ARTICLE IN PRESS

Computer-Aided Design ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Perception-driven adaptive compression of static triangle meshes☆

Stefano Marras [a,*], Libor Váša [b], Guido Brunnett [b], Kai Hormann [a]

[a] Università della Svizzera italiana, Lugano, Switzerland

[b] Technical University Chemnitz, Chemnitz, Germany

HIGHLIGHTS

- New adaptive compression method based on shape perception.
- Suitable for both single rate and progressive encoding.
- Works with any kind of perceptual error metric.
- Improvement with respect to most commonly used approaches.

ARTICLE INFO

Keywords:
Mesh compression
Shape perception
Progressive decoding

ABSTRACT

Mesh compression is an important task in geometry processing. It exploits geometric coherence of the data to reduce the amount of space needed to store a surface mesh. Most techniques perform compression employing a uniform data quantization over the whole surface. Research in shape perception, however, suggests that there are parts of the mesh that are visually more relevant than others. We present a novel technique that performs an adaptive compression of a static mesh, using the largest part of the bit budget on the relevant vertices while saving space on encoding the less significant ones. Our technique can be easily adapted to work with *any* perception-based error metric. The experiments show that our adaptive approach is at least comparable with other state-of-the-art techniques, while in some cases it provides a significant reduction of the bitrate of up to 15%. Additionally, our approach provides much faster decoding times than comparable perception-motivated compression algorithms.

1. Introduction

Mesh compression algorithms deal with the problem of efficiently storing surface meshes, which mainly consist of triangles. Usually a certain loss in the precision of the data is allowed in order to achieve a smaller size of the compressed data. The task at hand is to encode the given mesh with smallest possible distortion, that is, to represent the input triangle mesh $\mathcal{M}$ with a shortest possible sequence of bits, from which a reconstruction $\bar{\mathcal{M}}$ can be decoded, while the difference $d(\mathcal{M}, \bar{\mathcal{M}})$ is lower than some user-defined constant. Usually, mesh connectivity and mesh geometry are encoded separately. The connectivity encoding is lossless and efficient algorithms have been proposed that work near the theoretical performance limit [1].

Mesh compression plays the key role in several industries, such as computer aided design, in film and gaming industry, e-marketing and many others. The specifics of a particular application area play a key role in designing a compression algorithm. In general, there are two main scenarios where mesh compression is required: storage of surface models for further computer-aided processing, and storage of triangle meshes for viewing by human observers. These two scenarios pose radically different requirements on the desired properties of the algorithm.

When storing meshes for further processing, it is desirable to keep the precision loss at a minimum, since the following steps may emphasize some compression artefacts that are on the verge of being visible at the current stage. The allowed precision loss in this scenario is justified and guided by the discrepancy between the precision of acquisition (scanner accuracy) and representation (length of mantissa of the used floating point representation of coordinates). In this scenario, it is possible to express the loss, that is, the function $d(\mathcal{M}, \bar{\mathcal{M}})$, in terms of vertex displacements, and it is desirable that both encoding and decoding are performed very fast, since storing occurs equally frequently as decoding during the usual workflow.

☆ This paper has been recommended for acceptance by Dr. Vadim Shapiro.

* Corresponding author.
E-mail addresses: stefano.marras@usi.ch, mr.marras@gmail.com (S. Marras).

http://dx.doi.org/10.1016/j.cad.2014.08.002
0010-4485/

Please cite this article in press as: Marras S, et al. Perception-driven adaptive compression of static triangle meshes. Computer-Aided Design (2014), http://dx.doi.org/10.1016/j.cad.2014.08.002

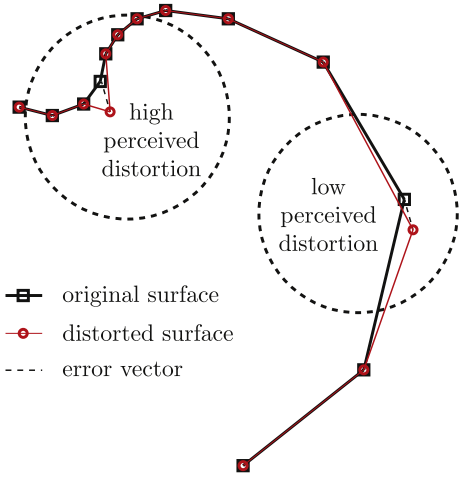

Fig. 1. Differently perceived distortion in different parts of the mesh, caused by the same vertex dislocation.

In this paper, we are going to address the second scenario, that is, storing meshes before the final delivery to a consumer, which is a human observer. This is for example the case in e-commerce (displaying a 3D model in a webshop) or gaming (storing a 3D model on a distribution medium) applications. This scenario has the following key properties:

- The decompressed mesh is required to be *perceptually close* to the original, that is, the allowed precision loss is justified and guided by the discrepancy between representation precision and the level of perceptible difference. Research in perceptual 3D metrics shows that visual similarity of 3D shapes correlates only weakly with vertex dislocations, and thus it must be measured using different approaches.
- The computational complexity of the *decompression is of much higher importance* than the computational complexity of the compression. Similarly to video encoding, longer processing times for encoding are acceptable, as long as the decoding can be done in real time.
- For some applications, such as e-commerce, it might be important to provide the user with the possibility to view a partially decoded mesh and the possibility to stop the decoding before the whole mesh is transmitted.

Out of these properties, the need for a perceptual metric is probably the one that has the largest influence on the design of a compression algorithm. Recently, a number of researchers [2–4] have focused on the problem of estimating the visual effects of mesh processing, including mesh simplification, watermarking, and compression. Metrics have been proposed that correlate well with the results of user studies in which human observers were asked to evaluate the amount of distortion in 3D meshes. Interesting phenomena have been identified and documented, such as the masking effect (increased sensitivity of the observers to distortion in smooth areas of the mesh) and generally higher sensitivity of the human visual system to local, relative changes in vertex positions, contrasting with the relatively low sensitivity to global changes in absolute vertex positions.

Mesh compression is usually applied in combination with mesh simplification, which reduces the number of vertices where possible. Consequently, the input data in most practical compression tasks exhibit a rather high variability in terms of sampling density and local smoothness. These factors contribute immensely to the local perceptibility of compression artefacts. For example, dislocating a vertex in a densely sampled area of the mesh can lead to a local flip of triangle normals, which is immediately visible to any human observer. On the other hand, dislocating a different vertex, this time in a sparsely sampled area of the mesh, by the same amount, may be completely imperceivable (see Fig. 1). This is the main motivation for our proposed algorithm.

Recent experiments with perceptual metrics also indicate that even the traditional metrics based on vertex dislocations are relevant for perception in some situations. Typically, when multiple objects interact by touching (shaking hands, feet touching floor), a too large absolute dislocation becomes observable and is perceived as an artefact. Our secondary motivation is therefore to design the algorithm so that the root-mean-squared error is reduced as well.

In this paper, we propose a triangle mesh compression algorithm with the following key features:

- **Adaptability** to any error metric of choice, investing bits only in areas where refinement is needed in order to reduce the distortion. In particular, adaptability to perceptually motivated metrics has been implemented and tested.
- **Improved performance** in traditional absolute metrics with respect to other perception-motivated algorithms.
- **Fast decoding** procedure, which only involves local operations on the triangle mesh. This allows application of our approach for larger meshes, where the decoding computational complexity makes competing algorithms impractical.
- **Progressive decoding** support.

Our algorithm builds on the parallelogram-based prediction strategy, but only uses this approach to build a coarse reconstruction of the mesh. Subsequently, an error metric of choice is used to determine where refinement bits are required in order to suppress perceivable artefacts. Our procedure is able to exploit the spatial coherence of mesh vertices by using the efficient parallelogram prediction for high-order bits, and at the same time it is able to distribute the bit budget efficiently across parts of the mesh that have different properties.

