Graphical Models 76 (2014) 682–690

Contents lists available at ScienceDirect

Graphical Models

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

Hierarchical Laplacian-based compression of triangle meshes ☆

P. Lobaz [a], L. Váša [b,*]

[a] University of West Bohemia, Plzen, Czech Republic
[b] Chemnitz University of Technology, Chemnitz, Germany

ARTICLE INFO

Article history:
Received 22 April 2014
Received in revised form 22 September 2014
Accepted 23 September 2014
Available online 2 October 2014

Keywords:
Compression
Perceptual metric
Laplacian
Discrete shape operator
Encoding
Distortion

ABSTRACT

In this paper, we present an algorithm for efficient encoding of triangle meshes. The algorithm preserves the local relations between vertices by encoding their Laplacian coordinates, while at the same time, it uses a hierarchy of additional vertex constraints that provides global rigidity and low absolute error, even for large meshes. Our scheme outperforms traversal based as well as Laplacian-based compression schemes in terms of both absolute and perceived distortion at a given data rate.

1. Introduction

Triangle meshes are getting increasingly popular as a new medium storing shapes of 3D objects, thanks to recent advances in 3D scanning and even 3D printing. Publishing 3D meshes representing products is an attractive new way of marketing, and sharing 3D meshes may be the next step in computer-aided social interaction. Efficient storage of this kind of data is essential in both allowing displaying of the information even on mobile devices with low bandwidth on the content consumer side, as well as in allowing storage of many highly detailed models on the side of the content distributor.

The task of mesh compression is to store the triangle mesh in a file that is as small as possible. As with other kinds of media, some precision loss is allowable in most applications in order to achieve even smaller file sizes. This problem has been studied for about two decades now, yet only recently scientists started to seriously analyse the perception-related issues arising from the problem. In particular, measuring the amount of distortion due to the precision loss has recently received much attention. New error metrics have been proposed that capture the perceived distortion much better than mean squared error (MSE) and its derivatives. Along with this progress, new compression algorithms have been suggested, which attempt to minimise the perceived distortion.

One of the most efficient algorithms in this regard is the high-pass coding (HPC) proposed by Sorkine et al. [13]. The idea is to express the mesh in terms of local details, using a combinatorial discrete Laplacian. These details are in turn transmitted to the decoder, which solves a linear inverse problem. The approach is very efficient and outperforms all previous methods in terms of perceptual metrics, which usually focus on local similarity of meshes. Despite these advantages, users seem reluctant in adopting this technique, mainly because it lacks a mechanism that would avoid error accumulation. As a result, the performance of HPC in terms of MSE is rather poor, and the algorithm is unable to provide a guarantee

☆ This paper has been recommended for acceptance by Peter Lindstrom.
* Corresponding author.

http://dx.doi.org/10.1016/j.gmod.2014.09.003
1524-0703/

of maximum absolute dislocation of vertex positions. This in turn leads to problems when several meshes interact in 3D space by touching each other – even though the HPC compressed meshes show little perceptual distortion, they may intersect or not touch each other correctly, which is a disturbing artifact.

In this work, we propose an extension of the HPC algorithm which avoids this problem. We include additional data into the data stream, which describe the higher-level relations between vertices of the mesh, thus limiting the error accumulation and providing a considerable improvement as measured by MSE as well as perceptual metrics. Our extension is expressed in the terms of the original HPC encoding algorithm, and thus it does not require significant changes in the encoding/decoding implementation.

2. Related work

Compression of triangle meshes has been intensively studied in the past. Separate approaches have been proposed for compression of mesh connectivity and mesh geometry, and it has been shown that using information from connectivity improves the performance of geometry compression and vice versa. Most commonly, compression schemes build on encoding/decoding connectivity first, followed by a connectivity guided encoding of the geometry.

Several methods of connectivity compression have been proposed in the past. Connectivity is usually encoded without any data loss, while reindexing of indices is usually used in order to reduce the amount of data required. The Topological Surgery approach [15] encodes a vertex spanning tree and a triangle spanning tree that uniquely identify the connectivity. The Edgebreaker scheme [11] provides a guarantee of 4 bits per vertex by encoding the so-called op-codes for each triangle. Valence based coding approaches [1,7] build on the fact that the main part of connectivity information is contained in the vertex degrees. A theoretical bound of 3.245 bits per vertex has been derived [5] under the assumption that every possible connectivity is equally probable. The valence-based encoder provide performance even below this limit, exploiting the higher probability of highly regular (vertex degrees close to 6) nature of most practical meshes.

Encoding of mesh geometry is a task with much more freedom regarding the loss of precision. Most algorithms perform quantization of the floating point values at some stage, while some advanced algorithms have other sources of precision loss, such as neglecting high frequencies in the mesh.

One large class of algorithms works during a mesh traversal, which attaches vertices to the decoded part of the mesh one at the time. The new vertex is predicted in some way, such as using the parallelogram predictor [14] or some of its extensions [3,16]. Finally, a correction vector is encoded that is added to the prediction, yielding the decoded position of the vertex. If both the encoder and the decoder work with the same prediction, then this scheme effectively eliminates error accumulation. An alternative of this approach working with angles rather than position vectors has been also proposed [8].

There have also been more complex algorithms proposed, which do not work in the traversal-based fashion. An algorithm based on eigenvalue decomposition of the mesh connectivity matrix [6] uses transformation of the coordinate functions into a basis of the discrete Laplace operator. A basis reduction is applied to reduce the dimensionality of the data, and the remaining amplitudes are quantized and encoded.

