

# Learning Mesh Geometry Prediction

Filip Háchá<sup>1</sup>[0000-0001-8956-6411] and Libor Váša<sup>1</sup>[0000-0002-0213-3769]

Department of computer science and engineering, Faculty of applied sciences,  
University of West Bohemia, Univerzitní 8, 301 00 Plzeň, Czech Republic  
{hachaf, lvasa}@kiv.zcu.cz

**Abstract.** We propose a single-rate method for geometry compression of triangle meshes based on using a neural predictor to predict the encoded vertex positions using connectivity and an already known part of the geometry. The method is based on standard traversal-based methods but uses a neural predictor for prediction instead of a hand-crafted prediction scheme. The parameters of the neural predictor are learned on a dataset of existing triangle meshes. The method additionally includes an estimate of the prediction uncertainty, which is used to guide the encoding traversal of the mesh. The results of the proposed method are compared with a benchmark method on the ABC dataset using both mechanistic and perceptual metrics.

**Keywords:** Mesh compression, Computer graphics, Deep learning

## 1 Introduction

The compression of triangle meshes is a very often solved problem in the field of computer graphics. Triangle meshes are a widely used representation of shapes, mainly due to their efficient rendering, simple structure, and good representation capabilities. In order to accurately represent even complex shapes using triangle meshes, triangle meshes reach a large number of vertices that carry information about the geometry of the shape. As the number of vertices increases, it is advisable to compress the triangle meshes to achieve compactness of representation.

A triangle mesh is the set of vertices, edges, and triangle faces  $\mathcal{M} = (V, E, F)$ , where the set of edges can be derived from the set of faces, and the mesh can thus only be described by the set of vertices and faces  $\mathcal{M} = (V, F)$ . The vertices of the mesh carry information about the *geometry* of the surface since they have a position in space. The triangle faces form the *connectivity* of the mesh and tell which triplets of vertices are to be connected into the resulting triangles.

Today, there is a large number of algorithms for compressing the geometry of triangle meshes that achieve a good ratio of bitrate to distortion. A significant group of these algorithms is connectivity-driven approaches that use the information contained in the connectivity to reduce the bitrate since the connectivity itself partially indicates the mesh geometry. These approaches also exploit the fact that, as the mesh is incrementally decoded, the decoder can use the already decoded part of the geometry to predict the rest of it. How the decoder performs

such a prediction is determined by the prediction scheme of the method. One of the simplest ways to predict a decoded vertex is the *parallelogram* prediction [21]. However, there are also more sophisticated methods that use, in addition to the vertex positions of adjacent triangles, e.g., vertex valence or already decoded interior angles of a triangle, such as the *weighted parallelogram* [22].

In this paper, we propose a prediction scheme based on encoded vertex prediction using a neural network that, similar to conventional prediction schemes, performs prediction based on an already decoded part of the geometry. We exploit the neural network’s ability to better predict the position of the encoded vertex across different meshes. We compare our results in terms of rate-distortion ratio with the *weighted parallelogram* as the reference method using mechanistic and perceptual mesh quality assessment metrics.

## 2 Related Work

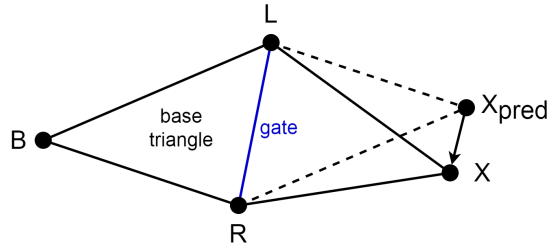
In the domain of compression single triangle meshes, we typically consider lossy compression, where the vertex coordinates are encoded with distortion while the connectivity is encoded losslessly. There is also a number of mesh-simplification approaches that allow for a lossy compression of connectivity. However, we do not consider these methods in this paper. When compressing a mesh, we can choose whether to encode the connectivity or the geometry first. Although there are methods that encode the mesh geometry first and then use its knowledge to encode the connectivity more efficiently [7, 15], most methods encode the connectivity first and then the geometry.

One of the simplest approaches for geometry compression is based on predicting the position of the encoded vertex using parallelogram prediction, proposed by Touma and Gotsman [21] in 1998. When encoding a vertex  $\mathbf{X}$  that incides with an already encoded base triangle  $\triangle \mathbf{BLR}$  over an edge  $(\mathbf{L}, \mathbf{R})$  (gate), we can predict its position using Equation 1 as shown in Figure 1. Then, instead of encoding the coordinates of the vertex  $\mathbf{X}$ , only the correction vector  $\mathbf{X} - \mathbf{X}_{\text{pred}}$  is encoded. This correction is typically further quantized, where the resolution of the quantization affects the distortion of the decoded mesh, and then it is compressed using an entropy encoder such as an arithmetic encoder.

$$\mathbf{X}_{\text{pred}} = \mathbf{L} + \mathbf{R} - \mathbf{B} \quad (1)$$

Also, in 1998, Taubin and Rossignac [20] proposed a compression method *topological surgery*. This method works with a spanning tree of the connectivity of triangle mesh. Based on the spanning tree, the encoded vertices of the mesh are then predicted from 2, 3, or 4 of its ancestors in the tree using linear extrapolation.