2. Related work

Mesh compression has been in the focus of researchers for many years. Many algorithms have been proposed, building on various concepts, and providing different properties and performance.

One of the first attempts at mesh compression is the *Topological Surgery* algorithm by Taubin and Rossignac [5]. It encodes the mesh connectivity represented by a vertex spanning tree and a triangle spanning tree. The mesh geometry is then encoded during a traversal of the vertex spanning tree, using a local prediction that exploits the spatial coherence in positions of neighbouring vertices.

Touma and Gotsman later proposed a more efficient connectivity encoding, based on storing vertex valences [6]. This approach has been extended [7] and related to theoretical bounds of lossless connectivity encoding [8,1]. Other mesh connectivity encoding schemes have been proposed as well, such as the *EdgeBreaker* scheme [9] with a guaranteed upper bound on the number of bits per vertex, the *cut-border machine* [10] or triangle-fan-based encoding [11], and *progressive meshes* [12] for progressively transmitting a mesh.

Several approaches have been proposed for the geometry encoding as well. Probably the best known approach is the parallelogram prediction, proposed by Touma and Gotsman [6]. The key idea is to traverse the mesh, expanding the part known to the decoder by one vertex at a time. The vertex is then predicted to lie at the tip of a parallelogram expanded from a known adjacent triangle. This approach has been applied together with many of the connectivity encoding approaches, where in some cases it can be used in a single mesh traversal that encodes the mesh connectivity and geometry simultaneously.

Several improvements of the parallelogram scheme have been proposed in order to improve its performance while preserving its

Please cite this article in press as: Marras S, et al. Perception-driven adaptive compression of static triangle meshes. Computer-Aided Design (2014), http://dx.doi.org/10.1016/j.cad.2014.08.002

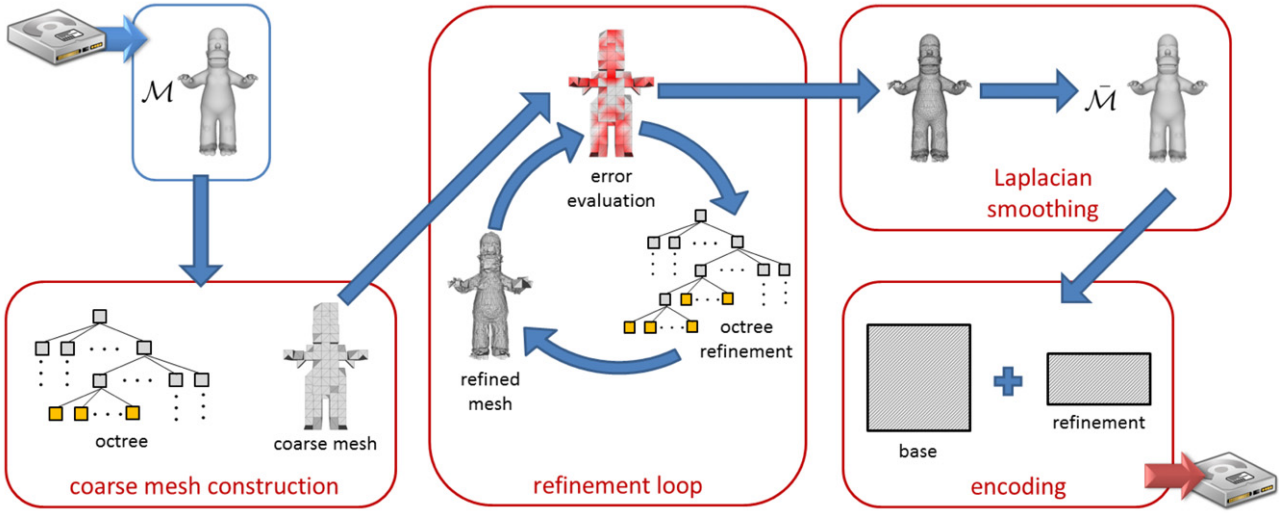

Fig. 2. Flowchart of the encoding process.

desirable properties, such as simplicity, locality, and low computational cost. Sim et al. [13] proposed using two parallelograms at the same time whenever it is possible to perform a prediction from two adjacent triangles. Courbet and Hudelot [14] generalize the parallelogram predictor to a more general Taylor prediction scheme. Recently, Váša and Brunnett [15] proposed using a weighted parallelogram scheme, building on an estimate of the interior angles of the triangles involved in the prediction.

Apart from extensions of the parallelogram prediction, several approaches have been proposed for geometry encoding, building on different concepts. Peng and Kuo [16] use an octree structure to encode the vertex positions in a way similar to our proposed algorithm, however their rules for the octree structure refinement are globally set and do not follow specific goals with respect to any perceptual error metric. Payan and Antonini [17] use wavelet decomposition to perform a frequency analysis of the geometry data. Similarly, Karni and Gotsman [18] find a set of eigenfunctions of the discrete combinatorial Laplacian of the mesh connectivity, and express the geometry in terms of this basis.

Sorkine et al. [19] proposed an approach based on discrete Laplace transformation, using the Kirchhoff Laplacian. The transformed vertex coordinates (denoted delta coordinates) are then quantized and transmitted to the decoder together with the positions of several so-called anchor points, which allow inverting the procedure, reconstructing the original coordinates from the delta coordinates. Note that many of the more advanced approaches are much more computationally expensive than the parallelogram based methods, requiring to solve a large sparse system of linear equations [19] or a large eigen-decomposition problem [18] at the decoder.

Apart from the so-called single-rate algorithms, which only allow decoding the mesh at a single quality level, algorithms have been also proposed that provide a series of approximations with increasing quality (number of vertices, vertex position precision) from a single data stream. This additional functionality usually comes at the cost of impaired absolute performance with respect to the single rate encoders [12], although recent work reports comparable results [20–22].

For quite a long time, the amount of distortion introduced by compression has been either measured as a sum of magnitudes of vertex dislocations, or using the Hausdorff distance. Although these measures are intuitive, it has been noted in [18,19] that these measures do not reflect the perceived amount of distortion. In order to alleviate the problem, the so-called visual error has been proposed in [18] and later slightly updated in [19].

Recently, however, more serious effort has been invested into the search for a mesh metric that correlates well with human evaluation of mesh distortion. It has been noted, that although the visual error represents an improvement over traditional measures, such as mean squared error (MSE) or Hausdorff distance, it still provides only a rather weak correlation with perception, especially in experiments where multiple types of distortion (for example stemming from various types of compression methods) are compared against each other.

This has motivated the emergence of several mesh comparison methods, aiming at providing a better estimate of mesh quality. Approaches based on curvature differences [3], roughness differences [23] and dihedral angle differences [4] have been proposed, providing high correlation with results captured in user studies. For a summary and comparison of these methods, we refer to a recent study of this topic by Corsini et al. [24].

3. Algorithm

The main idea behind our approach stems from the following observations:

- All perception-driven metrics tend to evaluate the visual error on a local neighbourhood of a vertex, edge, or face of the mesh.
- The visual error is usually not uniformly distributed over the surface of the mesh and the same vertex displacement can be perceived differently in different regions of the mesh (see Fig. 1).

Our aim is to distinguish between *high error* and *low error* regions, investing a different amount of bits to encode them. Regions where the error is small will be encoded with fewer bits, while regions where the error is higher will consume more bits. We assume that the mesh connectivity is encoded using any efficient connectivity coder, and that it is available at the decoder. Our algorithm stores the input mesh in terms of a *base mesh*, which is a low-precision version of the source mesh, plus a set of *refinement* bits, which improve the visual quality of the shape in the areas where such improvement is needed. We use an octree data structure to select an appropriate precision for each vertex. The vertex precision is represented by its depth in the tree, which is also encoded as auxiliary data. Note that the octree structure is only used as a temporary representation of the data, and it is *not* used to encode the mesh geometry or connectivity as done in previous work [16].