The high-pass encoding [13,2] also builds on the discrete Laplace operator, only this time using it directly to transform the coordinate functions into the Laplacian (delta) coordinates. Together with the anchor points, this data allows the decoder to reconstruct the original vertex positions by solving a sparse system of linear equations. Since we build on this method, it will be described in more detail in Section 3.

Although mean squared error and Hausdorff distance have been used extensively for evaluation of the amount of distortion caused by mesh compression, it has been recently conclusively shown that these metrics provide only limited correlation with distortion perception [4]. User studies have been performed and metrics such as MSDM2 [9], FMPD [19] or DAME [18] have been proposed to provide better correlation with the results. Currently researchers are designing new compression algorithms that minimise the newly proposed metrics, while the traditional metrics keep their relevance in situations where multiple objects interact with each other. While the perception based metrics ensure that each object is visually indistinguishable from the original, the absolute metrics ensure that the interacting objects, such as touching hands or shoe touching a floor, stay in correct position with respect to one another.

Currently, high-pass coding provides the best results in terms of perceived distortion, while the traversal based methods work best in terms of mean squared error. In this paper, we propose an algorithm that outperforms both of these approaches both in terms of perceptual metrics and in terms of mean squared error.

3. Algorithm overview

A triangle mesh $\mathcal{M}$ is defined as a set of vertex positions $v_1, v_2, \ldots, v_V$, representing points in 3D space and referred to as geometry, and a set of index triplets $t_1 = (t_1^1, t_1^2, t_1^3), t_2 = (t_2^1, t_2^2, t_2^3), \ldots, t_T = (t_T^1, t_T^2, t_T^3)$, referred to as connectivity. We assume that the connectivity has been transmitted to the decoder, and thus it is available at both sides of the transmission. The task is to encode the geometry as efficiently as possible.

In the high-pass coding of triangle meshes, the positions of vertices are not encoded as absolute coordinates. Instead, for each vertex, the so-called *Laplacian coordinates* are computed:

$$d_i = \left(\frac{1}{\|N(i)\|} \sum_{j \in N(i)} v_j \right) - v_i, \tag{1}$$

where $N(i)$ is the set of all vertices incident with vertex v_i, also known as the 1-ring. The Laplacian coordinates represent the relative position of a vertex with respect to the centre of mass of its neighbours, i.e. it captures the local relations between vertices. Computing all Laplacian coordinates for all vertices can be expressed as a matrix multiplication $\mathbf{r} = L\mathbf{g}$, where $\mathbf{r} = (d_1, d_2, \ldots, d_V)^T$, $\mathbf{g} = (v_1, v_2, \ldots, v_V)^T$ and L is in fact a matrix representation of a combinatorial Laplace operator on the mesh $\mathcal{M}$.

The intention is to transmit the vector $\mathbf{r}$ instead of $\mathbf{g}$, because the values in $\mathbf{r}$ have a much smaller entropy, yet they capture the necessary information about vertex relations. Unfortunately, the matrix L is rank deficient. In order to resolve this problem, additional rows are added to L, one for each connected component of the mesh. These rows contain only a single unit value at a position corresponding to one vertex of the connected component. The resulting rectangular matrix L^* is then used to obtain an extended vector $\mathbf{r}^*$, which additionally contains coordinates of one vertex per connected component. This vector is quantized and transmitted to the decoder, which in turn solves the system $L^*\overline{\mathbf{g}} = \overline{\mathbf{r}^*}$, where $\overline{\mathbf{r}^*}$ is the decoded vector $\mathbf{r}^*$, which differs from $\mathbf{r}^*$ slightly due to quantization.

The high-pass coding is very efficient at preserving the local distribution of vertices, since the encoded residuals (i.e. Laplacian coordinates) directly capture the local relations. On the other hand, the scheme does not prevent error accumulation in any way, and thus vertices that are located topologically far from anchor points may get reconstructed at positions that differ significantly from the original positions. This is captured by a rather poor performance of the algorithm in terms of mean squared error. Moreover, this effect is strongly affected by the random character of the accumulation of quantization error in areas that are distant from anchor points, and thus sometimes even increasing the precision of quantization actually leads to an increase of mean squared error.

In our extended version of the algorithm, we use additional equations that capture relations between topologically distant vertices as well, and build hierarchical "suspension" structure that is supported by the anchor points. These additional equations dramatically reduce the mean squared error. The following section describes the procedure in detail.

3.1. Construction of vertex hierarchy

In our scheme, we classify the vertices into several layers. Each vertex is assigned an integer, which determines at which level of a hierarchical structure the vertex is located, while initially, all vertices are assigned to level 0. We aim at a uniform distribution of vertices into layers, and thus we use the procedure described in Algorithm 1. The steps of the algorithm are also illustrated by Fig. 1.

