j\beta}^S = ROI_j^S \cup S(ROI_j^W)
\]  
(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\(_{j\beta-1}^{S}\), expanded to ROI\(_{j-1\beta}^{S}\), in order to ensure its accurate reconstruction at resolution \(j\). One can write:

\[
ROI_{j-1\beta}^S = ROI_j^S \cup S(ROI_j^W)
\]  
(15)

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

Recursively, to reconstruct \(M_{\beta \leq j}\) given this further expanded ROI\(_{j-1\beta}^{S}\), \(M_{\alpha \leq j-1}\) 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\(_{j-2\beta}^{S}\), with:

\[
ROI_{j-2\beta}^S = ROI_j^S \cup \sigma(ROI_{j-1\beta}^{S})
\]  
(16)

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

\[
ROI_{k\beta}^S = ROI_{k+1\beta}^S \cup \sigma(ROI_{k+1\beta}^{S})
\]  
(17)

The expansion ROI\(_{k\beta}^{S}\) of ROI\(_k^{S}\) given the newly determined ROI\(_j^S\) is the union of its expansion ROI\(_{k-1\beta}^{S}\) complying for the already known ROI\(_{j-1}^{S}\), and the propagation \(\sigma(ROI_{k+1\beta}^{S})\) of the higher-resolution expansion ROI\(_{k+1\beta}^{S}\).

Given the spatial-domain ROI\(_k^{S}\) with wavelet-domain ROI\(_j^W\) obtained via (6), and \(S(ROI_j^W) \notin M_{\alpha \leq j}\), the adaptive inverse wavelet transform can now be recursively defined to obtain \(M_{\beta \leq j}\):

\[
M_{\beta \leq 1} = WT^{-1}(M_0, ROI_{0\beta}^W)
\]  
(18)

\[
M_{\beta \leq k} = WT^{-1}(M_{\beta \leq k-1}, ROI_{k-1\beta}^W)
\]  
(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
performed steps are described in Listing 1. Observe that the mapping \( \mu \) is no longer surjective if \( M_{\alpha \leq j} \neq M_j \):

\[
v^T \in M_j^T \Rightarrow \exists v \in M_{\alpha \leq j} : \mu(v) = v^T
\]

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 \( ROI^S \) only the wavelet coefficients \( w_j \in ROI^W_j \) and \( w_k \in ROI^W_{j,k} \) 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-

**Listing 1** Example: Adaptive inverse wavelet transform

1: function UPSAMPLE(ROI^S_0)
2: given M_0 and ROI^S_0 ▷ (Fig. 4a)
3: \( ROI^W_0 = \{w^0_1, w^0_2, w^0_3\} \)
4: \( S(ROI^W_0) = S(w^0_2) \subset M_0 \) ▷ (Fig. 4b)
5: return \( M_{\alpha \leq 1} = WT^{-1}(M_0, ROI^W_0) \) ▷ (Fig. 4c)
6: end function

7: function UPSAMPLE(ROI^S_1)
8: given ROI^S_1 ▷ (Fig. 4b)
9: \( ROI^W_1 = \{w^1_1, w^1_2, w^1_3\} \)
10: \( S(ROI^W_1) = S(w^1_2) \cup S(w^1_3) \cup S(w^1_1) \) ▷ (Fig. 4f)
11: \( \Rightarrow S(ROI^W_1) \nsubseteq M_{\alpha \leq 1} \) ▷ (Fig. 4b)
12: function UPSAMPLE(ROI^S_0^{ROI^S_1})
13: \( ROI^W_{0^{ROI^S_1}} = ROI^W_0 \cup \sigma(ROI^W_{1^{ROI^S_1}}) \) ▷ (Fig. 4a)
14: \( ROI^W_{0^{ROI^S_1}} = \{w^0_1, w^0_2, w^0_3\} \)
15: \( S(ROI^W_{0^{ROI^S_1}}) = S(w^0_2) \cup S(w^0_3) \cup S(w^0_1) \) ▷ (Fig. 4g)
16: return \( M_{\beta \leq 1} = WT^{-1}(M_0, ROI^W_{0^{ROI^S_1}}) \) ▷ (Fig. 4c)
17: end function
18: return \( M_{\beta \leq 2} = WT^{-1}(M_{\beta \leq 1}, ROI^W_1) \) ▷ (Fig. 4d)
19: end function