We first focus on how to build the geometry of the base mesh and the refinement bits, and then we describe in more detail how each of the two parts is encoded. The algorithm is summarized in Fig. 2.

Please cite this article in press as: Marras S, et al. Perception-driven adaptive compression of static triangle meshes. Computer-Aided Design (2014), http://dx.doi.org/10.1016/j.cad.2014.08.002

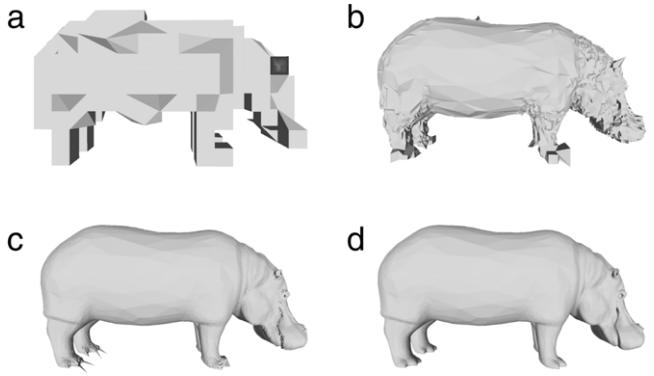

Fig. 3. The steps of the algorithm for the Hippo mesh. Initially, a coarse mesh (a) is obtained after 4 split operations of the octree root. Then, according to the DAME metric, vertices belonging to large and flat regions are refined first (b), thus significantly reducing the overall error. The precision of the remaining vertices is later iteratively increased (c) until the final configuration, corresponding to the target error, is reached (d).

3.1. Coarse mesh construction

The inputs of our algorithm are: a static mesh $\mathcal{M}$, a visual error metric, and a target error value. In order to encode the mesh, we proceed in the following way. First, we find the minimum cubic bounding box containing $\mathcal{M}$. We associate the bounding box with the root node of a common *octree*. In the beginning, the vertices of $\mathcal{M}$ are associated with the root node, since they all lie inside the bounding box. A *split* operation of an octree node subdivides the node into 8 child nodes and subdivides the ancestor cell into 8 equally sized cubic cells, one for each child node. The vertices belonging to the ancestor node are distributed among the child nodes according to their spatial position. To distinguish between the children, we assign to each new node a unique 3-bit binary code, from 000 to 111.

In the first phase of the process, we build a *coarse* version of $\mathcal{M}$ by recursively splitting the root node until a fixed depth t is reached for each node and then moving each vertex to the centre of the cell where it lies. At the end of this phase we have a mesh which is usually characterized by both a high objective and subjective error, with many vertices belonging to the same node collapsed to a single point (see Fig. 3). Each vertex is associated with a leaf and each leaf is characterized by its depth value plus a string of $3t$ bits that uniquely identifies it, which is the juxtaposition of the binary codes describing the path from the root to the leaf. The value t is usually selected between 2 and 5.

3.2. Refinement loop

As mentioned above, the coarse mesh is usually characterized by a large *visual error*, paired with an *objective error*, that is, the displacement in vertex positions (see Fig. 3). Here is where the refinement comes into play. The coarse mesh is compared to the source mesh $\mathcal{M}$ by evaluating the error according to the selected metric. Each element of the mesh (vertex, edge, or face) is then characterized by a certain amount of distortion. While objective metrics tend to uniformly distribute the error over the whole surface, for perceptually-motivated metrics, such as the KG-error [18], MSDM [2,3], FMPD [25], and DAME [4], the error is higher in the *perceptually relevant* regions and lower in the remaining parts of the surface.

We therefore select α% of the mesh elements, characterized by the highest error values, with α usually between 1 and 10. The idea is to reduce the global error by *refining* only the selected elements, increasing the precision of the regions that were identified as highly distorted by the error metric. To increase the precision of a vertex, we split the node it belongs to, therefore increasing the depth of the node and bringing its vertices closer to their position in the original mesh. The closer the vertices are to their final position, the lower the associated distortion gets. After the cells have been refined, we move the vertices to the centre of their cells and compare the newly refined mesh against $\mathcal{M}$, that is, we recompute the error characterizing each element. We continue to iteratively refine the mesh until the global error reaches the given target error value.

The results of different steps of the process are depicted in Fig. 3. The efficiency of the refinement loop can be improved by using a priority queue, sorted according to the local error, and by updating only the error of the previously refined elements. We can also modify the refinement condition by imposing a maximum tree depth, thus avoiding vertex over-refinement.

3.3. Laplacian smoothing

At the coarsest levels it is quite common to have a large number of duplicate vertices and zero-length edges. These artefacts are usually responsible for high visual errors. In order to reduce the distortion, we perform a post-processing step to enhance the quality of the mesh. In particular, a *constrained* Laplacian smoothing is applied to the reconstructed surface in the following way.

For each vertex v_i we compute

$$c_i = \frac{1}{n_i}\sum_{j\in N_i} p_j \quad \text{and} \quad d_i = c_i - p_i,$$

where p_i is the spatial position of v_i and N_i is its one-ring neighbourhood with cardinality n_i. The aim of the smoothing step is to adjust the position of each vertex v_i while keeping each vertex inside its cell. To determine the final position of v_i, we displace p_i by the amount of λ_i along d_i. We initially set $\lambda_i = 1$ and get the candidate position $p'_i = p_i + \lambda_i d_i$. If p'_i lies inside the cell of p_i, we move the vertex to its final position, otherwise we halve the value of λ_i and recompute p'_i, iterating the process until we find a valid candidate position. All vertices are translated to their new positions at the same time. If required, the smoothing step can be executed iteratively, even though one or two iterations are enough in most cases. A comparison between different levels of smoothing of the same mesh is depicted in Fig. 4.

Laplacian smoothing significantly reduces the visual distortion, especially at the coarser levels, while exploiting all the information known about the vertex positions (represented by their cell boundaries) and keeping the displacement error bounded. Vertices that are perceptually relevant are characterized by higher depth, which means smaller cells and, as a consequence, a smaller freedom in the displacement. However, after a certain level of precision is reached, the cells become too small and the smoothing no longer significantly affects the quality of the results. Notice that the smoothing step does not require storing any additional data, apart from the number of iterations, since it is based on a purely combinatorial Laplace operator. Fig. 5 shows the impact of Laplacian smoothing on a typical rate–distortion plot.

3.4. Encoding

At the end of the refinement process, each vertex v_i is characterized by a bit-string of length $3d_i$ where d_i is the depth of the leaf containing v_i. Since different leaves may have different depths, the lengths of the strings are usually different from vertex to vertex. Vertices that have been refined multiple times and hence belong to high-error regions of the mesh are characterized by a higher number of bits.