Algorithm 1. Vertex level assignment. In our implementation, we use *maxCount* = 100, i.e. new levels are added until there are less than 100 vertices on the highest level of the hierarchy. The asterisk at line 10 refers to Fig. 1

```
for each vertex v_i do
    initialize level[i] = 0;
    initialize N_0(i) to the 1-ring neighbourhood of
    vertex v_i;
end
currentLevel = 0;
repeat
    promotedCount = 0;
    for each vertex v_i where level[i]=currentLevel (in
    order of a random permutation) do //  (*)
        if there is a vertex v_j ∈ N_currentLevel(i) such that
        level[j] = currentLevel+1 then
            // the vertex is not suitable
               for promotion
            continue;
        else
            level[i]++;
            promotedCount++;
        end
    end
    for each promoted vertex v_i do
        breadth-first search from v_i over a graph
        connectivity defined by the set N_currentLevel,
        until 6 vertices v_j are found such that level[j] =
        level[i]. Add these 6 vertices to the higher
        neighbourhood N_level[i](i)
    end
    currentLevel++;
until promotedCount<maxCount;
```

The level assignment algorithm ensures that no two neighbouring vertices are promoted to a higher level. It also provides a set of higher level neighbours $N_{level}(i)$ for each higher level vertex. These neighbourhoods are then used for additional equations described in the next subsection. Note that we do not create additional equations for level 1 vertices, because they would create a too large number of additional residuals that would have to be encoded, while in our experiments, they did not bring a significant improvement of compression performance. Also note that the hierarchy itself does not require any additional data, since it can be constructed at the decoder from the mesh connectivity, using a predefined seed for the random number generator used in the algorithm.

3.2. Higher level equations

Having the vertex hierarchy with the highest level M, we add an additional equation for each vertex v_i such that $M > level[i] > 1$. The equation has the same form as (1)

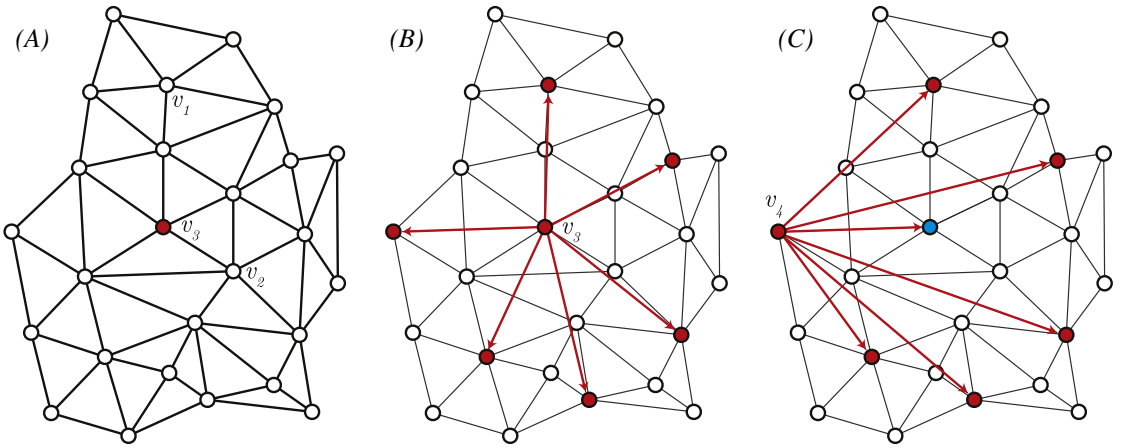

Fig. 1. Steps of the level assignment algorithm, depicting the intermediate steps at the point denoted by (∗) in Algorithm 1. In *A*), most of the vertices are still assigned to level 0 (white), only v_3 has been assigned to level 1 (red). At this point, vertex v_1 can (and will) be promoted to level 1, because there is no level 1 vertex in its neighbourhood $N_0(1)$, which is equivalent to its topological neighbourhood. In contrast to that, vertex v_2 cannot be promoted, because it has a neighbour that has already been promoted. The situation *B*) depicts the vertices in the next iteration of the algorithm. The vertex v_3 can be further promoted, since in its neighbourhood $N_1(3)$ (depicted by red arrows) there is no other vertex that has been promoted in this round. In *C*), after the promotion of v_3 (now blue), no other depicted level 1 vertex can be promoted. In the particular example, the neighbourhood $N_1(4)$ of the vertex v_4 contains the promoted vertex v_3 and therefore v_4 cannot be promoted. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

with the only difference that the higher level neighbourhood is used, i.e. we add euqations of following structure:

$$d_k = \left(\frac{1}{\|N_{level[i]}(i)\|} \sum_{j \in N_{level[i]}(i)} v_j \right) - v_i, \tag{2}$$

where k is used as index of the additional equation. These additional equations capture the relations between more distant vertices of the mesh, and provide a certain kind of rigidity to the reconstruction, which is hard to achieve by only working with the 1-ring neighbourhoods. Note that we do not use equations on level M, instead, we use the vertices on level M as anchor points. This way, we build a stable structure, where the position of level M vertices is fixed, and positions of level $K-1$ vertices are directly connected to the positions of vertices on level K. This suspension structure effectively limits the error accumulation by ensuring that each vertex has an anchor point that affects it over no more than $\mathcal{O}(\log(V))$ equations.

3.3. Prediction of higher level residuals

Each additional equation produces a residual (Laplacian coordinates with respect to a wider neighbourhood) which must be transmitted to the decoder. In contrast to the residuals obtained from the original Laplacian, the residuals at higher levels have generally much higher magnitude, as they relate vertices that are distant from each other. Moreover, despite of the level assignment procedure that attempts to distribute the levels evenly, it is often the case that the vertex lies at a position that differs considerably from the centre of mass of its neighbours, which leads to larger Laplacian coordinates. It is therefore desirable to perform a prediction of these in order to make their distribution more narrowly distributed around zero and thus reduce their entropy.

