A Predictor for Triangle Mesh Compression Working in Tangent Space

Petr Vaněček[a], Filip Hácha[b] and Libor Váša[c]
Department of Computer Science and Engineering, Faculty of Applied Sciences, University of West Bohemia, Univerzitní 8, 301 00 Plzeň, Czech Republic

Keywords: Geometry Compression, Parallelogram Prediction, Tangent Space.

Abstract: Triangle mesh compression has been a popular research topic for decades. Since a plethora of algorithms has been presented, it is becoming increasingly difficult to come up with significant performance improvements. Some of the recent advances in compression efficiency come at the cost of rather steep implementation and/or computational expense, which has profound consequences on their practicality. Ultimately it becomes increasingly difficult to come up with improvements that are reasonably easy to implement and do not harm the computational efficiency of the compression/decompression procedure. In this paper, we analyze a combination of two previously known techniques, namely using the local coordinates for expressing compression residuals and weighted parallelogram prediction, which were not previously investigated together. We report that such approach outperforms industry standard Draco on a large set of test meshes in terms of rate/distortion ratio, while retaining beneficial properties such as simplicity and computational efficiency.

1 INTRODUCTION

Triangle mesh compression is a common task in mesh processing pipelines, and even the ever continuing growth of commonly available computational power has not diminished its importance. Its is essential for both efficient storage and transfer of highly detailed meshes, since even the most basic approaches allow for a quite drastic reduction of code length in comparison with plain encoding.

The first generation of compression algorithms has focused on reducing mechanistic error measures (MSE, PSNR, Hausdorff distance) while achieving a low data rate. Later, the field has seen a renewed interest when perceptual metrics (DAME, MSDM, FMPD) came into focus and perceptual fidelity became the main goal (Sorkine et al., 2003; Marras et al., 2015). Now, after decades of research, a wide variety of algorithms is available, each bringing a certain improvement in compression efficiency. It is therefore quite difficult to come up with an algorithm that outperforms the state-of-the-art significantly.

On the other hand, many of the advanced techniques seem to lack in terms of real-life applicability. Compression is mostly used to save time for users, trading the slow data transmission time for fast processing (decompression). This purpose is often neglected in research papers, where only the compression efficiency is investigated, while the computational cost of advanced algorithms is often downplayed. Similarly, the ease of implementation is often neglected, while in real-world applications, a small compression efficiency gain can often be neglected when facing a high implementation cost, since in contrast with academic-level proof-of-concept testing, real-world implementations have much higher requirements on robustness and reliability. This is, for example, demonstrated by the current de-facto standard for industrial mesh compression, the Google Draco library: out of the plethora of advanced compression techniques, a rather modest subset of basic, most efficient algorithms is chosen and implemented at the industrial robustness level.

One of the evergreen approaches is the combination of traversal-based connectivity encoding (Edgebreaker (Rossignac, 1999) or valence-based coding (Alliez and Desbrun, 2001)) with the parallelogram prediction rule used for encoding the vertex positions. This approach has been used as a baseline for several advancements, such as weighted parallelogram coding, double/multiple parallelogram coding, angle-based coding, encoding of residuals in local coordinates and many others. Without exhaustive test-

[a] https://orcid.org/0000-0002-1858-2411
[b] https://orcid.org/0000-0001-8956-6411
[c] https://orcid.org/0000-0002-0213-3769

Vaněček, P., Hácha, F. and Váša, L.
A Predictor for Triangle Mesh Compression Working in Tangent Space.
DOI: 10.5220/0012352900003660

In *Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2024) - Volume 1: GRAPP, HUCAPP and IVAPP*, pages 236-243
ISBN: 978-989-758-679-8; ISSN: 2184-4321

ing, it is hard to tell which of these advancements "eat the same piece of the pie", making each other less efficient when combined, and which can be used in conjunction, combining their advantages. This is especially true when considering the performance in terms of perceptual metrics.

In these circumstances, the contributions of the presented paper are the following:

- we demonstrate, that a combination of weighted parallelogram prediction with encoding in local coordinates (tangential + normal) leads to a combined benefit in terms of rate-distortion performance,
- we show that non-uniform quantization of local coordinates leads to a considerable improvement in rate-distortion performance in terms of perceptual error metrics,
- we provide comparison with state-of-the-art industrial compression library, showing that the combined approach, in spite of its simplicity, provides a performance advantage at negligible additional cost in both implementation effort and execution computational expense.