In general, the depth distribution over the tree for a target error (see Fig. 6) exhibits a peak in the middle, with only a fraction of the

Please cite this article in press as: Marras S, et al. Perception-driven adaptive compression of static triangle meshes. Computer-Aided Design (2014), http://dx.doi.org/10.1016/j.cad.2014.08.002

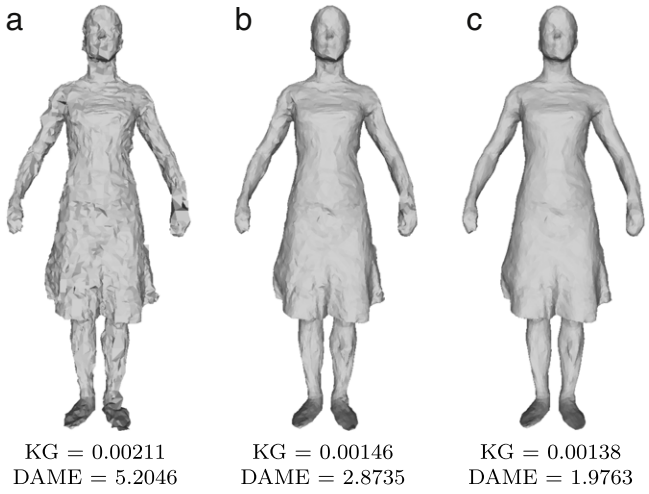

Fig. 4. Effect of smoothing for the Samba dataset. The mesh (a) is obtained by shifting the vertices to the centres of their octree cells, and it is affected by visual artefacts. A first iteration of the constrained Laplacian smoothing, (b) significantly reduces the visual error (30% in KG error, 45% in DAME error), making the mesh more visually pleasing. A second iteration (c) further improves the quality of the shape, but this time the error reduction is smaller (only 5% in the KG error, 31% in the DAME error).

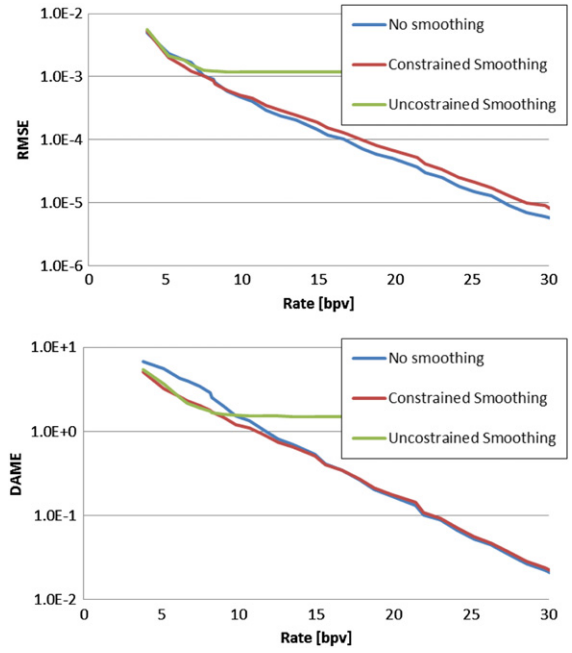

Fig. 5. Constrained Laplacian smoothing affects the rate–distortion plot of the horse dataset. Particularly, while only marginally affecting the objective error (top), it significantly reduces the perceptual error (bottom). We also compare the constrained Laplacian smoothing against the standard, unconstrained Laplacian smoothing. While the two approaches produce similar results at coarser levels, the unconstrained smoothing fails to converge to the original shape. In fact, the displacement of the vertices makes it impossible to further reduce the error in terms of both RMSE and DAME. This is the main reason behind our choice of constraining the displacement inside the cells.

vertices actually requiring further refinement. At the same time, there is a subset of vertices that do not require to be extremely refined, in the sense that increasing the precision of their position will not significantly affect the measured error. Therefore, these vertices are characterized by a smaller depth.

We select a depth value k, usually corresponding to the integer round of the average depth, as level of depth of the so-called *base mesh*. That means that the first k bits of each string associated with

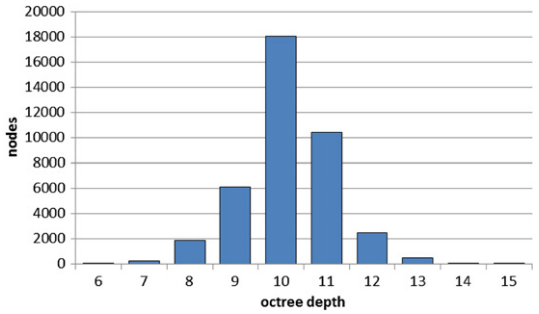

Fig. 6. Depth distribution for vertices of the James mesh (cf. Fig. 7), as the result of our adaptive compression, optimized for a DAME error of 0.001. The depth of the tree nodes is spread between 6 and 15, with a peak at depth 10. The average depth is 10.15, which means that, on average, each vertex requires about 30 bits to be stored. Encoding the same mesh, at the same distortion, using the parallelogram technique requires about 33 bits per vertex. The depth distribution over the surface is depicted in Fig. 7.

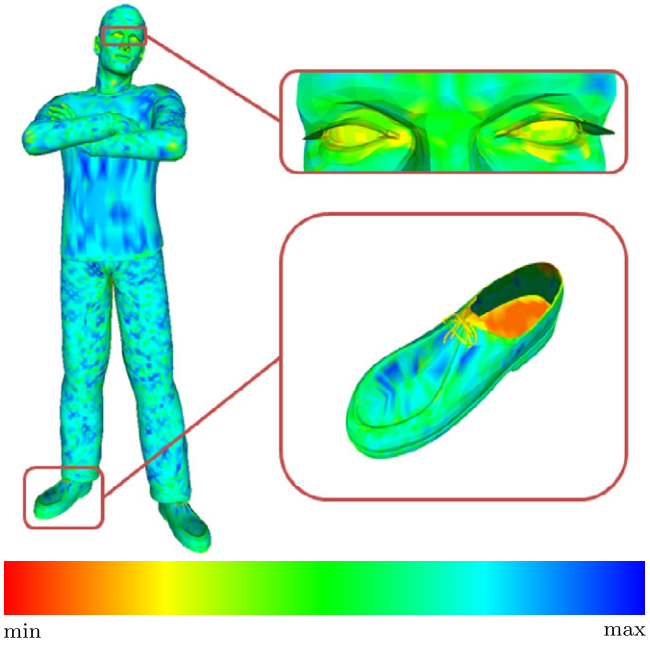

Fig. 7. Colour-coded depth distribution over the James mesh as a result of the adaptive compression, optimized for the DAME error. Parts of the shape that are characterized by low visibility, such as the eyes, usually have a small depth. Parts that are not visible at all, such as the interior of the shoe, are characterized by minimal depth, since their contribution to the error is basically zero. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

a vertex are used to encode the base mesh, while the remaining bits become the refinement. If a vertex has a bit-string with less than k elements, the string is extended by adding 0-bits until the length k is reached. These bits are used for encoding purposes only and do not have any influence on the final placement of the vertices, while only marginally affecting the compression rate. It is important to remark that the base mesh is usually different from the coarse mesh built at the beginning of the process, which is usually significantly coarser.

Obviously, the depth of each vertex depends on how each vertex affects the error computation. The DAME metric, for example, gives a low score to vertices that are not visible, while the FMPD metric gives a low score to vertices in flat regions. Fig. 7 shows the depth distribution over the surface of a mesh for the example in Fig. 6.

To encode the base mesh, we split each bit-string into 3 separate strings, representing the path of the vertex along the x-, y- and z-axis, then interpreting each sequence as an integer value. Each

Please cite this article in press as: Marras S, et al. Perception-driven adaptive compression of static triangle meshes. Computer-Aided Design (2014), http://dx.doi.org/10.1016/j.cad.2014.08.002

vertex is therefore characterized by three values corresponding to its spatial coordinates. The base mesh is encoded using a standard parallelogram predictor technique, applied directly on the integer values. The prediction residuals are binarized using the exp-Golomb code and encoded by a context adaptive binary arithmetic coder (CABAC) [26], which exploits their strongly biased distribution.

We then proceed with encoding the refinement bits, iteratively visiting the vertices of the mesh and storing, for each vertex, the first not-processed bit (if any). At the end of each iteration, the bits are encoded again using the CABAC coder. Note that the used prediction only influences (biases) the probability distribution of the higher order bits, that is, the ones that make up the base mesh. The refinement bits, on the other hand, have an almost unbiased probability distribution and regardless of whether a parallelogram-like prediction had been applied to them or not.