In the proposed scheme, the prediction is done for each level separately. The first step is equivalent to the high pass encoding, i.e. the decoder receives the residuals of all vertices with respect to their topological neighbourhood, and the quantized positions of the anchor vertices. This allows reconstructing the first approximation of the mesh geometry $\overline{\mathbf{g^0}} = (\overline{v_0^0}, \overline{v_1^0}, \ldots, \overline{v_V^0})^T$ by solving $L^* \overline{\mathbf{g^0}} = \overline{\mathbf{r^*}}$. This in turn allows predicting the Laplacian coordinates corresponding to the additional equations related to level 1 vertices as

$$\widehat{d_k} = \left(\frac{1}{\|N_1(i)\|} \sum_{j \in N_1(i)} \overline{v_j^0} \right) - \overline{v_i^0}. \tag{3}$$

The encoder also evaluates this prediction and transmits only a correction $c_k = d_k - \widehat{d_k}$. The decoder can now reconstruct the Laplacian coordinates as $\overline{d_k} = \overline{c_k} + \widehat{d_k}$. The probability distribution of the corrections c_k is much narrower and their entropy is much smaller than that of the residuals d_k.

The reconstructed residuals $\overline{d_k}$ corresponding to equations related to vertices at level 2 are now appended to the vector $\overline{r^*}$, forming an extended vector $\overline{r_1^*}$. We can now extend the matrix L^* to L_1^* by appending rows corresponding to the additional equations related to level 2 vertices. This now allows solving for an improved reconstruction $L_1^* \overline{\mathbf{g^1}} = \overline{\mathbf{r_1^*}}$. The reconstruction $\overline{\mathbf{g^1}}$ is used to predict the residuals d_k related to level 3 vertices, allowing for another extension of the system matrix, until the highest level with matrix L_{M-2}^* is reached, which is used for the final reconstruction. The structure of the involved linear systems is illustrated in Fig. 2.

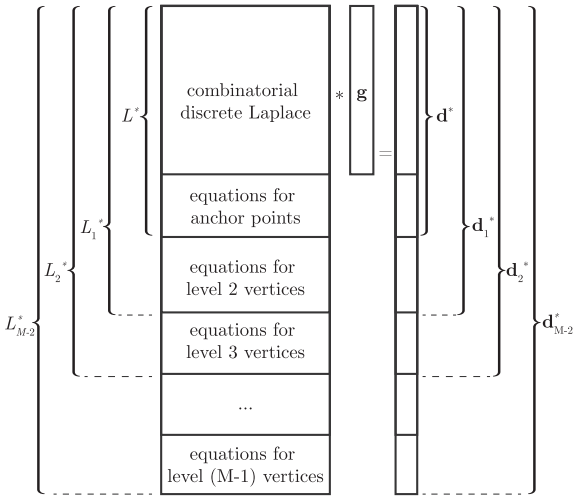

Fig. 2. Matrices used in the reconstruction process.

3.4. Encoding of residuals

After the encoding the whole geometry is represented by a set of V quantized residuals d_i, resulting from Eq. (1), quantized coordinates of anchor points (their number is determined by the number of vertices on the highest level), and quantized corrections c_k for each vertex on levels from 2 to $(M-1)$. In our implementation, we use a simple uniform quantization, i.e. each floating point value is multiplied by a user specified constant Q and rounded to the nearest integer. We encode these integer values using a context adaptive binary arithmetic coder similar to the CABAC implementation [10]. We use a separate context for the residuals, anchor points, and for each level of corrections. At the decoder, the integers are then simply multiplied by $1/Q$.

4. Results

In our experiments, we have measured the amount of data required to encode the geometry, i.e. our numbers do not include the bits required for encoding of connectivity. Our scheme can be used with any connectivity compression algorithm, which is also true for all the state of the art algorithms we have compared against. For the comparison, we have used four error metrics:

- Mean squared error, which captures the absolute dislocations of vertices.
- The visual error [6], which is in fact a combination of mean squared error and discrete shape operator difference. We are using the weighting constant $\alpha = 0.15$ as suggested in [13]. This metric combines absolute and local error.
- The FMPD [19] perceptual error metric.
- The DAME [18] perceptual error metric.

Figs. 3–6 represent the typical result of our algorithm in comparison to the high pass coding (HPC) [13], parallelogram prediction [14] and weighted parallelogram prediction [16] on a single high resolution model. Fig. 3 shows that our algorithm provides results on par with parallelogram prediction, and is only slightly worse than the weighted parallelogram scheme, while it provides a massive improvement against HPC. At the same time, Figs. 5 and 6 show that our scheme outperforms both par-

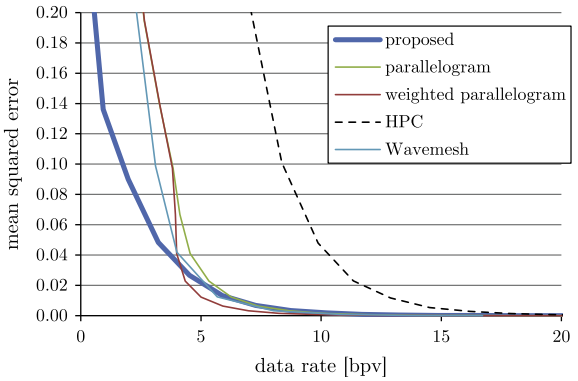

Fig. 3. Compression performance on the Blattkachel model measured by mean squared error.

