DOI: 10.1111/cgf.12304
EUROGRAPHICS 2014 / B. Lévy and J. Kautz
(Guest Editors)

Volume 33 (2014), Number 2

Compressing dynamic meshes with geometric Laplacians

L. Váša[1], S. Marras[2], K. Hormann[2], G. Brunnett[1]

[1]Technical University Chemnitz, Chemnitz, Germany
[2]Università della Svizzera italiana, Lugano, Switzerland

Abstract

This paper addresses the problem of representing dynamic 3D meshes in a compact way, so that they can be stored and transmitted efficiently. We focus on sequences of triangle meshes with shared connectivity, avoiding the necessity of having a skinning structure. Our method first computes an average mesh of the whole sequence in edge shape space. A discrete geometric Laplacian of this average surface is then used to encode the coefficients that describe the trajectories of the mesh vertices. Optionally, a novel spatio-temporal predictor may be applied to the trajectories to further improve the compression rate. We demonstrate that our approach outperforms the current state of the art in terms of low data rate at a given perceived distortion, as measured by the STED and KG error metrics.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Animation—Image Processing and Computer Vision [I.4.2]: Compression (Coding)—Approximate methods

1. Introduction

The task of mesh compression is to find a compact binary representation of 3D models, easy to store and to transmit due to its small size, which allows reconstructing the mesh at the decoder side. In many applications the reconstruction does not necessarily have to be exactly equal to the input data, that is, some loss of precision is acceptable.

While some applications require the maximum dislocation of the vertices to be less than some given threshold, in many other applications it is the *perceived amount of distortion* that should be limited. Recent research has proven that these criteria are fundamentally different, and while vertex dislocations are traditionally used as a measure of mesh distortion, it is probably different features, such as dihedral angles, local curvature values, or edge lengths, that are important for distortion perception. Preserving these quantities, however, requires approaches different from those of traditional mesh compression [CLL*13].

One of the first attempts at perception-oriented mesh compression is high-pass encoding [SCOT03], which builds on the idea of transmitting the so-called *differential coordinates*, obtained by applying a combinatorial Laplacian to the coordinate function and reconstructing the coordinates by inverting the (extended) Laplacian. This approach succeeds at capturing local relations of vertices and usually outperforms other approaches, such as parallelogram prediction [TG98], when the mesh distortion is evaluated using some perception-based error measure.

At the same time, high-pass encoding can be interpreted as a particular prediction technique, where each vertex is being predicted at the centre of mass of its neighbouring vertices. The accuracy of the prediction is directly linked to the *entropy* of the residuals, that is, the differential coordinates. The accuracy of Laplacian-based prediction is usually slightly better than the accuracy of the parallelogram predictor. The residual entropy can be attributed to two sources in this case: the normal component of the Laplacian, which can be interpreted as the discrete mean curvature at each vertex, and the tangential component of the Laplacian, which can be interpreted as a sampling offset to each vertex.

Our observation is that the second component can be partially removed and hence the residual entropy be reduced, if a geometric Laplacian is used instead of the purely combinatorial operator. Popular choices of such geometric Laplacians include the cotangent Laplacian and the mean value (MV) Laplacian, which stems from the concept of mean value coordinates [Flo03]. Unfortunately, reconstructing such geometric Laplacians requires the availability of the geometry of the mesh, or at least its edge lengths, at the decoder, and it turns out that transmitting this information is as expensive as transmitting the mesh itself.

While compression based on geometric Laplacians is therefore not suitable for single meshes, it turns out to be beneficial for compressing dynamic meshes. Our main idea is to first build an *average* mesh of all the frames in the sequence, and we found that this is best done using the *edge shape space* [WDAH10]. We then compute the geometric Laplacian of this average mesh and encode the given mesh sequence in a PCA-reduced basis. The rationale behind this approach is that the average mesh contains the sampling information shared by all the other frames. For the data sets that we consider, we achieve a data rate reduction of 10–40%, compared to [VP11], keeping the same amount of distortion as measured by the STED error metric [VS11].