2 RELATED WORK

In terms of single-rate triangle mesh compression, the most prominent approaches are driven by connectivity encoding algorithms (e.g., *Edgebreaker* (Rossignac, 1999), *Valence coding* (Touma and Gotsman, 1998) or *TFAN* (Mamou et al., 2009)). Such algorithms process a mesh during a connectivity traversal, which these approaches use to exploit the spatial coherence of vertex positions by prediction inferred from already encoded vertices. Their popularity mainly stems from their simplicity.

While not the first, certainly the most influential is the *Parallelogram scheme* proposed by Touma and Gotsman (Touma and Gotsman, 1998). It forms a planar parallelogram from vertex positions of an adjacent triangle to predict the position of the currently coded vertex. Not only are most of the modern connectivity-driven methods directly derived from it, but it is also still being used in modern mesh compression software (e.g., *Google Draco* (Galligan et al., 2018)).

The data rate of geometry can be improved by averaging over multiple predictions where possible. *Dual parallelogram* scheme proposed by Sim et al. (Sim et al., 2003) uses a connectivity traversal which quite often allows prediction of a single vertex by two parallelograms. The *FreeLence* method proposed by Kälberer et al. (Kälberer et al., 2005) added an additional parallelogram prediction computed from three incident vertices on the boundary of the already processed area.

Another way of improvement is the adjustment of the prediction itself. On curved surfaces, the planar prediction might be inefficient in terms of the normal direction. To this end, Gumhold and Amjoun (Gumhold and Amjoun, 2003) proposed to fit a higher-order surface to already encoded geometry to estimate a dihedral angle. Ahn et al. (Ahn et al., 2006) also predicted the dihedral angles but from neighbouring triangles. By introducing weighting in the parallelogram formula, some approaches were able to improve the tangential part of the prediction. This was discussed by Kälberer et al. (Kälberer et al., 2005) and elaborated on by Courbet and Hudelot (Courbet and Hudelot, 2011) who used Taylor expansion to determine weights even for more complicated stencils than parallelograms. More recently, Váša and Brunnett (Váša and Brunnett, 2013) estimated the weights from vertex valences and already-known inner angles to obtain state-of-the-art performance in predicting tangential information.

The quantization and encoding of correction vectors is also a crucial part of the problem. Most of the methods (e.g., (Touma and Gotsman, 1998; Váša and Brunnett, 2013)) quantize the encoded values in a global coordinate system, each coordinate quantized with equal precision. As was pointed out by Lee et al. (Lee et al., 2002), this, however, leads to high normal distortion particularly visible on planar surfaces not aligned with one of the main coordinate axes. Their proposal was to work in a local frame either represented by two inner and one dihedral angle, or by axes aligned with an adjacent triangle. Both these representations enable the separation of normal and tangent information, thus exploiting different distributions of values in entropy coding. Gumhold and Amjoun (Gumhold and Amjoun, 2003) combined these two representations and used a dihedral angle for normal information, while tangent information was represented by planar coordinates. Kälberer et al. (Kälberer et al., 2005) improved this approach by quantizing the normal and tangent information with different precision.

All the listed methods aimed to optimize mechanistic distortion measures (e.g., MSE, Hausdorff distance or PSNR), which give an upper bound in the error of absolute coordinates. These metrics, however, do not reflect the way the human brain perceives distortion. This is better measured by perception-oriented metrics (e.g., *MSDM2* (Lavoué, 2011), *DAME* (Váša and Rus, 2012), *FMPD* (Wang et al., 2012), *TPDM* (Torkhani et al., 2014) and

TPDMSP (Feng et al., 2018)). In the past, there were multiple methods that optimized such criteria (e.g., (Karni and Gotsman, 2000; Sorkine et al., 2003; Marras et al., 2015)). These, however, are quite complex and are usually outperformed by conventional methods in terms of mechanistic measures. More recently, there have been proposed a few methods that aim at both criteria (Alexa and Kyprianidis, 2015; Lobaz and Váša, 2014; Váša and Dvořák, 2018), but require additional complexity.

3 PRELIMINARIES

The problem that we address in this paper is the following: a triangle mesh is given, consisting of a *connectivity* $\mathcal{C} = \{\mathbf{t}_i\}_{i=0}^{T-1}$, and *geometry* $\mathcal{G} = \{\mathbf{v}_i\}_{i=0}^{V-1}$. Each $\mathbf{t}_i$ is a triplet of integers, representing the indices of the vertices that form the i-th triangle, and each $\mathbf{v}_i$ is a triplet of floats (v_i^x, v_i^y, v_i^z), representing the Cartesian coordinates of the i-th vertex. The objective is to store this information into a binary stream of the shortest possible length so that reconstruction can be built from the stream with the following properties:

1. the connectivity is reconstructed without loss, up to an allowed reordering of triangles and reindexation of vertices, and
2. the geometry is reconstructed with a certain user-controllable precision.