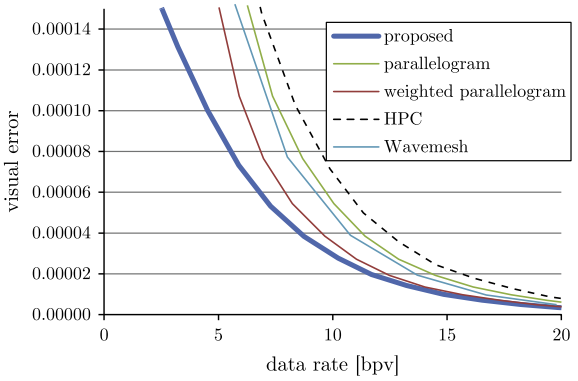

Fig. 4. Compression performance on the Blattkachel model measured by visual error [6].

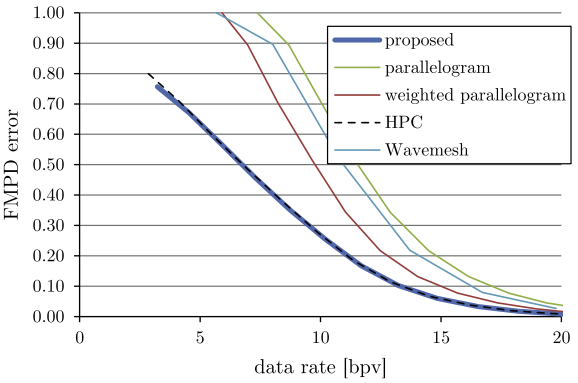

Fig. 5. Compression performance on the Blattkachel model measured by FMPD error [19].

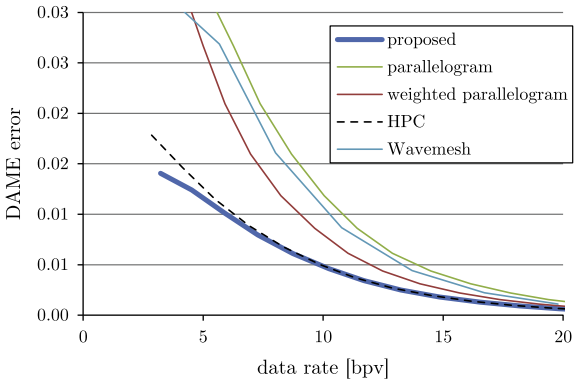

Fig. 6. Compression performance on the Blattkachel model measured by DAME error [18].

allelogram prediction and weighted parallelogram prediction schemes in terms of perceptual error, providing results on par with HPC. In terms of the visual error, which combines both local and absolute error, our algorithm provides the best results. Similar results were obtained for other models as well, as documented in Table 1.

We have also compared our results against a simpler approach of increasing the number of anchor points in the HPC algorithm. The typical results are shown in Figs. 7 and 8. The figures show that by adding random anchor points, the mean squared error is indeed improved, however at the cost of an immediate performance drop with respect to a perceptual metric FMPD. The performance drop is caused by increasing the data rate by adding anchor points which are not predicted, i.e. have a rather high entropy. By adding more anchor points, the FMPD performance deteriorates further, while the improvement of MSE performance stops long before reaching the performance of the proposed algorithm, when about 3% of vertices are used as anchors. Adding even more anchors only leads to decrease in performance in all metrics.

Note that it would be possible to promote specifically those vertices with largest error to anchors, as done in [12], probably leading to better results. Although it is possible to quickly update the factorization after adding a single anchor, one still has to perform a (at least) linear step of solving the factorised system and finding the next vertex with largest error. Since the total number of anchors that have to be added is linear, it would lead to a quadratic complexity of the algorithm.

As for the computational complexity of our algorithm, is comparable with the HPC. The main bottleneck of the HPC (and our) algorithm is the factorization and solution of the extended Laplacian matrix, both of which work for sparse matrices in roughly linear time. More generally, if the factorization has a complexity of $\mathcal{O}(f_1(V))$, where V is the

Table 1
Results on different models. The data rates (row 4) are given in bits per vertex, and they do not include the data required for storing the mesh connectivity. PP stands for parallelogram prediction [14], WPP stands for weighted parallelogram prediction [16], HPC stands for high-pass coding [13], and Wavemesh refers to the method proposed in [17]. The colour coding is applied on comparable results, i.e. error values of the same model, compressed at the same data rate and evaluated using the same error metric. One such set of results is marked by the blue rectangle.