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 ROIs are shaded in dark gray, and the corresponding \( S(ROI^W) \) are depicted on the bottom line.
mation \( w_c \in C_j \) is represented by assigning a binary value \( \beta(e^k_j) \) to each edge \( e^k_j \), indicating whether or not the edge is preserved when upsampling. This is shown in Figure 4e for \( M^T_o \) 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 \( v_{\tilde{o}}^{j+1} \) is added per patch, and the patch is retriangulated (depicted in Figure 4f). The geometry information \( w_y \in G_j \) 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, and can consequently be mapped to a single tile after partitioning. At each resolution \( j \), the connectivity samples can be tiled. A partial, ROI-based decoding of these geometric samples does not hinder subsequent decoding steps as each 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 \( M^T_j \) to \( M^T_{\alpha \leq j+1} \), breaking the symmetry with the encoder side. Denote the tiles obtained after tiling using any criterion as \( T^j_\alpha \). Contrary to the tiling of geometry samples in Section VI-A where \( M^T_j \) 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 \( k < j \), \( M^T_j \) 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 \( \omega(v_{\tilde{o}}^o) \) per wavelet coefficient, properly upsampling the connectivity requires multiple samples \( \beta(e^k_j) \) 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 \( T^j_\alpha \) 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 \( \tilde{T}_j^x \) to form the tiles \( T_j^x \) which are encoded. Due to only partially reconstructing \( M_{\alpha \leq j} \), the mapping of vertices to tiles \( T(v_j) = T_j^x \) is no longer straightforward. The vertices at resolutions where (fixed or dynamical) tiling is performed, are trivially mapped. However, for vertices \( v_j^o \) reconstructed at higher resolutions, \( S(\omega(v_j^o)) \) 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:

\[
T(v_j^e) = T(\gamma(v_j^e))
\]

\[
T(v_j^o) = \bigcup_{v \in S(\omega(v_j^o))} T(v)
\]

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

\[
T_j^\text{req} = \bigcup_{v \in ROI_j^S} T(v)
\]

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 \( w_j \in C_j \), 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 \( \tilde{T}_j^x \) can result in inaccurate connectivity near the tile borders, which in turn results in drift when decoding geometry samples for the tile \( T_j^x \). The most straightforward approach encodes the tiles \( T_j^x \) which extend \( \tilde{T}_j^x \) as follows:

\[
\tilde{T}_j^x \subset T_j^x \quad \text{and}
\forall w_j \in W_j : S(w_j) \cap \tilde{T}_j^x \neq \emptyset \implies S(w_j) \subset T_j^x
\]

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

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

The construction of the tiles \( T_j^x \) 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 \( M_0 \). The wave line indicates the samples added to \( \tilde{T}_0^{x,y} \) 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 \( T_0^{\pm 1,x,y} \) and \( T_0^{x,y,\pm 1} \) 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 \( T_0^{x,y} \) can be decoded (Figure 5c). In general, if an \( ROI_{j}^S \) requires \( T_j^{x,y} \), 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 $T^x_j$, we propose a tiling $T^x_j$ which takes into account all higher resolutions. In a first step, we traverse all higher resolutions to find wavelet coefficients affected by vertices in $T_j$, defining temporary tiles $T^x_k$ for $k \geq j$. Let $T^x_j = T^x_j$; $T^x_k$ is recursively determined as follows:

\[
\forall v^x \in M^x_k : \gamma(v^x) \in T^x_{k-1} \implies v^x \in T^x_k
\]

\[
\forall v^x \in M^x_k : S(\omega(v^x)) \cap T^x_{k-1} \neq \emptyset \implies v^x \in T^x_k
\]

for each $k \in [j+1, r_{\text{max}}]$, with $r_{\text{max}}$ the highest resolution. $T^x_{\text{max}}$ encompasses all highest-resolution vertices which are, directly or indirectly, affected by the vertices in $T^x_j$. 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 $T^x_j$ for the traversal back to resolution $j$. Let $T^x_{\text{max}} = T^x_{\text{max}}$; we have:

\[
\forall v^x_{k+1} \in (T^x_{k+1} \cap M^x_{k+1}) : \gamma(v^x_{k+1}) \in T^x_k \nexists v^x_{k+1} \in (T^x_{k+1} \cap M^x_{k+1}) : S(\omega(v^x_{k+1})) \cap T^x_{k-1} \neq \emptyset
\]

Using (29) we eventually find $T^x_j$. The minimally required tile data which needs to be encoded is $T^x_j = T^x_j$. This ensures that, given an initial $T^x_j$, 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 $ROI^x_j + k$ is composed of vertices which are (directly or indirectly) affected by vertices in the original $T^x_j$. 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 $T^x_j$. We again consider the vertex rings around $T^x_j$ which need to be added until $T^x_j$ is entirely taken into account. This number of vertex rings is communicated to ensure that the highest-resolution vertices are accurately reconstructed, the tile is split 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 $T^{\tau_{\text{TS}}}$ 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 $T^0_0 = T^0_0$. 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 $x \in [x_{\text{min}}, x_{\text{max}}], y \in [y_{\text{min}}, y_{\text{max}}], z \in [z_{\text{min}}, z_{\text{max}}]$ and

\[
x_{\text{min}} = \min_{v \in T^0_0} \epsilon(v)_0 \quad \text{and} \quad x_{\text{max}} = \max_{v \in T^0_0} \epsilon(v)_0
\]

\[
y_{\text{min}} = \min_{v \in T^0_0} \epsilon(v)_1 \quad \text{and} \quad y_{\text{max}} = \max_{v \in T^0_0} \epsilon(v)_1
\]