The connectivity of a triangle mesh can be efficiently encoded based on the encoding of the valences of vertices [1, 21]. Due to the non-uniform probability for different vertex valences, it is possible to use shorter *words* to compress more probable valences and longer *words* to compress less probable valences. Rossignac proposed a method for connectivity compression called *Edgebreaker* [17], which



**Fig. 1.** Diagram showing parallelogram prediction of the vertex.

is based on compressing a sequence of *CLERS* symbols that are formed by sequentially traversing the mesh and encoding the different ways in which an encoded vertex can be attached to an already decoded part of the mesh.

Geometry prediction was improved by Lee et al. with the Angle-Analyzer method [13], which encodes the vertex position using the interior angles of the triangle and the dihedral angle. This work also considers a priority-based mesh traversal, in which the priority of the decoded triangle is controlled based on the already decoded angles. Sim et al. [18] proposed an extension of the parallelogram prediction to dual parallelogram prediction, which predicts the position of an encoded vertex by making a prediction based on each pair of successive triangles in a triangle fan. Gumhold and Amjoun [8] proposed a prediction scheme based on separating the tangential and normal components of the correction. Their method first encodes the tangential coordinates of the vertex, which are predicted by the parallelogram. Subsequently, a higher-order surface is fitted to predict the normal component.

Kälberer et al. proposed *FreeLence*, a connectivity coding method based on free valences [12]. Free valence codes benefit from a geometry-guided mesh traversal. Mesh traversal is guided by the so-called *opening angle*, which is given by the difference of  $2\pi$  and the sum of the known angles around a given vertex. This information is also used to separate the encoded symbols into different contexts, which exploits the fact that each free valence has its own distribution of correction values. Courbet and Hudelot [6] derived a modified set of weights for various linear geometry predictors using Taylor expansion of the mesh geometry function. Váša and Brunnett [22] proposed *weighted parallelogram*, which uses a parallelogram predictor with weights, which are derived based on known angles that are already decoded and the valences of vertices.

In contrast to the above-mentioned single-rate compression methods, Pajarola and Rossignac [16] proposed a progressive coding method. This method first transmits a simplified coarse version of the original mesh and then encodes the vertex splits into batches to refine the coarse mesh progressively. The displacement of a new vertex that is created during vertex splitting is predicted based on the Butterfly subdivision scheme, which further leads to a reduction in the data rate. Cohen and Irony proposed another progressive method [5].

They also use the idea of averaging multiple parallelogram predictions to obtain a more accurate estimate of the position of the encoded vertex.

In addition to methods based on a prediction scheme that uses mesh connectivity and the decoded part of the geometry, there are also methods based on spectral analysis, such as Karni and Gotsman [9], which use a projection of the mesh geometry onto an orthonormal basis derived from the mesh topology. This approach offers the possibility of separating low and high-frequency signals and encoding them with different precision, given that different frequencies of distortion are perceived with different intensities.

Another popular geometry compression approach is the encoding of delta-coordinates [4,19,23], which are obtained as a result of discrete Laplace operators of the geometry function. For a more detailed look into the compression of triangle meshes, we refer the reader to the surveys of Alliez and Gotsman [2] or Maglo et al. [14].

When comparing algorithms for triangle mesh compression, it is essential how the distortion caused by lossy geometry compression is measured. For this purpose, quality assessment metrics are used. Mechanistic metrics, such as *Mean Squared Error* (MSE) of the vertices, Hausdorff distance, or *Average Absolute Surface Distance* (AvgD), are typically easy to compute. However, mechanistic metrics depend on properties such as the distance between points of a surface and often do not correlate well with the human perception of the distortion. In contrast, perceptual metrics such as *Dihedral Angle Mesh Error* (DAME) [24] or *Fast Mesh Perceptual Distance* [25] (FMPD) describe the distortion better in the context of human perception. These metrics typically depend on features such as curvature, roughness, or dihedral angles.

### 3 Compression Method

Our compression method, as well as many other connectivity-first compression methods, uses the *Edgebreaker* method [17] to compress the connectivity of a triangle mesh as a sequence of *CLERS* symbols. The connectivity of the triangle mesh is encoded first and can thus be used to remove redundancy in geometry encoding.