	daz_body		blattkachel		hand		chindragon		maxplanck		bunny		ramesse	
vertices	59727		229330		327323		655980		25445		35946		826266	
triangles	119344		456419		654666		1311956		50801		71888		1652528	
bits per v.	6	12	6	12	6	12	6	12	6	12	6	12	6	12
MSE														
proposed	5.79E-4	3.84E-5	1.28E-2	8.05E-4	2.45E-2	1.33E-3	4.32E-3	2.54E-4	9.12E-3	5.52E-4	4.25E-3	2.17E-4	1.61E-2	9.08E-4
PP	1.25E-2	2.67E-4	1.52E-2	6.54E-4	1.87E-2	3.71E-4	6.25E-3	1.60E-4	1.74E-2	6.52E-4	3.77E-3	6.22E-5	2.00E-2	5.13E-4
WPP	7.06E-4	1.92E-5	6.06E-3	2.72E-4	2.19E-2	5.67E-4	4.97E-3	1.09E-4	6.05E-3	2.10E-4	5.31E-3	1.34E-4	8.95E-3	7.43E-4
HPC	7.65E-3	3.40E-4	3.72E-1	1.41E-2	9.41E-1	3.60E-2	2.02E-1	1.14E-2	6.89E-2	4.26E-3	3.66E-2	1.64E-3	8.26E-1	4.92E-2
Wavemesh	3.58E-3	7.69E-5	1.12E-2	5.71E-4	1.25E-2	3.27E-4	3.48E-3	1.56E-4	1.17E-2	6.06E-4	7.03E-3	1.57E-4	1.18E-2	5.09E-4
Visual error														
proposed	5.33E-7	1.40E-7	7.16E-5	1.86E-5	8.46E-7	1.92E-7	1.42E-5	3.45E-6	8.48E-6	2.17E-6	3.58E-7	8.27E-8	8.66E-5	2.06E-5
PP	4.98E-6	6.86E-7	1.73E-4	3.40E-5	2.25E-6	5.18E-7	3.73E-5	5.89E-6	2.86E-5	4.50E-6	6.58E-7	9.16E-8	2.24E-4	3.39E-5
WPP	1.13E-6	1.84E-7	1.05E-4	2.18E-5	1.70E-6	2.50E-7	3.34E-5	5.00E-6	1.41E-5	2.55E-6	8.38E-7	1.32E-7	1.36E-4	2.61E-5
HPC	1.01E-6	2.32E-7	1.90E-4	3.99E-5	2.37E-6	4.71E-7	4.64E-5	1.10E-5	1.42E-5	3.60E-6	6.26E-7	1.39E-7	2.84E-4	6.81E-5
Wavemesh	2.76E-6	3.78E-7	1.43E-4	3.07E-5	1.87E-6	4.53E-7	2.78E-5	5.64E-6	2.18E-5	4.22E-6	8.91E-7	1.37E-7	1.60E-4	3.11E-5
FMDP														
proposed	2.25E-1	4.56E-2	5.58E-1	1.54E-1	9.26E-1	3.32E-1	6.84E-2	8.91E-3	5.13E-1	1.32E-1	1.74E-1	2.33E-2	1.85E-1	2.26E-2
PP	1.00E+0	3.16E-1	1.00E+0	4.47E-1	1.00E+0	3.73E-1	4.02E-1	3.14E-2	1.00E+0	3.62E-1	5.68E-1	3.45E-2	7.35E-1	9.35E-2
WPP	4.57E-1	9.40E-2	9.91E-1	2.59E-1	1.00E+0	4.85E-1	3.56E-1	2.41E-2	9.08E-1	1.96E-1	6.91E-1	6.45E-2	6.60E-1	5.23E-2
HPC	2.21E-1	4.50E-2	5.60E-1	1.55E-1	9.35E-1	3.44E-1	6.05E-2	8.08E-3	5.43E-1	1.45E-1	1.86E-1	2.49E-2	2.21E-1	2.33E-2
Wavemesh	7.43E-1	1.93E-1	9.86E-1	3.93E-1	9.68E-1	3.26E-1	2.82E-1	2.96E-2	9.80E-1	3.34E-1	6.82E-1	6.79E-2	6.42E-1	8.53E-2
DAME														
proposed	6.30E-7	1.64E-7	9.95E-3	3.23E-3	1.72E-6	4.89E-7	1.05E-3	2.53E-4	1.93E-5	5.44E-6	6.06E-8	1.43E-8	2.22E-2	5.85E-3
PP	6.86E-6	9.18E-7	2.79E-2	7.62E-3	1.98E-6	4.85E-7	3.69E-3	5.83E-4	6.87E-5	1.37E-5	1.42E-7	1.92E-8	6.25E-2	1.32E-2
WPP	9.07E-6	2.48E-7	2.05E-2	4.96E-3	4.06E-6	7.46E-7	3.33E-3	4.96E-4	4.35E-5	7.79E-6	1.84E-7	2.91E-8	5.49E-2	8.94E-3
HPC	6.36E-7	1.65E-7	1.06E-2	3.25E-3	1.88E-6	5.15E-7	1.00E-3	2.43E-4	2.11E-5	5.92E-6	6.37E-8	1.51E-8	2.34E-2	5.98E-3
Wavemesh	3.15E-6	4.97E-7	2.54E-2	6.89E-3	1.66E-6	4.39E-7	2.75E-3	5.59E-4	5.86E-5	1.28E-5	1.83E-7	2.97E-8	5.15E-2	1.23E-2

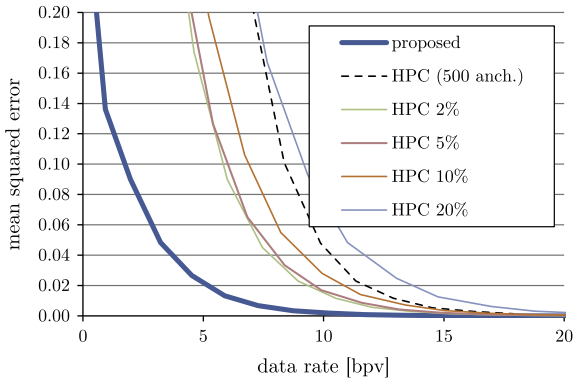

Fig. 7. Compression performance on the Blattkachel model measured by mean squared error, compared with HPC with varying number of anchor points.