Finally, we need to additionally encode and transmit depth information in order to provide all the necessary information to the decoder. Depth information is encoded in the parallelogram prediction fashion. Additional information, such as the size of the bounding box, its centre, and the number of smoothing iterations, is transmitted without quantization, but it does not significantly affect the final compression rate.

3.5. Decoding

The decoding process is significantly faster than the encoding step, since it does not involve the evaluation loop. The decoder receives and decodes, in a single pass, both depth information and integer values representing the coordinates of each vertex up to depth k. In case a vertex is characterized by a depth smaller than k, we simply discard the unused bits. Each integer is then converted to a binary code, using both the decoded information and the remaining refinement bits.

The advantage of the decoding step is that it does not need to build the full octree to recover the final positions of the vertices. Suppose that we decoded a sequence of d bits $b_1^x, \ldots, b_d^x$, representing the path from the root of the tree to the final node along the x-axis. We also decoded the length of the cubic bounding box edge l. The x-coordinate p_x is computed as

$$p_x = \left(\sum_{i=1}^{d} b_i^x \frac{l}{2^i} \right) + \frac{l}{2^{d+1}},$$

with the last term representing the shifting to the centre of the cell. Following the same procedure, we recover also the y- and z-coordinates. To speed up the decoding process, we precompute all the possible values of $l/2^i$ once, up to the maximum sequence length. The bounding box of the mesh is centred in the origin, therefore a translation may be required to reach the correct position of the shape.

When all vertices have been placed, the Laplacian smoothing is performed for a given number of iterations. Again, the constrained Laplacian smoothing is performed without explicitly building the octree, bounding the maximum displacement in vertex position according to the depth of the vertex.

4. Results

In this section we present the results of our algorithm in terms of rate–distortion (R–D) plots. We mainly focus on the error evaluated in terms of visual distortion. Although various texturing techniques are often used to support efficient mesh simplification, we focus solely on geometric error metrics. Nevertheless, our algorithm supports using any, even image-based error metrics, provided that the error can be evaluated locally on each vertex, edge, or face, and therefore it could generally also be used to drive mesh compression with texture. Our technique also outperforms other existing approaches in terms of objective error, measured as root-mean-squared error (RMSE) of vertex positions, even though the algorithm is not optimized in that sense.

In our experiments we employ the original Karni–Gotsman error (KG) [18], the Dihedral Angle Mesh Error (DAME) [4], and the Fast Mesh Perceptual Distance (FMPD) [25]. All these metrics focus on perceptual error, with the partial exception of the KG metric, which also takes into account the absolute vertex displacement. The meshes we use in our experiments are from the Aim@Shape repository and the dataset in [27].

The KG metric measures the error related to vertex v_i as the average of its absolute displacement and of a distortion of a visual component called *smoothness*. The smoothness of v_i is computed as

$$s(v_i) = p_i - \frac{\sum_{v_j \in N_i} l_{ij}^{-1} p_j}{\sum_{v_j \in N_i} l_{ij}^{-1}},$$

where N_i is the one-ring neighbourhood of v_i, p_j represents the spatial coordinates of vertex v_j and l_{ij} is the length of the edge connecting v_i and v_j. Given a mesh $\mathcal{M}$ and an encoded mesh $\bar{\mathcal{M}}$ with n vertices, the KG error is computed as

$$d_{\mathrm{KG}}(\mathcal{M}, \bar{\mathcal{M}}) = \frac{1}{n} \sum_{i=1}^{n} \big(\|p_i - \bar{p}_i\| + \|s(v_i) - s(\bar{v}_i)\| \big).$$

In order to reduce the smoothness error of a particular vertex v_i, it is necessary to increase its depth and the depth of its one-ring neighbours. Therefore, at each iteration, we select the α% vertices characterized by highest error plus their neighbours, which means that on average we refine 7α% of the vertices. Fig. 8 presents a comparison between the KG-optimized adaptive encoding, the parallelogram predictor [6], and the high-pass quantization technique [19] with 50 anchors points, on a mesh with about 10k vertices, with $\alpha = 5$. The plots show that our technique is better than both of them in terms of KG error. If we have a closer look at the absolute displacement in vertex positions, we can see that even if the compression is optimized for the KG error, we achieve an improvement in terms of RMS error as well. This is no surprise, since, by construction, the error in vertex displacement is bounded by the size of the cells associated to the leaves of the tree. On the other hand, the smoothness term of the KG error shows that, in general, high-pass quantization produces visually better results. The RMS error is computed using the METRO tool [28].

The DAME metric focuses instead on the difference in dihedral angles between the source mesh $\mathcal{M}$ and the decoded mesh $\bar{\mathcal{M}}$. The error is computed as

$$d_{\mathrm{DAME}}(\mathcal{M}, \bar{\mathcal{M}}) = \frac{1}{E} \sum_{e_{ij} \in \mathcal{M}} \|\theta_{ij} - \bar{\theta}_{ij}\| m_{ij} (w_i + w_j),$$

where E is the total number of edges of $\mathcal{M}$, θ_{ij} is the *signed dihedral angle* of the edge e_{ij} connecting v_i and v_j, m_{ij} is a term that penalizes distortion in flat regions, and w_i, w_j are the so-called *visibility weights* associated to v_i and v_j, depending on the position of the vertices on the surface of the mesh. In order to reduce the DAME error, we select the α% of the edges characterized by the highest error in terms of dihedral angle difference, but we also take into account the penalty term and the visibility weight. However, since both the penalty term and the visibility weights are computed only on $\mathcal{M}$, we focus on reducing the difference in dihedral angles of corresponding edges. The computation of the dihedral angle involves the 4 vertices of the faces incident to e_{ij}, so for each selected edge we refine 4 vertices. Each iteration refines 12α% of

Please cite this article in press as: Marras S, et al. Perception-driven adaptive compression of static triangle meshes. Computer-Aided Design (2014), http://dx.doi.org/10.1016/j.cad.2014.08.002

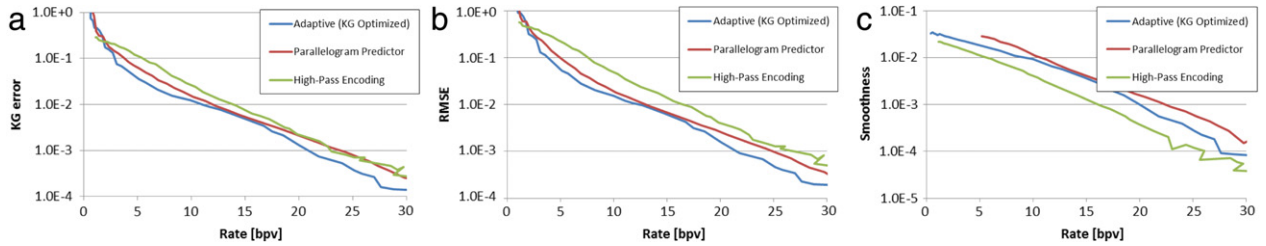

Fig. 8. R–D plots for the adaptive compression of the Hippo dataset (22,260 vertices), optimized for the KG error. Plot (a) shows that the proposed approach effectively outperforms both [6,19]. However, if we focus on the components of the metric, it can be seen that, while our approach produces the best results in terms of objective vertex displacement measure (b), the visual component of the error is between the other methods (c).