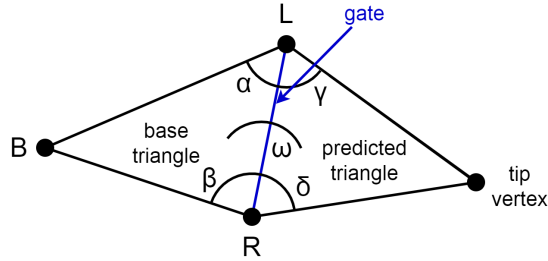
An improvement in compression ratio over other single-rate state-of-the-art methods, primarily the *Weighted parallelogram*, is achieved by our method at the geometry compression level. We use an approach similar to *FreeLence* or *Weighted parallelogram*. The method is based on predicting the position of the encoded vertex. This then makes it unnecessary to encode the positions themselves; only the corrections, i.e., the difference between the actual and predicted vertex positions, are encoded. Similarly to the above-mentioned methods, our method uses the base triangle together with information about vertex valences and already decoded angles to generate the vertex position prediction. While the prediction of the weighted parallelogram is given by an explicit equation, we learn this prediction through a neural network (further denoted as a neural predictor). As a result, among other things, we are able to incorporate additional

features that would be difficult to include in the *weighted parallelogram*. The acquired correction vectors are then quantized and further compressed using an arithmetic encoder.

### 3.1 Input Features

While training the neural network, it is necessary to properly describe the current configuration, which is mainly determined by the positions of the vertices of the base triangle but also by other available information about the already decoded geometry around the gate (edge over which a new vertex is encoded). For the purposes of the neural predictor, it is convenient that this data is suitably normalized in terms of its range. It is also desirable that the resulting feature vectors are invariant to transformations that should not, in principle, affect the shape of the predicted triangle. These are primarily rigid transformations (translation and rotation), but likewise, the description of the gate should be invariant to uniform scaling, which we would expect to affect the size of the predicted triangle in the same way, but not its shape.

A natural way of constructing feature vectors representing the shape of the base triangle, which is also offered, given the other information used, is the inner angles of the triangle. We can describe the shape of a base triangle using any two inner angles. We choose the pair of angles  $\alpha$  and  $\beta$  that lie next to the gate. The shape of the predicted triangle can also be described by the two inner angles  $\gamma$  and  $\delta$  and the dihedral angle of the base and predicted triangle  $\omega$  (see Figure 2). Given these five features, it is possible to reconstruct the shape of both triangles.



**Fig. 2.** Diagram showing features describing the base and the predicted triangles.

In addition to the shape of the base of the triangle described by the inner angles, the feature vectors contain information about the valence of the vertices **B**, **L**, **R** and **X**. Since the mesh connectivity is encoded before the geometry, this information is also available to the decoder side at the moment of computing the prediction. In order to normalize the vertex valences  $n_i$  of the  $i$ -th vertex, we convert them to the relative angle  $\xi_i$  from the total angle  $2\pi$  of a given vertex, as shown in Eq. 2.

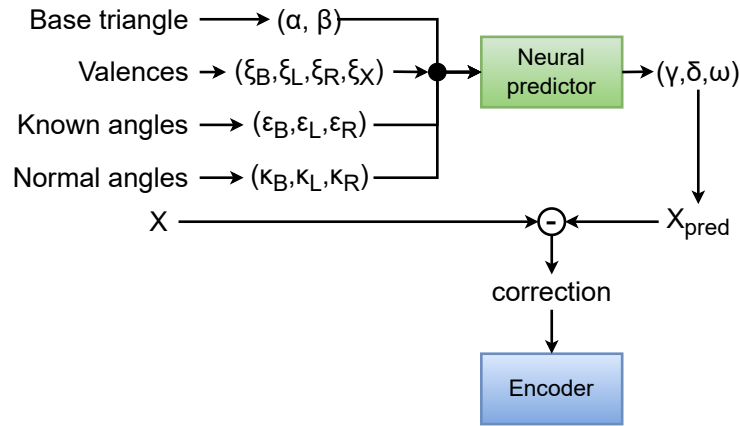
$$\xi_i = \frac{2\pi}{n_i} \quad (2)$$

Another information used by the neural predictor is the angle estimation based on the already decoded angles around the vertices  $\mathbf{B}$ ,  $\mathbf{L}$  and  $\mathbf{R}$ . The feature vector contains the proportion  $\epsilon_i$  of the difference between the angle  $2\pi$  and the sum of the  $k$  already decoded angles  $\varphi_j$  to the remaining number of angles that have not yet been decoded, as shown in Equation 3.

$$\epsilon_i = \frac{2\pi - \sum_{j=1}^k \varphi_j}{n_i - k} \quad (3)$$

To allow the neural predictor to work with the normal component of the prediction by predicting the dihedral angle  $\omega$ , the feature vector also contains information about the curvature of the surface around the base triangle. This information is represented by three angles. During mesh traversal, the vertex normal is estimated at the mesh vertices as the average of the normals of the adjacent triangles. These vertex normals  $\mathbf{n}_B$ ,  $\mathbf{n}_L$ ,  $\mathbf{n}_R$  of the base, left and right vertices are then compared with the normal of the base triangle. At each prediction, the angle between vertex normal and triangle normal is determined. We denote these angles as *normal angles* ( $\kappa_B, \kappa_L, \kappa_R$ ).