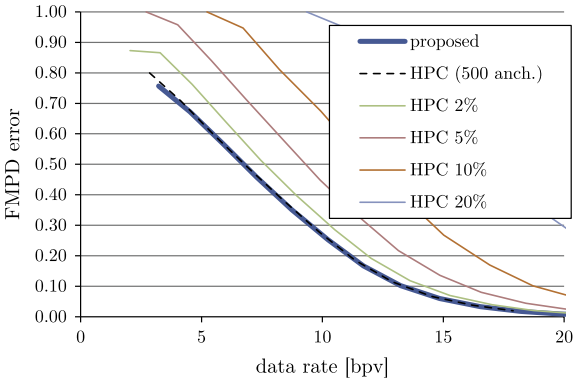

Fig. 8. Compression performance on the Blattkachel model measured by FMPD error [19], compared with HPC with varying number of anchor points.

number of vertices and $f_1(x)$ is some implementation dependent function, and the solution of the system based on the factorization has a complexity of $\mathcal{O}(f_2(V))$, one can express the overall decoding complexity of HPC as $\mathcal{O}(f_1(V)+f_2(V))$, while our algorithm runs in $\mathcal{O}(f_1(V)\log(V)+f_2(V)\log(V))$, because the decomposition and solution of the system has to be performed $\log(V)$ times. If both functions $f_1(x)$ and $f_2(x)$ are linear, then the complexity of the proposed algorithm is $\mathcal{O}(V\log(V))$ for both encoding and decoding. The Tables 2 and 3 report the times required for encoding and decoding. The reported times show that our algorithm is indeed slower than the competing algorithms, yet the complexity remains feasible.

Table 3
Times required for building the hierarchy of test models, in milliseconds.

Model	Time to build hierarchy	T
Maxplanck	162	50,801
Bunny	209	71,888
Daz_body	321	119,344
Blattkachel	1288	229,330
Hand	1747	654,666
Chindragon	3678	1,311,956
Ramesse	5273	1,652,528

Should the encoding and/or decoding time pose a problem for applicability of the proposed algorithm, then it is possible to speed the algorithm up at the cost of sacrificing some of the compression performance. Table 4 shows that a majority of bytes ($>98\%$) is spent to encode the basis level (i.e. the normal Laplacian coordinates) and the first level of the hierarchy. Therefore by omitting the prediction for higher levels of the hierarchy, one can speed the algorithm up while impairing the compression performance only marginally. In such case the system is going to be solved only twice at the decoder, yielding a performance that is no more than twice slower than the HPC. Another possibility of improving the processing times would naturally be to carefully optimise the code.

Finally, Fig. 9 shows that the improvement achieved by our scheme is well visible in realistic situations. Our scheme provides a good performance in both absolute error (in contrast with HPC) and perceptual error (in contrast with traversal based algorithms). It is also interesting to note that visually, our result in Fig. 9 seems better than the one by HPC, although the measured perceptual distance is roughly equal in both FMPD and DAME. This seems to indicate that there exists a practical kind of artifacts that are well perceivable, yet not detected by these perceptually motivated metrics. It would be interesting to perform a subjective experiment involving the compression results similar to the ones depicted here, that would confirm our hypothesis.

4.1. Conclusions

We have demonstrated that by adding a support structure of additional equations it is possible to construct an algorithm that provides excellent results both in terms of perceived and absolute distortion. Results provided by our compression scheme are visibly better than those of

Table 2
Times required for encoding and decoding of test models, in milliseconds. The times for the proposed algorithm include the times required for building the hierarchy.

	Daz_body		Blattkachel		Hand		Chindragon		Maxplanck		Bunny		Ramesse	
	Enc.	Dec.	Enc.	Dec.	Enc.	Dec.	Enc.	Dec.	Enc.	Dec.	Enc.	Dec.	Enc.	Dec.
Proposed	5969	5981	30,550	30,603	40,392	40,454	128,638	128,769	1782	1785	3298	3304	185,319	185,475
HPC	640	1566	2434	6527	3273	8901	6808	20,667	293	671	406	942	9972	29,643
PP	212	190	1033	958	1438	1332	3051	2836	78	69	119	106	4143	3844
WPP	509	271	1984	1310	2658	1797	5444	3750	275	109	365	168	7554	5148
Wavemesh	1420	446	6003	1429	7388	1941	12,137	2902	646	259	814	281	21,990	5310

Table 4
Number of bytes required for encoding of different levels for different models. The highest level represents the anchor points.

Level	Daz_body	Blattkachel	Hand	Chinadragon	Maxplanck	Bunny	Ramesse
0	51,962	366,888	434,954	816,240	39,278	45,473	1,351,937
2	2871	9218	12,381	25,685	911	1323	33,643
3	700	2126	2913	5846	239	327	7674
4	195	553	764	1533	508	100	1998
5	429	135	211	405			503
6		355	441	106			126
7				213			315

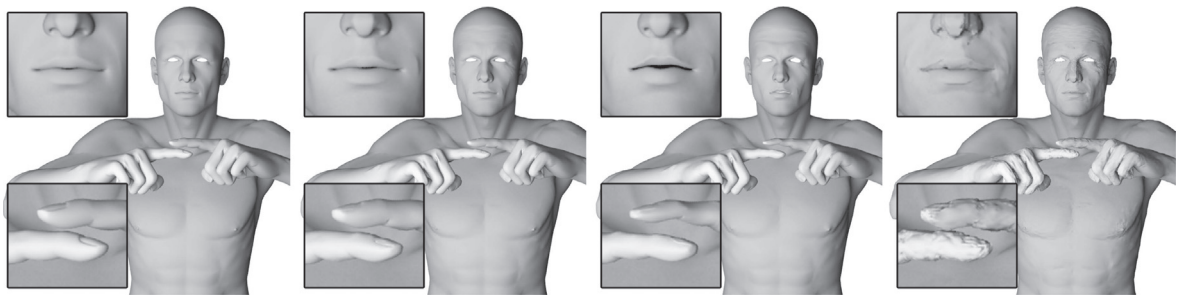

Fig. 9. Visual comparison of performance. From left to right: original mesh, result of proposed algorithm, result of HPC algorithm, result from weighted parallelogram prediction algorithm. All decompressed meshes were obtained from a 32 kB size file.

the state of the art, and this conclusion is supported by the measured values.