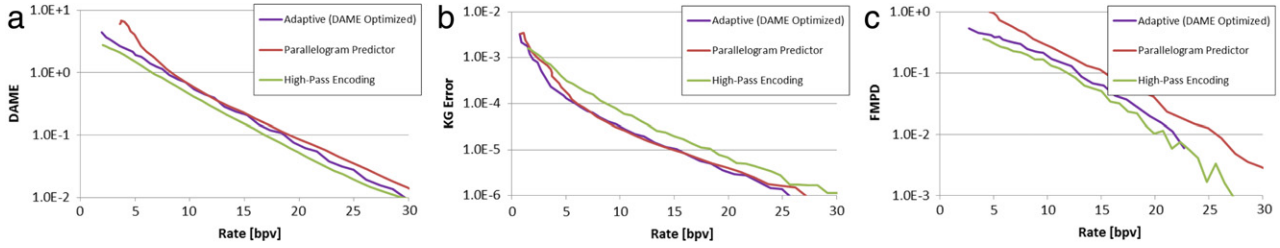

Fig. 9. R–D plots for the adaptive compression of the Dance dataset (7061 vertices), optimized for the DAME metric. Plot (a) shows that our approach fails to overcome [19] in terms of visual quality, but it significantly improves [6]. As a side-effect, the DAME optimization has an effective impact on the KG error (b) and on the FMPD error (c).

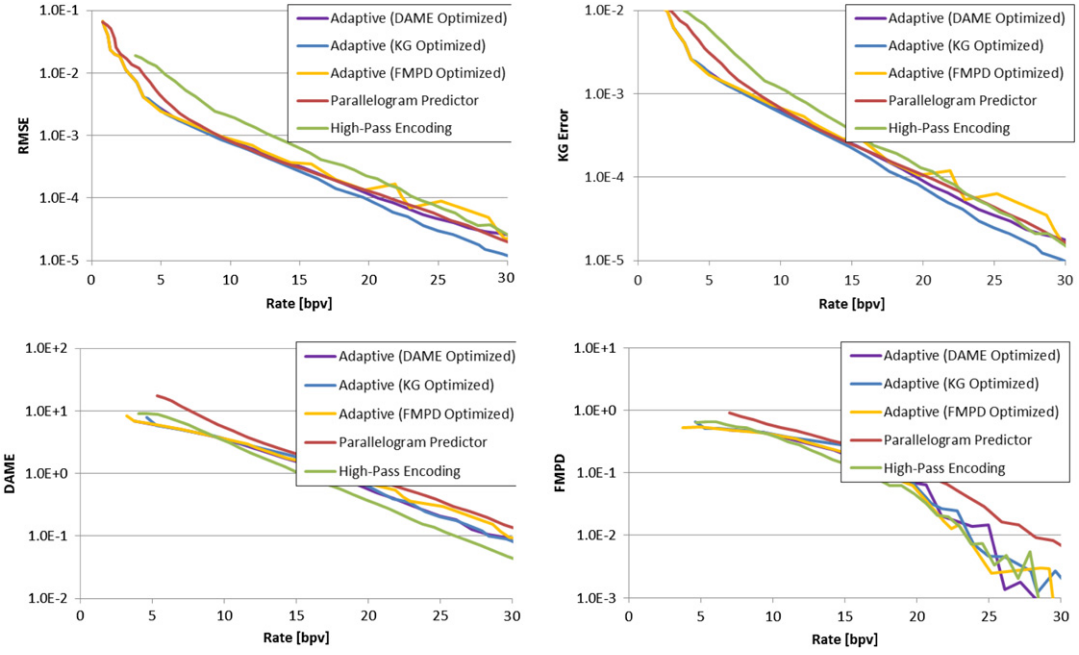

Fig. 10. R–D plots for the Homer dataset (5103 vertices, shown in Fig. 2).

the total number of mesh vertices. In our experiments, α is usually set to 2.5.

Finally, the FMPD metric measures the distance between two meshes as a combination of both local and global roughness error terms. For each vertex of the mesh, the local roughness is defined as

$$L(v_i) = \left| K_i + \frac{\sum_{v_j \in N_i} d_{ij} K_j}{d_{ii}} \right|$$

where K_i is the Gaussian curvature at vertex v_i and d_{ij} is the coefficient of the (i, j) entry of the cotan-Laplacian matrix of the mesh. A *global* roughness term G is computed as the integral of L over the surface of the mesh, and the distance between two meshes is finally computed as

$$d_{\mathrm{FMPD}}(\mathcal{M}, \bar{\mathcal{M}}) = |G - \bar{G}|.$$

Each iteration of the framework requires to refine the central vertex v_i and its one-ring neighbours, thus reducing the local roughness term but also slightly affecting the global roughness. As happens for the KG-driven optimization, we refine on average $7\alpha\%$ of the vertices per iteration, with α usually set to 5.

In general it can be noticed that optimizing for one metric improves the R–D plots also for other metrics, as depicted in Fig. 9. In Figs. 10 and 11 we present a comparison between three different optimizations and the other techniques in terms of both objective and subjective error measures.

To give a better idea of the results of our technique, we present a visual comparison in Fig. 12. If we focus on the visual error only, our

Please cite this article in press as: Marras S, et al. Perception-driven adaptive compression of static triangle meshes. Computer-Aided Design (2014), http://dx.doi.org/10.1016/j.cad.2014.08.002

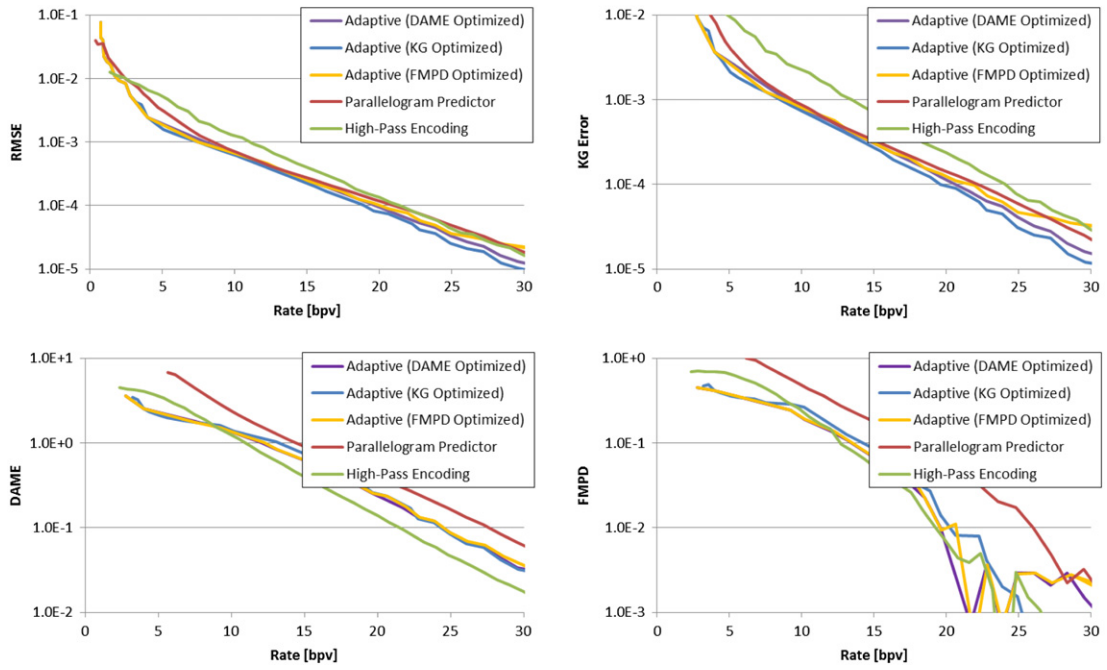

Fig. 11. R–D plots for the Samba dataset (9971 vertices, shown in Fig. 4).