As a result, the neural predictor uses feature vectors  $\mathbf{x} \in \mathbb{R}^{2+4+3+2}$  as input and predicts a triplet of angles based on which the predicted vertex  $\mathbf{y} \in \mathbb{R}^3$  is reconstructed. The predictor neural network is then described by the function  $\mathbf{X}_{\text{pred}} : \mathbb{R}^{2+4+3+2} \rightarrow \mathbb{R}^3$ . Figure 3 shows a diagram with an outline of the normalization of the input features and the computation of the coded correction.



**Fig. 3.** Diagram showing an outline of the normalization of the input features and the computation of the coded correction.

### 3.2 Optimization

The neural network that represents our neural predictor is a fully connected feedforward network with 4 layers and 256 units in hidden layers. For the hidden layers, we use the *ReLU* activation function, and the output layer contains a modified hyperbolic tangent activation function such that the range of values matches the interval  $(0, 2\pi)$  (see Equation 4) given by the extreme values of dihedral angle.

$$\sigma(x) = \pi \cdot (\tanh(x) + 1) \quad (4)$$

The optimization algorithm *Adam* [10] with learning rate  $lr = 1 \cdot 10^{-4}$  is used to optimize the weights of the neural network. As a loss function for training the neural network, we use the *Mean Absolute Error* between the predicted triplet of angles  $(\gamma_{\text{pred}}, \delta_{\text{pred}}, \omega_{\text{pred}})$  and the true triplet of angles  $(\gamma, \delta, \omega)$  from the encoded triangle (see Equation 5).

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^n (|\gamma_{\text{pred}} - \gamma| + |\delta_{\text{pred}} - \delta| + |\omega_{\text{pred}} - \omega|) \quad (5)$$

The neural predictor is trained on a dataset acquired from a wide range of triangle meshes. The dataset is divided into the training and test sets. The training set is used to optimize the weights of the neural network. Part of the training dataset is used as a validation set for the early stopping of the optimization process. The test set is used to evaluate the rate-distortion measures. The test set data are separated from the rest at the full-mesh level, so that data from the meshes that were used by the neural network during learning are not used to measure the accuracy of the neural predictor. The dataset consists of pairs of base and predicted triangles that are found during the traversal of the meshes during their compression.

For the validation set, a different loss function is used during learning than for the training set. The validation set is not evaluated using the Mean Absolute Error between the predicted and true angles but the Mean Absolute Error between the coordinates of the predicted vertex position, which is reconstructed based on the predicted angles, and the true coordinates of the encoded vertex. This loss function better corresponds to the resulting corrections that are finally encoded during the actual mesh compression (see Equation 6).

$$\mathcal{L}_{\text{val}} = \frac{1}{n} \sum_{i=1}^n \|\mathbf{X}_{\text{pred}}(\gamma_{\text{pred}}, \delta_{\text{pred}}, \omega_{\text{pred}}) - \mathbf{X}\|_1 \quad (6)$$

### 3.3 Uncertainty prediction

In many connectivity-driven geometry compression methods, the order of encoded vertices is arbitrarily determined by the connectivity encoding method used, such as *Edgebreaker*. Since the connectivity and geometry decoding can

be done separately, there is no obstacle to choosing a different order of vertex encoding that will lead to an improved bitrate.

We propose an uncertainty-driven traversal in which the order of the encoded vertices is given by the estimated uncertainty of the prediction of the position of the encoded vertex. The goal of this approach is first to encode the vertices that are more likely to achieve a more accurate prediction and, therefore, necessitate the encoding of shorter correction vectors. In addition, a single vertex can generally be encoded over different edges (gates) with varying degrees of uncertainty. In this way, we are partially able to arrange shorter correction vectors before longer ones in the output stream and also encode vertices over more convenient gates.

Since the magnitude of the correction vectors is not normalized, it is necessary to normalize them before training the neural network. The magnitude of the correction vector can be expected to increase with the size of the triangle. Therefore, we divide the magnitudes of the correction vectors by the square root of the base triangle surface area. In turn, when the uncertainty is evaluated during the encoding process, the output of the neural network is then multiplied by that value:

$$e = \frac{\|\mathbf{X} - \mathbf{X}_{\text{pred}}\|}{\frac{1}{2}\|(\mathbf{L} - \mathbf{B}) \times (\mathbf{R} - \mathbf{B})\|} \quad (7)$$

In order to be able to estimate the uncertainty of the coded vertex prediction, we train a separate neural network  $u : \mathbb{R}^7 \rightarrow \mathbb{R}$  that uses the same feature vector that is used for the vertex position prediction itself to predict the uncertainty, which is correlated with the size of the correction vector. To optimize this neural network, we use Concordance Correlation Loss [3], which is used to optimize the correlation between ground-truth values and predicted values. It is defined as  $1 - \text{Concordance Correlation Coefficient}$  as shown in the following Equation:

$$\mathcal{L}_{\text{unc}} = 1 - \frac{2\rho_{eu}\sigma_e\sigma_u}{\sigma_e^2 + \sigma_u^2 + (\mu_e - \mu_u)^2}, \quad (8)$$

where  $e$  is the relative error in the prediction,  $u$  is estimated uncertainty,  $\rho_{eu}$  denotes Pearson correlation coefficient,  $\sigma$  is the standard deviation, and  $\mu$  is a mean value. The structure of the uncertainty estimation neural network is the same as the structure of the vertex position prediction network, except that it contains a different activation function at the output, specifically the square function.