Many algorithms only work when additional conditions are met. In particular, a *connectivity-manifold* is a mesh, where each edge is part of at most two triangles (mesh border is allowed). In the following, we assume that the input triangle mesh fulfills this property. In order to put our contribution into context, we shortly review the main building blocks of a typical single-rate triangle mesh encoder. Next, we are also going to discuss some extensions proposed previously.

The most common approach to static mesh compression is to process the surface in a traversal driven by the connectivity. In each traversal step, a single triangle is attached to the part of the mesh that has been already processed. The edge of the known part of the mesh is identified as *gate*, and the vertex opposite of the gate is either already part of the processed part of the mesh, in which case it is the task of the connectivity encoder to identify which one it is, or it has not been processed yet: in that case, the vertex position is encoded into the data stream in some efficient way.

The authors of the popular connectivity encoder EdgeBreaker have shown, that for connectivity-manifold meshes, it suffices to store one of five symbols C, L, E, R or S per triangle, where the C symbol represents encountering a previously unknown vertex, while the remaining symbols encode which known vertex is the tip vertex of the current triangle. After each expansion step, the next gate is selected using certain implicit rules, i.e. without the need for storing any additional data.

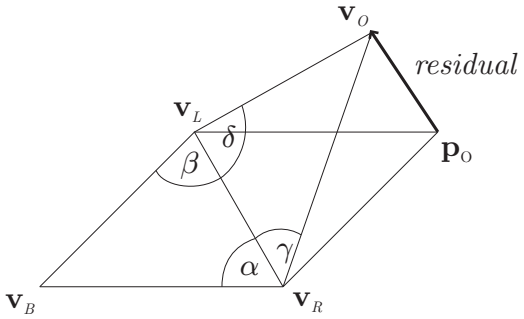


Figure 1: Prediction stencil.

A typical situation that arises when a previously unvisited vertex is encountered during mesh traversal is depicted in Fig. 1. The local configuration is also known as the *prediction stencil*. The objective is to store information that allows reconstructing the coordinates of the opposite vertex $\mathbf{v}_R$ with as few bits as possible. A common prediction-correction strategy is usually applied: based on the known information, the decoder builds a prediction of the following encoded value(s); the encoder mimics the prediction and encodes the difference between the actual value(s) and the prediction, usually quantized to some precision. If the predictions are accurate, then the magnitudes of the corrections are generally small in comparison with original data, leading to a decrease in the entropy of the data stream and ultimately to efficient compression.

A triangle adjacent to the gate in the processed part of the mesh (*base triangle*) consists of vertices $\mathbf{v}_L$, $\mathbf{v}_R$ and $\mathbf{v}_B$. The coordinates of these vertices are known to both the encoder and the decoder (up to quantization error), and therefore the decoder can use them to estimate the position of the opposite vertex $\mathbf{v}_O$. A popular choice of prediction scheme is the parallelogram predictor, formulated as

$$\mathbf{p}_O = \bar{\mathbf{v}}_L + \bar{\mathbf{v}}_R - \bar{\mathbf{v}}_B, \tag{1}$$

where $\mathbf{p}_O$ is the prediction and $\bar{\mathbf{v}}$ stands for the coordinates of a vertex $\mathbf{v}$ as known by the decoder, i.e. possibly affected by quantization.

This basic scheme has been modified in different ways, usually trying to incorporate additional information available at the decoder in order to improve the accuracy of the prediction. In particular, when a new vertex is encountered, it can be predicted from

two (Sim et al., 2003) or multiple (Cohen-Or et al., 2002) gates. The predictions are usually averaged and for practical meshes, this results in an improved performance.

Alternatively, it is possible to transmit the mesh connectivity first, processing the mesh geometry in a separate subsequent traversal. This way, the information from the connectivity, in particular vertex valences, can be used for the prediction as well. With the valences known, it is possible to estimate the inner angles in the currently processed triangle as $2\pi/d_i$, where d_i is the valence of the i-th vertex. The authors of the weighted parallelogram scheme have derived that after normalising the inner angles to sum up to π, so that they form a proper triangle, it is possible to predict the position of the vertex $\mathbf{v}$ using a generalised parallelogram predictor

$$\mathbf{p}_O = w_1\bar{\mathbf{v}}_L + w_2\bar{\mathbf{v}}_R - (1-w_1-w_2)\bar{\mathbf{v}}_B, \qquad (2)$$

where the weights w_1 and w_2 are computed as

$$w_1 = \frac{cot(\beta)+cot(\delta)}{\cot(\delta)+\cot(\gamma)}, \qquad (3)$$

$$w_2 = \frac{cot(\alpha)+cot(\gamma)}{\cot(\delta)+\cot(\gamma)}, \qquad (4)$$

where the angles are the known inner angles of the base triangle α and β, and the predicted inner angle of the newly attached triangle γ and δ, as shown in Fig. 1. By incorporating the inner angles around the vertices $\mathbf{v}_L$ and $\mathbf{v}_R$ in other triangles known to the decoder, it is possible to build an even more accurate prediction of the angles γ and δ, leading to further improvement of the data rate.