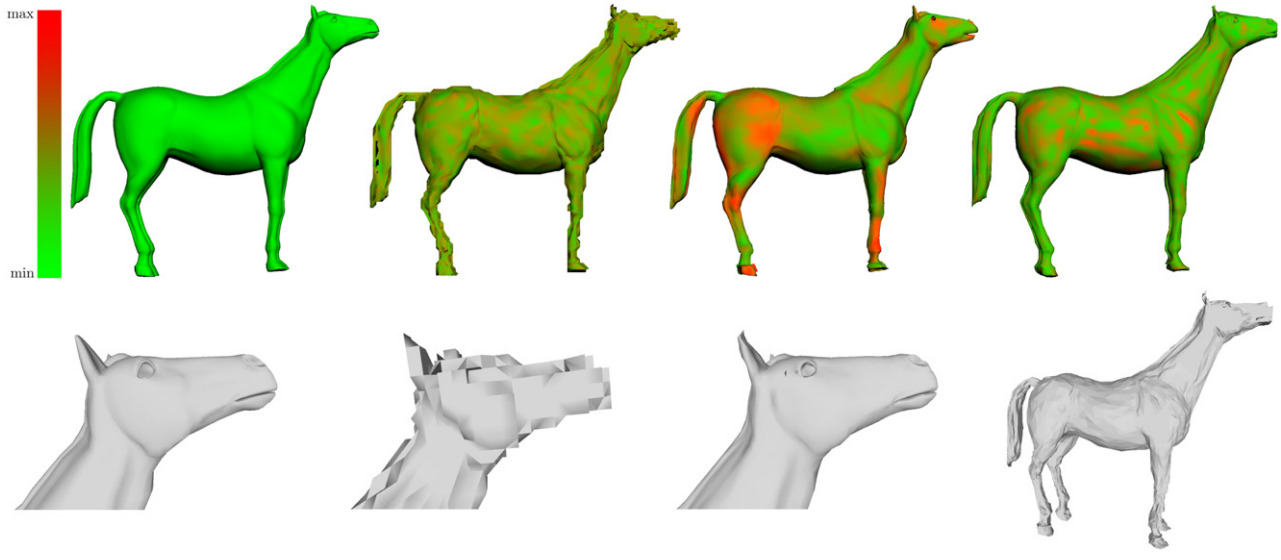

Fig. 12. The source Horse mesh (left) is encoded at a bitrate of 4.5 bits per vertex using different techniques. The mesh compressed using [6] is visibly affected by the uniform quantization (a) and appears to be "blocky". On the other hand, the Laplacian encoding [19] produces visually nicer results (b). Our approach (c) provides even better results as measured by the RMSE, KG and FMPD metrics, while in the DAME metric, the Laplacian encoding is slightly better. Colour-coded is the distance of each vertex to its original position. The close-ups in the second row show the different level of distortion for the aforementioned approaches. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

algorithm significantly improves over the parallelogram predictor rates, but it may be overcome by the high-pass quantization. On the other hand, considering the error in terms of vertex displacement, our technique outperforms the other two approaches. In Fig. 13 we also provide a visual comparison of the same mesh compressed at different compression rates, in order to show the different amount of visual error introduced by the process.

The timings in the encoding process mostly depend not only on the size of the mesh in terms of number of vertices, but also on the complexity of the error evaluation and the number of iterations required to reach the final configuration. Consequently, the encoding is usually slower than common techniques, since the number of iterations may easily reach 20 or 30. On the other hand, the decoding step is much faster: the decoder does a single

Please cite this article in press as: Marras S, et al. Perception-driven adaptive compression of static triangle meshes. Computer-Aided Design (2014), http://dx.doi.org/10.1016/j.cad.2014.08.002

Table 1
Timings for encoding and decoding of several meshes. Timings are expressed in milliseconds. Each mesh is compressed with a KG-error of about 0.0001.

	Vertices	Proposed			Parallelogram		High-pass	
		iters.	enc.	dec.	enc.	dec.	enc.	dec.
Homer	5,103	22	469	7	5	5	11	68
Horse	8,431	22	725	13	9	8	18	122
Samba	9,971	24	1,117	15	11	10	26	160
Hippo	22,260	45	4,249	41	24	24	55	372
James	39,864	12	2,249	82	65	66	131	562
Elephant	42,321	23	4,460	64	44	42	146	603
Armadillo	165,954	23	23,873	362	338	351	992	3,335
Welsh dragon	1,105,352	26	71,674	2431	2044	2083	6227	30,619

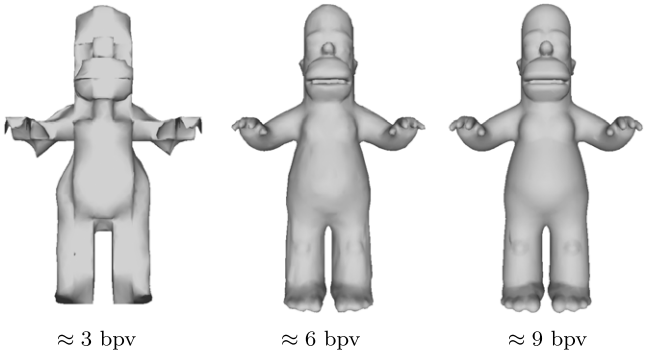

Fig. 13. The Homer dataset, compressed at different bitrates, using the proposed adaptive quantization approach, optimized for the DAME metric. Improvement is significant for low compression rates, while the difference is negligible for higher compression rates.

parallelogram predictor pass to decode both the depth and the bitstreams of the base mesh, then it reconstructs the mesh one vertex at a time. The Laplacian smoothing, however, introduces a small overhead in the decoding process. Table 1 compares the timings on both sides against the performances of other techniques.

5. Conclusions and future work

We presented an algorithm that allows compressing a triangle mesh in a format oriented at progressive transmission, combining aspects of both single-rate and progressive paradigms. It can be easily adapted to work with any kind of metric, achieving considerable improvements for perception-based error metrics. The experiments show that our technique represents a reasonable trade-off between the visual error and the absolute vertex displacement. It outperforms the traditional and still widely used parallelogram predictor in terms of perception-related error, with an improvement of up to 15% in terms of data rate. In some cases, our algorithm is outperformed by the Laplacian encoding of the mesh, yet at the same time it provides other features that make it more attractive:

- The decompression speed of our approach is usually more than 10 times faster, which is an important property for practical applications. Since the time needed for the decompression is linear in number of mesh vertices, while the decompression in Laplacian encoding involves solving a large system of linear equations, it is likely that the performance gap will become even larger for more detailed meshes.
- Thanks to the octree spatial partitioning, our technique provides considerably improved performance in terms of absolute vertex displacements, which is of importance in practical situations where multiple objects interact with each other.
- Our algorithm is able to provide progressive transmission and decoding of a mesh, as demonstrated in Fig. 14.

One of the main limitations of our technique is related to the high cost of the encoding process, especially when compared to other techniques. However, we believe that this should not represent a problem, since the decoding process is comparable to the other state-of-the-art methods and the encoding part of the process can be performed in a batch process on the encoder side.