For training the neural network estimating uncertainty, we use the same data as for training the neural predictor. However, outliers are excluded from this data. The filtering of outliers is done based on the Z-score of relative error  $e$ , which is defined as the distance from the mean divided by the standard deviation. For our purposes, we only keep samples with Z-scores less than 3. This helps to provide a more stable learning process, as degenerated triangles (having zero area) and nearly degenerated ones that are close to singular are removed in this step.



$$e_{\text{train}} = \left\{ e : \frac{e - \mu_e}{\sigma_e} < 3 \right\}_{i=1}^N \quad (9)$$

When using uncertainty estimation, the encoding traversal is controlled by a priority queue that contains the gates and their corresponding uncertainty estimations. In each iteration, the gate with the minimum uncertainty is dequeued, a new vertex is encoded, and then the uncertainty of the newly created gates is estimated and enqueued.

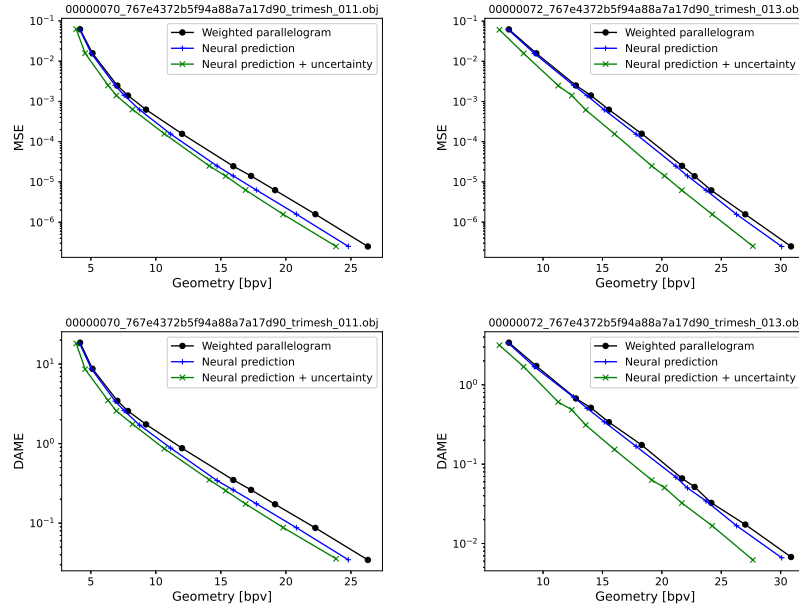
## 4 Experimental Results

To evaluate the proposed neural predictor, we compare our method with the implementation of the *weighted parallelogram* [22], a state-of-the-art single-rate method for connectivity-driven compression of triangle meshes. We use the *Mean Squared Error* (MSE) of mesh vertices, a commonly used metric for evaluating mesh compression algorithms that do not modify mesh connectivity, to measure mesh geometry distortion. In addition, we also use *Dihedral Angle Mesh Error* (DAME) [24], which is a perceptual metric that better correlates with how humans perceive the distortion of the compressed mesh.

The proposed compression method was tested in a scenario of a general geometry prediction, where the neural network of the predictor is trained for a general triangle mesh. The weights of the neural network are then part of the encoder implementation itself, and therefore, the encoder and decoder have these weights available without having to be transferred through the stream with the encoded mesh. For geometry prediction experiments, meshes from the ABC dataset [11] were used. Since both *weighted parallelogram*, as a reference method, and our proposed method require the input mesh to be orientable 2-manifold, meshes that the reference implementation could not handle were filtered out from the dataset.

We evaluated the rate-distortion function on the ABC dataset using the two metrics mentioned above on 100 meshes that were not used in learning the neural predictor, which was trained using 385 meshes. Figure 4 contains a representative example of RD curves comparing the neural predictor and *weighted parallelogram*. This graph shows the reduction of the data rate of the geometry using the proposed method over a large portion of the bitrate interval. To compare the distortion of different compression methods at the same bitrate, an RD-curve with sufficiently dense sampling was computed for each mesh, and the desired bitrate was found by interpolating the RD-curve. These statistics were measured for all mesh test sets containing 100 meshes, and the data rate was compared with that of the *weighted parallelogram*. Figure 5 contains the average relative improvement in bitrate of the neural predictor over the *weighted parallelogram* over the entire test dataset. These experiments show that the proposed method provides a better rate-distortion ratio for most of the test data.