Another modification to the basic algorithm has been proposed by the authors of the Angle-Analyzer algorithm(Lee et al., 2002). They proposed two independent ideas: first, rather than representing the encoded vertex in terms of Cartesian coordinates (or their corrections), they represent it in terms of inner angles of the new triangle and a dihedral angle between the known adjacent triangle and the newly attached triangle. Second, they propose separating the normal and tangential components of the correction in separate contexts of the entropy coder. This approach can be manifested as either

- encoding the inner angles in a single context and dihedral angles in another, when encoding the vertex position in terms of angles, or
- separating the tangential and normal components of the correction and encoding each in separate contexts, when encoding the vertex position in terms of coordinates.

In the latter case, the authors use a local coordinate frame as follows: $\mathbf{v}_L$ is used as the origin, and the following orthonormal vectors are used as the basis:

$$\begin{aligned}\mathbf{x}_1 &= \frac{(\mathbf{v}_L-\mathbf{v}_R)\times(\mathbf{v}_B-\mathbf{v}_R)}{\|(\mathbf{v}_L-\mathbf{v}_R)\times(\mathbf{v}_B-\mathbf{v}_R)\|}\\ \mathbf{x}_2 &= \frac{\mathbf{v}_L-\mathbf{v}_R}{\|\mathbf{v}_L-\mathbf{v}_R\|}\\ \mathbf{x}_3 &= \mathbf{x}_1\times\mathbf{x}_2\end{aligned} \qquad (5)$$

The coordinates of the opposite vertex $\mathbf{v}_O$ are expressed in this coordinate system and encoded. This way the first coordinate of the encoded vector represents the normal component, while the remaining ones represent the tangential component. The main benefit of this approach is that the statistical distribution of the normal component is different from that of the tangential components, which is in turn exploited by the used entropy coder.

Another approach to encoding vertices in local coordinates has been taken by the authors of the Higher Order Prediction scheme (Gumhold and Amjoun, 2003). They use a cylindrical coordinate system with the origin at the center of the gate and axis-aligned with the gate. A vertex in this coordinate system is described by a triplet of scalars (x,y,α), and reconstructed as

$$\bar{\mathbf{v}_O} = (\bar{\mathbf{v}_L}+\bar{\mathbf{v}_L})/2 + x\mathbf{x}_2 + sin(\alpha)y\mathbf{x}_1 + cos(\alpha)y\mathbf{x}_3, \qquad (6)$$

where $(\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3)$ represent the same local basis orientation as in Eq. (5). The local coordinates x and y can be predicted using the parallelogram rule, while the angle α can be estimated using the higher order fitting of already known vertices, at a non-negligible computational cost. Note that in order to compensate for different sampling rates at varying distances from the coordinate system axis, rather than quantizing the angle α directly, the quantity $y\alpha$ is quantized end encoded. In decoding, y is decoded first and used to decode the actual value of α.

These approaches to encoding vertex positions using local coordinates suffer from serious flaws. The approach of Lee et al. (Lee et al., 2002) uses the left gate vertex as origin. This way the information from known positions of the right gate vertex and base vertex is essentially ignored, except for gate orientation and base triangle normal, which influence the spatial alignment of the basis. The approach by Gumhold and Amjoun (Gumhold and Amjoun, 2003) on the other hand uses a non-orthogonal, non-uniform cylindrical grid, which leads to a hard-to-predict behaviour of the quantization procedure. In the following, we propose a simple way to avoid both these problems.

4 PROPOSED ALGORITHM

We propose using the common connectivity-driven traversal based approach, with corrections encoded in a local orthonormal basis. The key insight is that origin of the basis can be put to the predicted position of the tip vertex $\mathbf{p}_O$. In terms of resulting symbol entropy, this approach is equivalent to estimating the local coordinates using some prediction scheme. In contrast with the approach of Gumhold and Amjoun (Gumhold and Amjoun, 2003), we use the weighted parallelogram scheme for prediction.