In the future we plan to extend our technique in order to work with encoding approaches based on differential coordinates. In these techniques it is not possible to locally refine the precision

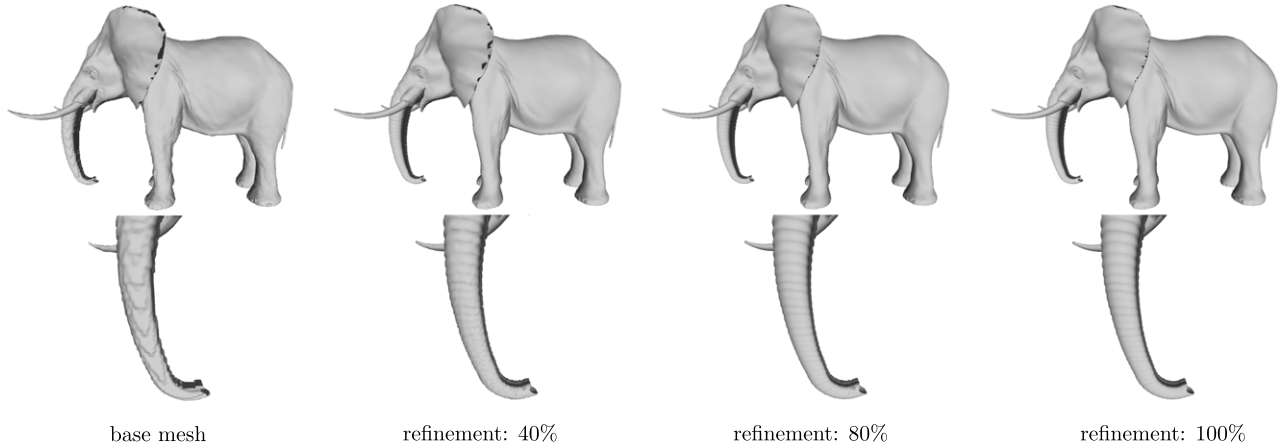

Fig. 14. The proposed approach can also be used for progressive transmission and encoding of a mesh. The base mesh (24 bits per vertex) is encoded, transmitted, and decoded as a single unit, using the parallelogram predictor approach. The mesh, while not as bad as the coarsest version, still exhibits roughness on some parts of the surface. After having transmitted and decoded 40% of the refinement bits, the roughness is significantly reduced. However, artefacts are still visible on some parts of the mesh, such as the trunk (see close-up), the ears, and the tail. If the client requests more bits in order to improve the quality of the shape, we transmit and decode another 40% of the refinement bits. As a result, the distortion on the trunk almost vanishes, with only few vertices, at the end of the trunk, still misplaced. In case we need more precision, we can transmit and decode the remaining bits, getting rid of the distortion and reaching the final configuration of the mesh.

Please cite this article in press as: Marras S, et al. Perception-driven adaptive compression of static triangle meshes. Computer-Aided Design (2014), http://dx.doi.org/10.1016/j.cad.2014.08.002

in vertex reconstruction, due to the nature of the problem. We plan to include the adaptive quantization in the setting by introducing different levels of quantization in the process, aiming at improving the compression rate without significantly affecting the visual error. Other future work aims at reducing the data rate by developing both a depth predictor, exploiting the correlation between vertices, error metric, and octree depth, and a refinement-bits-predictor, since the arithmetic encoding does not usually bring any significant gain.

Acknowledgements

This work was supported by the SNF under project number 200020-146764. The authors thank Kai Wang for kindly providing the tool for the computation of the Fast Mesh Perceptual Difference.

References

[1] Gotsman C. On the optimality of valence-based connectivity coding. Comput Graph Forum 2003;22(1):99–102.
[2] Lavoué G, Gelasca ED, Dupont F, Baskurt A, Ebrahimi T. Perceptually driven 3D distance metrics with application to watermarking. In: Tescher AG, editor. Applications of digital image processing XXIX. Proceedings of SPIE, vol. 6312. 2006. p. 63120L.
[3] Lavoué G. A multiscale metric for 3D mesh visual quality assessment. Comput Graph Forum 2011;30(5):1427–37.
[4] Váša L, Rus J. Dihedral angle mesh error: a fast perception correlated distortion measure for fixed connectivity triangle meshes. Comput Graph Forum 2012; 31(5):1715–24.
[5] Taubin G, Rossignac J. Geometric compression through topological surgery. ACM Trans Graph 1998;17(2):84–115.
[6] Touma C, Gotsman C. Triangle mesh compression. In: Proceedings of Graphics Interface. 1998. p. 26–34.
[7] Kälberer F, Polthier K, Reitebuch U, Wardetzky M. Free-Lence—coding with free valences. Comput Graph Forum 2005;24(3):469–78.
[8] Alliez P, Desbrun M. Valence-driven connectivity encoding for 3D meshes. Comput Graph Forum 2001;20(3):480–9.
[9] Rossignac J. Edgebreaker: connectivity compression for triangle meshes. IEEE Trans Vis Comput Graphics 1999;5(1):47–61.
[10] Gumhold S, Straßer W. Real time compression of triangle mesh connectivity. In: Proceedings of SIGGRAPH. 1998. p. 133–40.
[11] Mamou K, Zaharia T, Prêteux F. TFAN: a low complexity 3D mesh compression algorithm. Comput Anim Virtual Worlds 2009;20(2–3):343–54.
[12] Hoppe H. Progressive meshes. In: Proceedings of SIGGRAPH. 1996. p. 99–108.
[13] Sim J-Y, Kim C-S, Lee S-U. An efficient 3D mesh compression technique based on triangle fan structure. Signal Process Image 2003;18(1):17–32.
[14] Courbet C, Hudelot C. Taylor prediction for mesh geometry compression. Comput Graph Forum 2010;30(1):139–51.
[15] Váša L, Brunnett G. Exploiting connectivity to improve the tangential part of geometry prediction. IEEE Trans Vis Comput Graphics 2013;19(9):1467–75.
[16] Peng J, Kuo C-CJ. Geometry-guided progressive lossless 3D mesh coding with octree (OT) decomposition. ACM Trans Graph 2005;24(3):609–16.
[17] Payan F, Antonini M. Wavelet-based compression of 3D mesh sequences. In: Proceedings of the Second International Conference on Machine Intelligence, Tozeur, Tunisia. 2005.
[18] Karni Z, Gotsman C. Spectral compression of mesh geometry. In: Proceedings of SIGGRAPH. 2000. p. 279–86.
[19] Sorkine O, Cohen-Or D, Toledo S. High-pass quantization for mesh encoding. In: Proceedings of SGP. 2003. p. 42–51.
[20] Bayazit U, Konur U, Ates HF. 3-D mesh geometry compression with set partitioning in the spectral domain. IEEE Trans Circuits Syst Video Technol 2010;20(2):179–88.
[21] Maglo A, Courbet C, Alliez P, Hudelot C. Progressive compression of manifold polygon meshes. Comput Graph 2012;36(5):349–59.
[22] Lee D-Y, Sull S, Kim C-S. Progressive 3D mesh compression using MOG-based Bayesian entropy coding and gradual prediction. Vis Comput 2013;1–15.
[23] Lavoué G. A local roughness measure for 3D meshes and its application to visual masking. ACM Trans Appl Percept 2009;5(4):1–23. Article 21.
[24] Corsini M, Larabi M-C, Lavoué G, Petrik O, Vasa L, Wang K. Perceptual metrics for static and dynamic triangle meshes. Comput Graph Forum 2013;32(1): 101–25.
[25] Wang K, Torkhani F, Montanvert A. A fast roughness-based approach to the assessment of 3D mesh visual quality. Comput Graph 2012;36(7):808–18.
[26] Marpe D, Wiegand T, Schwarz H. Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard. IEEE Trans Circuits Syst Video Technol 2003;13(7):620–36.
[27] Vlasic D, Baran I, Matusik W, Popović J. Articulated mesh animation from multi-view silhouettes. ACM Trans Graph 2008;27(3):1–9. Article 97.
[28] Cignoni P, Rocchini C, Scopigno R. Metro: measuring error on simplified surfaces. Comput Graph Forum 1998;17(2):167–74.

Please cite this article in press as: Marras S, et al. Perception-driven adaptive compression of static triangle meshes. Computer-Aided Design (2014), http://dx.doi.org/10.1016/j.cad.2014.08.002