To verify the contribution of the normal component prediction of the encoded vertex, which is represented by the dihedral angle, we compared the performance



**Fig. 4.** Representative comparison of RD curve *weighted parallelogram* and neural predictor on ABC dataset. The x-axis shows the bitrate of the compressed geometry (excluding connectivity), and the y-axis contains the bias of the compressed mesh as measured by the chosen metric. Top: MSE. Bottom: DAME.

of the same neural predictor with and without the normal component. This experiment was evaluated across the entire test set of the ABC dataset. Figure 6 contains a plot of the relative distortion of the coded mesh versus the *weighted parallelogram* measured using both MSE and DAME. These results show that it is advantageous to include the normal prediction since without the prediction of the normal component, no better results were obtained with respect to using *weighted parallelogram*.

## 5 Conclusions

In this paper, a new single-rate method for geometry compression of triangle meshes was proposed. The method uses an artificial neural network to predict the mesh geometry based on its connectivity and the already decoded part of the geometry around the vertex. The method uses principles similar to some existing state-of-the-art methods, such as parallelogram prediction, prediction based on the valence of mesh vertices, and estimation of interior angles of triangles.

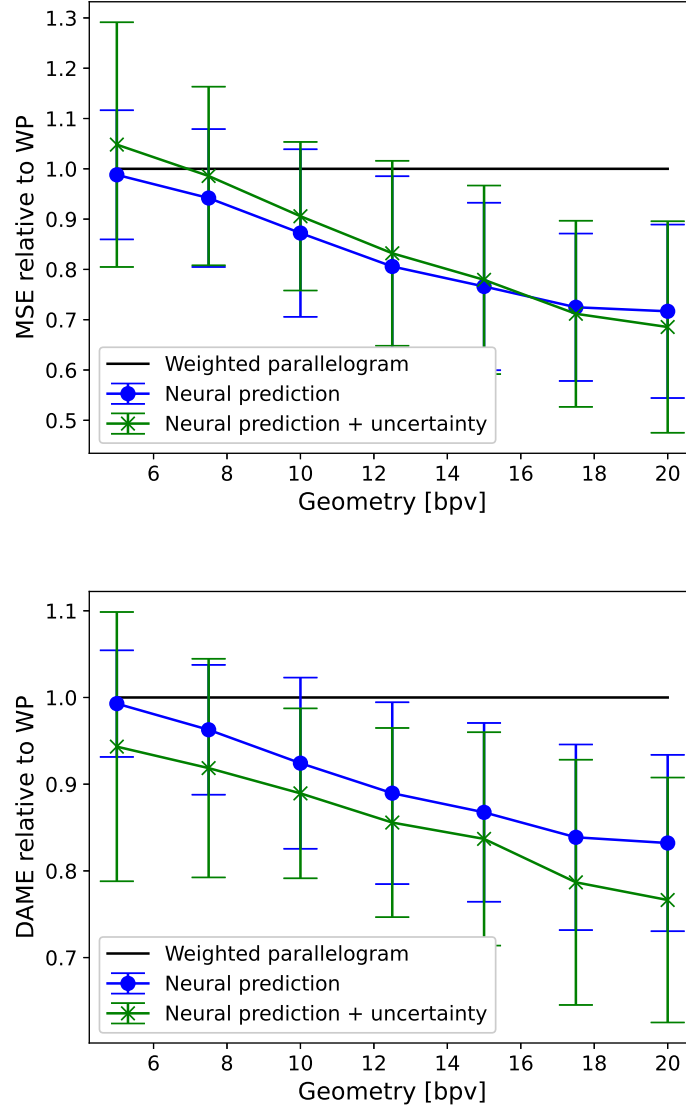
The performed experiments show that the proposed method is able to achieve a better ratio between bitrate and mesh distortion on a large dataset of static

meshes than the tested state-of-the-art method, using both mechanistic and perceptual metrics to assess mesh quality.