Another design choice is the section of the basis orientation. We have experimented with two possibilities. One is equivalent to the basis described by Eq. 5, i.e. aligned with the gate edge. However, a different basis with similar properties can be constructed as follows:

$$\begin{aligned}\mathbf{x'}_1 &= \frac{(\mathbf{v}_L - \mathbf{v}_R) \times (\mathbf{v}_B - \mathbf{v}_R)}{\|(\mathbf{v}_L - \mathbf{v}_R) \times (\mathbf{v}_B - \mathbf{v}_R)\|} \\ \mathbf{x'}_2 &= \frac{\mathbf{p}_O - \mathbf{v}_B}{\|\mathbf{p}_O - \mathbf{v}_B\|} \\ \mathbf{x'}_3 &= \mathbf{x}_1 \times \mathbf{x}_2 \end{aligned} \tag{7}$$

Rather than aligning the basis with the gate orientation, this basis is aligned with the direction towards the prediction from the known vertex $\mathbf{v}_B$. Note that, for a symmetric prediction stencil, the bases are equivalent, and in both cases the bases separate the normal information into the first component of the local coordinate triplet.

Separating the normal has two profound consequences: first, the normal component usually has a significantly different statistical distribution than the tangential components. While the tangential components represent the sampling of the surface, which may or may not be regular, the normal component essentially carries the information about the shape.

This then implies the second consequence: having the sampling and shape information separated, it is possible to use a different precision in encoding each. While for textured meshes, even the tangential position of vertices may be important, from the point of view of shape the sampling is mostly irrelevant, and therefore it may be beneficial to sample the normal component more precisely.

5 EXPERIMENTS

As mentioned above, the distortion of normals plays a significant role in perceiving the quality of a compressed triangle mesh. Figure 2 shows the comparison of mesh distortion while maintaining the same compression ratio using the prediction corrections in world coordinates and in coordinates aligned with the tangential plane.

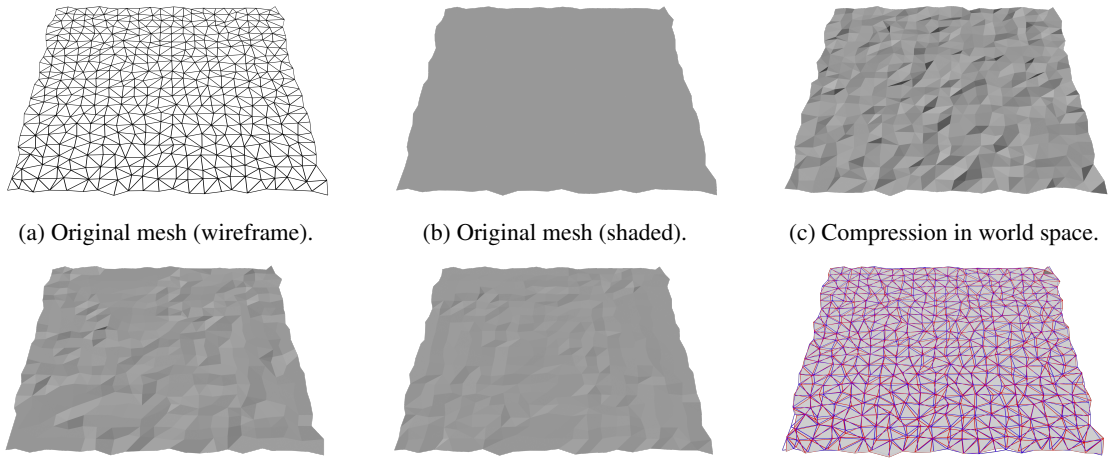
We have tested the compression distortion of a planar surface in various orientations. The traditional world space prediction space is very sensitive to the orientation of the surface. This is significantly reduced when using the tangent space. While targeting a human perception metric (high sensitivity to changes in dihedral angles close to flat), it is even possible to obtain better results by increasing the number of quantization levels for the normal direction, while reducing the number of quantization levels in tangent directions to maintain the same compression ratio.

To compare the proposed method, we have selected three existing algorithms. Draco (Galligan et al., 2018) is a well-known and commonly used library for mesh compression. In our tests, we use the default settings. Error propagation control in laplacian mesh compression (EPC) (Váša and Dvořák, 2018) is a compression method that focuses on achieving good results in both perceptual and mechanistic quality of the compressed meshes. Finally, we compare the results against the Weighted parallelogram prediction (WP) (Váša and Brunnett, 2013) in world space.