The construction of intermediate reconstructions leads to an increase in running time, which is however only of order $\mathcal{O}(\log(V))$ with respect to HPC. The algorithm running time remains within practical range.

In the future, we intend to focus on improving the performance of the algorithm, both in terms of efficient compression, as well as fast execution. We believe that the result of intermediate reconstructions can be used to accelerate the solution in the later steps, thus speeding the algorithm up.

Another topic of future research is the possibility of encoding the values in reversed order, which would allow the user to display an intermediate decompressed mesh even before the whole data stream has been decoded. Such feature is available in our implementation as well, however, the first approximation is only available after the data from the lowest level (which take up the largest part of the stream) are decoded. Reversing the order of encoding would allow displaying an intermediate reconstruction much sooner, however, the influence of such change on the compression performance is to be investigated.

Finally, our results seem to indicate that the used perceptual metrics fail to correctly evaluate the amount of perceived distortion in some cases. It is an interesting pointer for future research to confirm this by a statistically justified subjective experiment. We also believe that the character of the proposed compression method and the proposed hierarchical Laplacian may also be used as a starting point in search for a better error metric.

Acknowledgments

This work was supported by the European Regional Development Fund (ERDF), Project "NTIS – New Technologies for the Information Society", European Centre of Excellence, CZ.1.05/1.1.00/02.0090, and by the UWB Grant SGS-2013-029 Advanced Computer and Information Systems (Pokročilé výpočetní a informační systémy).

References

[1] P. Alliez, M. Desbrun, Valence-driven connectivity encoding for 3d meshes, Comput. Graph. Forum 20 (3) (2001) 480–489.
[2] D. Chen, D. Cohen-Or, O. Sorkine, S. Toledo, Algebraic analysis of high-pass quantization, ACM Trans. Graph. 24 (4) (2005) 1259–1282.
[3] C. Courbet, C. Hudelot, Taylor prediction for mesh geometry compression, Comput. Graph. Forum 30 (1) (2010) 139–151.
[4] M. Corsini, M.C. Larabi, G. Lavoué, O. Petřík, L. Váša, K. Wang, Perceptual metrics for static and dynamic triangle meshes, Comput. Graph. Forum 32 (1) (2013) 101–125.
[5] C. Gotsman, On the optimality of valence-based connectivity coding, Comput. Graph. Forum 22 (1) (2003) 99–102.
[6] Z. Karni, C. Gotsman, Spectral compression of mesh geometry, in: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, 2000, pp. 279–286.
[7] F. Kälberer, K. Polthier, U. Reitebuch, M. Wardetzky, Freelence – coding with free valences, Comput. Graph. Forum 24 (3) (2005) 469–478.
[8] H. Lee, P. Alliez, M. Desbrun, Angle-analyzer: a triangle-quad mesh codec, Comput. Graph. Forum 21 (3) (2002).
[9] G. Lavoué, A multiscale metric for 3d mesh visual quality assessment, Comput. Graph. Forum 30 (5) (2011) 1427–1437.
[10] D. Marpe, H. Schwarz, G. BlÃĎttermann, G. Heising, T. Wieg, Context-based adaptive binary arithmetic coding in the h.264/avc video compression standard, IEEE Trans. Circ. Syst. Video Techn. 13 (2003) 620–636.

[11] J. Rossignac, Edgebreaker: connectivity compression for triangle meshes, IEEE Trans. Visual. Comput. Graph. 5 (1999) 47–61.
[12] O. Sorkine, D. Cohen-Or, D. Irony, S. Toledo, Geometry-aware bases for shape approximation, IEEE Trans. Visual. Comput. Graph. 11 (2) (2005) 171–180.
[13] O. Sorkine, D. Cohen-Or, S. Toledo, High-pass quantization for mesh encoding, in: Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, 2003, pp. 42–51.
[14] C. Touma, C. Gotsman, Triangle mesh compression, Graph. Interface 98 (1998) 26–34.
[15] G. Taubin, J. Rossignac, Geometric compression through topological surgery, ACM Trans. Graph. 17 (2) (1998) 84–115.
[16] L. Váša, G. Brunnett, Exploiting connectivity to improve the tangential part of geometry prediction, IEEE Trans. Visual. Comput. Graph. 19 (9) (2013) 1467–1475.
[17] S. Valette, R. Prost, Wavelet-based progressive compression scheme for triangle meshes: Wavemesh, IEEE Trans. Visual. Comput. Graph. 10 (2) (2004) 123–129.
[18] L. Váša, J. Rus, Dihedral angle mesh error: a fast perception correlated distortion measure for fixed connectivity triangle meshes, Comput. Graph. Forum 31 (5) (2012) 1715–1724.
[19] K. Wang, F. Torkhani, A. Montanvert, A fast roughness-based approach to the assessment of 3d mesh visual quality, Comput. Graph. 36 (7) (2012) 808–818.