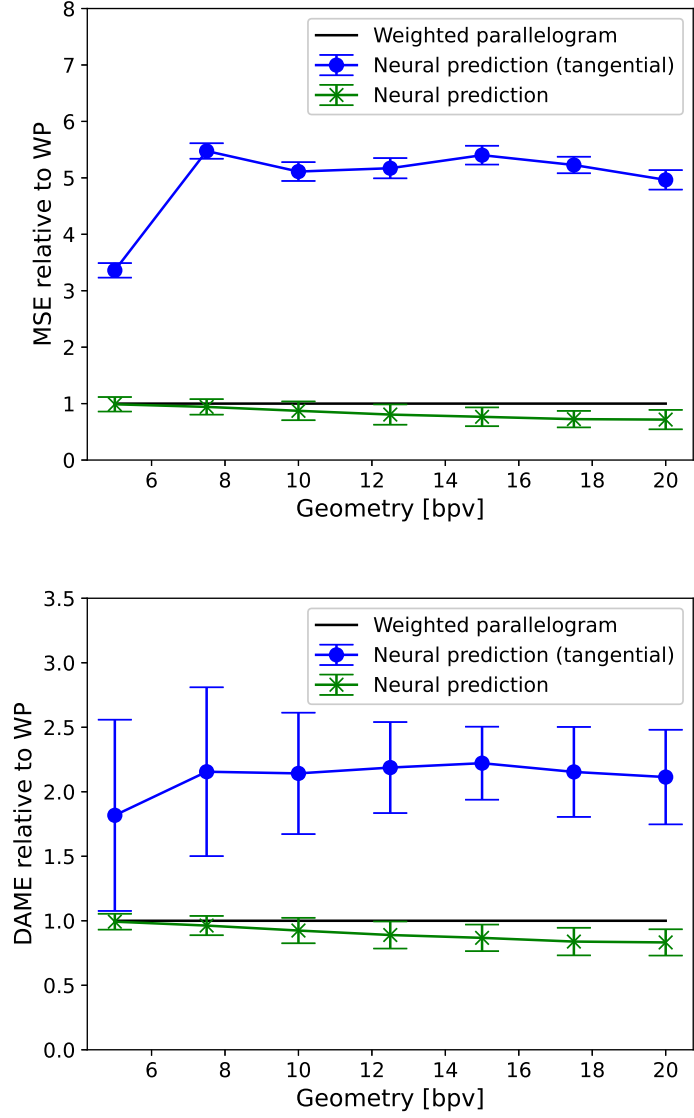
Although static mesh compression is already a well-researched area, it can be seen that using modern machine learning approaches, such as neural networks, can improve the existing state-of-the-art methods. In future work, it is possible to try to improve the neural predictor further. It might be interesting to try to incorporate other known local geometric properties of the compressed mesh into the feature vectors of the neural predictor. At the same time, it is worth exploring whether some of the global features could be used as latent code that would be constant for all predictions within a single mesh, further helping to improve the accuracy of the predictor. Global features describing meshes could be particularly useful in compressing larger sets of meshes with similar properties, such as sequences of triangle meshes with varying connectivity.

## Acknowledgement

The authors have no competing interests to declare that are relevant to the content of this article. This work was supported by the project 23-04622L, Data compression paradigm based on omitting self-evident information - COMPRO-MISE, 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.



**Fig. 5.** Comparison of the ratio of bitrate and distortion of compressed meshes of the proposed method relative to *weighted parallelogram*. The x-axis shows the bitrate of the compressed geometry (excluding connectivity), and the y-axis contains relative distortion to the *weighted parallelogram* measured by the chosen metric. Relative distortion smaller than one means lower distortion than the reference method. The chart contains the average relative distortion values across the test dataset, and the length of the bar corresponds to the variance of the relative distortion.



**Fig. 6.** Comparison of the relative bias of the neural predictor to WP using normal component prediction (green) and tangential component only prediction (blue). Top: MSE. Bottom: DAME.

## References

1. Alliez, P., Desbrun, M.: Valence-driven connectivity encoding for 3d meshes. *Computer Graphics Forum* **20**(3), 480–489 (2001). <https://doi.org/https://doi.org/10.1111/1467-8659.00541>, <https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-8659.00541>
2. Alliez, P., Gotsman, C.: Recent advances in compression of 3d meshes. In: Dodgson, N.A., Floater, M.S., Sabin, M.A. (eds.) *Advances in Multiresolution for Geometric Modelling*. pp. 3–26. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
3. Atmaja, B.T., Akagi, M.: Evaluation of error- and correlation-based loss functions for multitask learning dimensional speech emotion recognition. *Journal of Physics: Conference Series* **1896**(1), 012004 (apr 2021). <https://doi.org/10.1088/1742-6596/1896/1/012004>, <https://dx.doi.org/10.1088/1742-6596/1896/1/012004>
4. Chen, D., Cohen-Or, D., Sorkine, O., Toledo, S.: Algebraic analysis of high-pass quantization. *ACM Trans. Graph.* **24**(4), 1259–1282 (oct 2005). <https://doi.org/10.1145/1095878.1095880>, <https://doi.org/10.1145/1095878.1095880>
5. Cohen, R., Irony, R.: Multi-way geometry encoding. *Transactions on Computational Science* (01 2002)
6. Courbet, C., Hudelot, C.: Taylor prediction for mesh geometry compression. *Comput. Graph. Forum* **30**, 139–151 (03 2011). <https://doi.org/10.1111/j.1467-8659.2010.01838.x>
7. Dvořák, J., Káčereková, Z., Vaněček, P., Váša, L.: Priority-based encoding of triangle mesh connectivity for a known geometry. *Computer Graphics Forum* **42**(1), 60–71 (2023). <https://doi.org/https://doi.org/10.1111/cgf.14719>, <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14719>
8. Gumhold, S., Amjoun, R.: Higher order prediction for geometry compression. In: *2003 Shape Modeling International*. pp. 59–66 (2003). <https://doi.org/10.1109/SMI.2003.1199602>
9. Karni, Z., Gotsman, C.: Spectral compression of mesh geometry. In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. p. 279–286. SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., USA (2000). <https://doi.org/10.1145/344779.344924>, <https://doi.org/10.1145/344779.344924>
10. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. *International Conference on Learning Representations* (12 2014)
11. Koch, S., Matveev, A., Jiang, Z., Williams, F., Artemov, A., Burnaev, E., Alexa, M., Zorin, D., Panozzo, D.: Abc: A big cad model dataset for geometric deep learning. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2019)
12. Kälberer, F., Polthier, K., Reitebuch, U., Wardetzky, M.: Freelence - coding with free valences. *Computer Graphics Forum* **24**(3), 469–478 (2005). <https://doi.org/https://doi.org/10.1111/j.1467-8659.2005.00872.x>, <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2005.00872.x>
13. Lee, H., Alliez, P., Desbrun, M.: Angle-analyzer: A triangle-quad mesh codec. *Computer Graphics Forum* **21**(3), 383–392 (2002). <https://doi.org/https://doi.org/10.1111/1467-8659.t01-1-00598>, <https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-8659.t01-1-00598>
14. Maglo, A., Lavoué, G., Dupont, F., Hudelot, C.: 3d mesh compression: Survey, comparisons, and emerging trends. *ACM Comput. Surv.* **47**(3) (feb 2015). <https://doi.org/10.1145/2693443>, <https://doi.org/10.1145/2693443>