To measure the quality of compressed models, we have chosen three different metrics:

1. *Mean squared error* (MSE) is a widely used metric, unfortunately, this metric only measures the distances between the original and distorted vertices. Similarly to other areas, this metric is not very suitable for measuring the quality of compression with respect to shape preservation or visual quality, yet, it is used very often.
2. *Metro* (Cignoni et al., 1998) metric is another commonly used mechanistic error measure. It builds on analyzing the nearest point pairs on the two compared meshes, and it does not require the compared meshes to have the same connectivity.
3. *Dihedral angle mesh error* (Váša and Rus, 2012) (DAME) is one of the metrics designed to take human perception into account, which we have used, since comparing artifacts introduced by compression, it is important to measure, how these artifacts are perceived by humans.

For our tests we have used models from the thingi10k archive (Zhou and Jacobson, 2016). In order to be able to compare all methods at the same BPV, we use a subset fulfilling these criteria: (a) has more than 20.000 vertices, (b) has a single component, (c) is 2-manifold and (d) is watertight. Out of the available models, over 500 meet these criteria.



(a) Original mesh (wireframe). (b) Original mesh (shaded). (c) Compression in world space.

(d) Compression in tangent space. (e) Finer quantization in normal direction. (f) Precise encoding of the first triangle.

Figure 2: The comparison of quantization in world space and tangential space. The original model is a plane rotated 17.5° around the x axis - (a) the wireframe and (b) shaded model. Compressed versions have a compression rate of approx. 13 BPV. (c) The compression in world space coordinates is very sensitive to rotation. (d) In tangent space, the errors are mainly introduced by the precision of the first triangle, but generally it is less sensitive to rotation than compression in world space. (e) It is even possible to increase the precision in the normal direction, while reducing the precision in target directions while maintaining the same compression rate. (f) In this particular scenario, when all triangles have the same normal, the normal component should be zero. Due to the distortion of the first triangle, this is not generally true. Storing the first triangle (dark triangle in top right corner) without distortion allows a significant increase of the compression rate (8.6 BPV), while maintaining constant normal vectors. The triangles of the original mesh are blue, and the compressed mesh is red.

The proposed method based on weighted parallelogram is presented in two variants - the standard tangent space compression and compression with double precision of the normal component.

We have tested two formulations of local basis. The choices are the basis aligned with the prediction gate, as defined in Eq. 5, and the basis aligned with the direction towards tip vertex, as defined in Eq. 7. In the experiments, the basis of Eq. 7 provides slightly better results on average, but slightly worse in the median. For the rest of our experiments, we have used this basis, but choosing the better of these two options for a particular model can in some cases lead to a considerable reduction of the resulting distortion of up to 45% in terms of MSE and up to 22% in terms of Metro and DAME. Note, however, that for a majority of models, the difference tends to be rather small.

Note that Draco does not allow fine control over the data rate (measured in bits per vertex - BPV), but only allows choosing an integer-valued quality control parameter. For this reason, we use Draco as the baseline method and we compare the other methods relative to Draco, since the other competing method allow fine tuning the resulting data rate and matching the Draco result in terms of achieved data rate. Therefore, in the figures, we present a quantity that relates a particular error measure achieved by a particular algorithm to the same error measure achieved by Draco at the same data rate. A value of 1 indicates equivalent result, while a values smaller than 1 indicate a performance gain w.r.t. Draco (lower distortion at the same data rate), and values larger than 1 indicate a performance loss.

In terms of MSE, our method outperforms Draco and EPC in most cases (Figure 3). On the other hand, in the given dataset, there appears a number of models where it performs rather poorly. One such example is depicted in Figure 4. Its MSE is 151× higher than for Draco, maintaining the same BPV (these outliers are not in the plot). Due to these outliers, there is a large difference between median and average cases. A common feature of these outliers are usually a rather low data rate for Draco standard settings (5 - 6 BPV) and a high variability of triangle sizes. Using Metro metric, the proposed algorithm is slightly better than Draco when using uniform quantization in all directions. It outperforms Draco when halving the size of quantization bins (doubling the number of levels) for normal direction. For most of the models, the proposed method performs better in DAME metric than Draco. Similarly to MSE, the outliers are pushing the average much higher than the median case.

The increasing computational power allows the processing of huge triangle meshes. The compression process itself, although not time-critical, should not exceed a user-acceptable limit, both in compres-

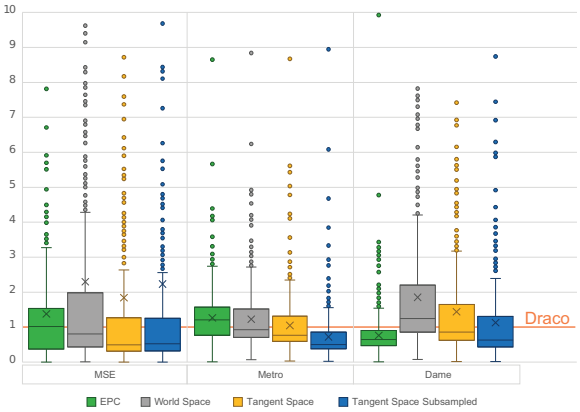


Figure 3: Comparison of compression methods relative to Draco using MSE, Metro and Dame.