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

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

\[
\forall v \in T^0_0 : \epsilon(v)_i \in (\epsilon(v)_i - \epsilon(v)_i) > 0, i \in [0, 2]
\]

The dynamic tile decoding algorithm is described in Listing 2. $T'$ represents the set of decodable tiles. At each resolution, $ROI^x_j$ determines the required tiles $T^x_j$ 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 $x$ 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 $T^0_0$. 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 $\tau_{\text{TS}}$ so instead of encoding a single tile $T^3_0$, the tile is split and $T^3_0$ and $T^3_0$ are encoded instead (see Figure 6). Information is duplicated, such that each subtree is now processed independently. Because tiles are processed
individually 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\(^j\) per resolution \(j\) determines the tiles \(T_{\text{req}}^j\) 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 \(T\) in Listing 2. In this example, \(T = \{T_5^3, T_6^3, T_9^2, T_3^1\}\) represents the four tiles and their respective resolutions for decoding a specific mesh \(M_{\leq 7}\). \(T_3^2\) spans approximately half of the bounding box, \(T_5^3\) spans a quarter and \(T_6^3\) and \(T_9^2\) 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\(^j\), 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.

![Dynamic tiling](image)

**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.

![Resolution profile](image)

**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 \(p_{\text{max}}\) bitplanes, and the connectivity information using an additional data layer. The data layers are labeled decreasingly as \(p_{\text{max}}\) for the connectivity information, and \(p_{\text{max}}-1, \ldots , 0\) for the geometry information from most significant to least significant bitplanes. This allows for constructing rate-distortion curves per resolution, where each curve has \(p_{\text{max}}+2\) 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, \(P_L = p_{\text{max}}+1\), 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:

\[
(j', k') = \arg \max_{j \in [0,L], k \in [1,P_j]} \frac{D_p^{(j)} - D_p^{(j)-k}\text{ }k}}{R_p^{(j)} - R_p^{(j)-k}}
\]

(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 $p_{\text{max}} + 2$ 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:

$$
(j', t', k') = \arg \max_{j \in [0, J], t \in [0, T_j - 1], k \in [1, P_{j,t}]} \frac{D_j^{(j,t)} - D_{j,t-k}^{(j,t)}}{R_j^{(j,t)} - R_{j,t-k}^{(j,t)}} \quad (35)
$$

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

$$
\begin{align*}
R_p^{(j,t)}_{\text{max}} + 1 &= 0; \\
R_{\text{geom}}^{(j,t)} &= R_{\text{geom}}^{(j,t)} - R_{\text{geom}}^{(j,t)}_{\text{max}} + 1 = R^{(j,t)}_{\text{max}} \\
R_{\text{geom}}^{(j,t)} &= R_{0}^{(j,t)} - R_{\text{geom}}^{(j,t)}_{\text{max}} \\
D_p^{(j,t)}_{\text{max}} + 1 &= D_{\text{max}}^{(j,t)} \\
\end{align*} \quad (36) \quad (37) \quad (38) \quad (39)
$$

These conventions signify that the rate for each tile at each resolution starts at 0 as given by (36); the rate for each tile 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 $n$ be the surface normal of a triangle, and $v$ the direction to the camera. A triangle is front-facing if $n$ and $v$ 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 $j$, 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:

$$\rho = \frac{v_{\text{ROI}}}{v_{\text{total}}} \quad (40)$$

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

For front-view decoding, $\rho$ 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 $\rho$ goes up to 80\%. For point-based decoding, $\rho$ 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 $\rho$ 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
Fig. 8. “Front-view ROI” (8a, 8b, 8c) and “Front-view (half) ROI” (8d).

Fig. 9. Point-based ROI.

Fig. 10. Percentage of vertices of ROI-decoded models. This plot shows the percentage $\rho$ 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 $\rho = 0$.

Given a fraction $p$, we evaluate the effect of splitting tiles by setting $\tau_{TS} = p.n_v$, 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 $\tau_{TS}$ 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 437645 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 $\tau_{TS} = 8753$ for this model, any ROI consisting of several thousands of vertices will be decoded at a similar bitrate. Similarly, the Asian dragon model of 3609600 vertices decodes the ROI around a single vertex at 4.96bpv (of the 39.7 lossless bpv); as $\tau_{TS} = 72192$, 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,
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 $r$.

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


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 video 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.