15. Marais, P., Gain, J., Shreiner, D.: Distance-ranked connectivity compression of triangle meshes. *Computer Graphics Forum* **26**(4), 813–823 (2007). <https://doi.org/https://doi.org/10.1111/j.1467-8659.2007.01026.x>, <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2007.01026.x>
16. Pajarola, R., Rossignac, J.: Compressed progressive meshes. *IEEE Transactions on Visualization and Computer Graphics* **6**(1), 79–93 (2000). <https://doi.org/10.1109/2945.841122>
17. Rossignac, J.: Edgebreaker: connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics* **5**(1), 47–61 (1999). <https://doi.org/10.1109/2945.764870>
18. Sim, J.Y., Kim, C.S., Lee, S.U.: An efficient 3d mesh compression technique based on triangle fan structure. *Signal Processing: Image Communication* **18**(1), 17–32 (2003). [https://doi.org/https://doi.org/10.1016/S0923-5965\(02\)00090-5](https://doi.org/https://doi.org/10.1016/S0923-5965(02)00090-5), <https://www.sciencedirect.com/science/article/pii/S0923596502000905>
19. Sorkine, O., Cohen-Or, D., Toledo, S.: High-Pass Quantization for Mesh Encoding. In: Kobbelt, L., Schroeder, P., Hoppe, H. (eds.) *Eurographics Symposium on Geometry Processing*. The Eurographics Association (2003). <https://doi.org/10.2312/SGP/SGP03/042-051>
20. Taubin, G., Rossignac, J.: Geometric compression through topological surgery. *ACM Trans. Graph.* **17**(2), 84–115 (apr 1998). <https://doi.org/10.1145/274363.274365>, <https://doi.org/10.1145/274363.274365>
21. Touma, C., Gotsman, C.: Triangle mesh compression. In: *Proceedings of the Graphics Interface 1998 Conference*, June 18–20, 1998, Vancouver, BC, Canada. pp. 26–34 (June 1998), <http://graphicsinterface.org/wp-content/uploads/gi1998-4.pdf>
22. Vasa, L., Brunnett, G.: Exploiting connectivity to improve the tangential part of geometry prediction. *IEEE Transactions on Visualization and Computer Graphics* **19**(09), 1467–1475 (sep 2013). <https://doi.org/10.1109/TVCG.2013.22>
23. Váša, L., Dvořák, J.: Error propagation control in laplacian mesh compression. *Computer Graphics Forum* **37**(5), 61–70 (2018). <https://doi.org/https://doi.org/10.1111/cgf.13491>, <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13491>
24. Váša, L., Rus, J.: Dihedral angle mesh error: a fast perception correlated distortion measure for fixed connectivity triangle meshes. *Computer Graphics Forum* **31**(5), 1715–1724 (2012). <https://doi.org/https://doi.org/10.1111/j.1467-8659.2012.03176.x>, <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2012.03176.x>
25. Wang, K., Torkhani, F., Montanvert, A.: A fast roughness-based approach to the assessment of 3d mesh visual quality. *Computers & Graphics* **36**(7), 808–818 (2012). <https://doi.org/https://doi.org/10.1016/j.cag.2012.06.004>, <https://www.sciencedirect.com/science/article/pii/S0097849312001203>, augmented Reality Computer Graphics in China