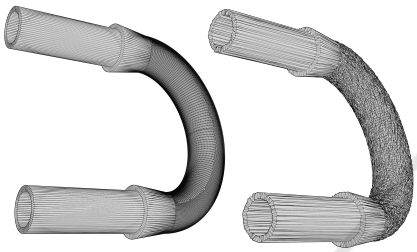


Figure 4: Original model (left) and model compressed by proposed method (right) on 5.77 BPV. The MSE is 151 times higher than the Draco compression at comparable BPV. The model consists of triangles with extremely varying sizes and shapes, from large narrow triangles to relatively small, nearly equilateral triangles.

sion and especially in decompression phase. We have measured the compression and decompression time for all methods. Unfortunately, it is not possible to measure purely the processing time, as the process is usually combined with I/O operations. For this reason, we have measured the overall time of the process, including the reading and parsing of the input file and writing the result to the disk, targeting at the same BPV. For the measurement, we have used a commodity hardware - a notebook with AMD Ryzen 7 5800H, 16 GB of RAM, and an SSD hard drive. Our method is implemented in .NET 6 as a single-threaded application, without any special optimization.

The compression processes for Draco and our implementation exhibit a linear behavior in terms of number of vertices, while EPC is less predictable (Figure 5), probably since the process involves solving a system of linear equations. Due to the simplicity of the compression algorithm, the weighted parallel prediction and our proposed modifications are much faster than Draco, even in the non-optimized developer version. There is also only a negligible impact of the proposed modification on the overall performance

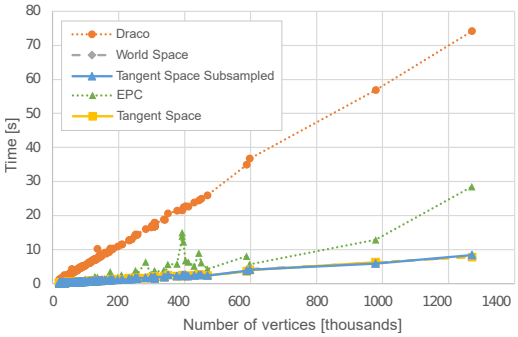


Figure 5: Comparison of compression time.

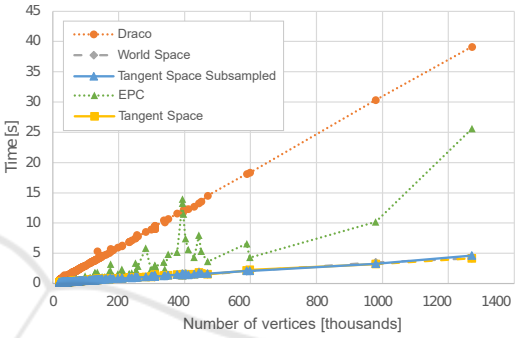


Figure 6: Decomparison of compression time.

caused by the transformation of the coordinates into the tangent space. It is also worth noting that Draco compression exceeds 10 seconds for models with approximately 200,000 vertices.

While the compression process usually affects only a limited group of users - the creators, decompression affects the user experience of the end users (Figure 6). The decompression process is faster than the compression process for all methods. While Draco and Parallelogram-based method nearly halved the time, in the case of EPC, the difference is much smaller. Similarly to the compression process, our implementation is more than 8 times faster than Draco.

6 CONCLUSION

We proposed a modification of the weighted parallelogram compression method based on the transformation of vertex coordinates into tangential space. The transformation considerably improves the quality of compression in terms of objective and subjective metrics, while maintaining a simple algorithm. According to our tests, this modification outperforms in terms of rate-distortion ratio some well-known compression methods , such as Draco and EPC, for most meshes, while being considerably faster.

There are, however, some triangle configurations that are not suitable for the proposed algorithm. In the future, we plan to investigate the particular properties that cause the drop of compression performance of our algorithm in these cases, and the possible strategies to mitigate this performance loss.

ACKNOWLEDGEMENTS

This work was supported by the project 20-02154S of the Czech Science Foundation. Filip Hácha was partially supported by the University specific research project SGS-2022-015, New Methods for Medical, Spatial and Communication Data.

REFERENCES

Ahn, J.-H., Kim, C.-S., and Ho, Y.-S. (2006). Predictive compression of geometry, color and normal data of 3-d mesh models. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(2):291–299.

Alexa, M. and Kyprianidis, J. E. (2015). Error diffusion on meshes. *Computers & Graphics*, 46:336–344. Shape Modeling International 2014.

Alliez, P. and Desbrun, M. (2001). Valence-Driven Connectivity Encoding for 3D Meshes. *Computer Graphics Forum*.

Cignoni, P., Rocchini, C., and Scopigno, R. (1998). Metro: Measuring error on simplified surfaces. *Computer Graphics Forum*, 17:167 – 174.

Cohen-Or, D., Cohen, R., and Irony, R. (2002). Multi-way geometry encoding. Tech. Rep., Tel Aviv University.

Courbet, C. and Hudelot, C. (2011). Taylor prediction for mesh geometry compression. *Computer Graphics Forum*, 30(1):139–151.

Feng, X., Wan, W., Xu, R. Y. D., Chen, H., Li, P., and Sánchez, J. A. (2018). A perceptual quality metric for 3d triangle meshes based on spatial pooling. *Frontiers of Computer Science*, 12(4):798–812.

Galligan, F., Hemmer, M., Stava, O., Zhang, F., and Brettle, J. (2018). Google/draco: a library for compressing and decompressing 3d geometric meshes and point clouds.

Gumhold, S. and Amjoun, R. (2003). Higher order prediction for geometry compression. In *Proceedings of Shape Modeling International 2003*, SMI '03, page 59, USA. IEEE Computer Society.

Karni, Z. and Gotsman, C. (2000). Spectral compression of mesh geometry. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, page 279–286, USA. ACM Press/Addison-Wesley Publishing Co.

Kälberer, F., Polthier, K., Reitebuch, U., and Wardetzky, M. (2005). Freelence - coding with free valences. *Computer Graphics Forum*, 24(3):469–478.

Lavoué, G. (2011). A multiscale metric for 3d mesh visual quality assessment. *Computer Graphics Forum*, 30(5):1427–1437.

Lee, H., Alliez, P., and Desbrun, M. (2002). Angle-analyzer: A triangle-quad mesh codec. *Computer Graphics Forum*, 21(3):383–392.

Lobaz, P. and Váša, L. (2014). Hierarchical laplacian-based compression of triangle meshes. *Graphical Models*, 76(6):682–690.

Mamou, K., Zaharia, T., and Prêteux, F. (2009). Tfan: A low complexity 3d mesh compression algorithm. *Comput. Animat. Virtual Worlds*, 20(2-3):343–354.

Marras, S., Váša, L., Brunnett, G., and Hormann, K. (2015). Perception-driven adaptive compression of static triangle meshes. *Computer-Aided Design*, 58:24–33. Solid and Physical Modeling 2014.

Rossignac, J. (1999). Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):47–61.

Sim, J.-Y., Kim, C.-S., and Lee, S.-U. (2003). An efficient 3d mesh compression technique based on triangle fan structure. *Signal Processing: Image Communication*, 18(1):17–32.

Sorkine, O., Cohen-Or, D., and Toledo, S. (2003). High-pass quantization for mesh encoding. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '03, page 42–51, Goslar, DEU. Eurographics Association.

Torkhani, F., Wang, K., and Chassery, J.-M. (2014). A curvature-tensor-based perceptual quality metric for 3d triangular meshes. *Machine GRAPHICS & VISION*, 23(1/2):59–82.

Touma, C. and Gotsman, C. (1998). Triangle mesh compression. In *Proceedings of the Graphics Interface 1998 Conference, June 18-20, 1998, Vancouver, BC, Canada*, pages 26–34.

Váša, L. and Rus, J. (2012). Dihedral angle mesh error: a fast perception correlated distortion measure for fixed connectivity triangle meshes. *Computer Graphics Forum*, 31(5):1715–1724.

Váša, L. and Brunnett, G. (2013). Exploiting connectivity to improve the tangential part of geometry prediction. *IEEE Transactions on Visualization and Computer Graphics*, 19(9):1467–1475.

Váša, L. and Dvořák, J. (2018). Error propagation control in laplacian mesh compression. *Computer Graphics Forum*, 37(5):61–70.

Wang, K., Torkhani, F., and Montanvert, A. (2012). A fast roughness-based approach to the assessment of 3d mesh visual quality. *Computers & Graphics*, 36(7):808–818. Augmented Reality Computer Graphics in China.

Zhou, Q. and Jacobson, A. (2016). Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*.