Main contributions

The main contributions of our work include:

- investigating and reducing the geometric redundancy in dynamic meshes by using discrete Laplace operators;
- designing a novel prediction technique that exploits spatial and temporal coherence of differential coordinates.

While the algorithm is based on similar ideas as [VS07] and [VP11], a significant gain in the compression performance is due to both the proposed application of the geometric Laplacian and the mesh averaging technique. The geometric Laplacian substantially reduces the redundancy in differential coordinates over the sequence. However, in order to build the geometric Laplacian, our technique requires to explicitly encode the average mesh, while [VP11] does not require to store any additional data, since it employs a purely combinatorial Laplacian that can be recovered from connectivity only. A further improvement in the compression performance is achieved by means of the proposed differential coordinates predictor. The predictor, while similar to other general techniques, is also novel in terms of formulation and application to the framework, and can potentially be applied to similar approaches, such as [VP11], to improve their performance.

The rest of the paper is organized as follows. After a brief overview of the most relevant work in the field of dynamic mesh compression (Section 2), the overview of the proposed approach is sketched in Section 3. Each step of the algorithm is then described in more detail. Section 4 describes how principal component analysis is used to reduce the dimensionality of the problem, Section 5 focusses on the computation of the average mesh, and Section 6 explains the usage of geometric Laplacians for the encoding. Section 7 describes the novel predictor for delta trajectories and the results of our experiments are presented in Section 8. We conclude by discussing the limitations of our approach in Section 9.

2. Related work

The aim of compressing an animated mesh sequence is to reduce the amount of space needed to store or to transmit all frames of the sequence. In principle, it is possible to simply apply one of the common techniques for the compression of static geometry (see [PKJK05] for a survey) and compress each frame separately. However, most of the static techniques exploit only *spatial coherence* of the shape, while also the *temporal coherence* between frames should be exploited in case of an animated sequence.

The most common way to reduce the data rate is to encode only a subset of the data and use them to predict the missing information, storing only the correction of the prediction. In case of mesh sequences, a first reference frame is usually compressed using a static compression technique, while the following frames are reconstructed by predicting the position of each vertex from both the spatial positions of the already decoded neighbouring vertices and the position of the same vertex in previous frames, achieving a full resolution reconstructed sequence with possible loss of precision due to the quantization of the correction vectors. This principle has been exploited, for example, in the MPEG-4 Animation Framework eXtension [BSJ04, JKJ*04]. Most techniques take advantage of the assumption of local rigidity of motion, that is, the assumption that vertices, which are close to each other, are characterized by the same motion. Rossignac et al. [RSS01] compute the motion of a vertex as the average motion of its neighbours, while Stefanoski and Ostermann [SO06] compress consecutive frames assuming that most of the dihedral angles do not significantly change. Other techniques [MSK*05, MSKW06, ZO04, ZO05] combine the same assumption with shape partitioning.

Instead of focussing on spatial and temporal coherence, a number of techniques achieve a significant reduction in data rate by exploring the data and finding the most important component via *principal component analysis* (PCA) of the data. The original approach by Alexa and Müller [AM00] uses PCA to find a set of *key-frames* to form a basis of the sequence. Each frame is encoded as a linear combination of these basis vectors, which may not correspond to any real frame. The approach was later extended by Karni and Gotsman [KG04] by introducing a linear predictor to estimate the weights of the linear combination and by Lee et al. [LKT*07], who describe how to determine the optimal number of key-frames. These approaches aim to find and encode a collection of $\mathbb{R}^{3n}$ vectors (n being the number of vertices) plus a number of coefficients for each frame of the sequence. The disadvantage is that the number of basis vectors may be quite high, and in case of meshes with a large number of vertices the approach is computationally inefficient and may even fail to capture the optimal basis of the sequence.

Several variations of the approach have been proposed. In [SLKO07] a multi-resolution PCA-based approach is presented. In [SSK05] and then in [AS07] the vertices are clustered according to their motion and then PCA is performed on each cluster, while in [LCS13] the frames are

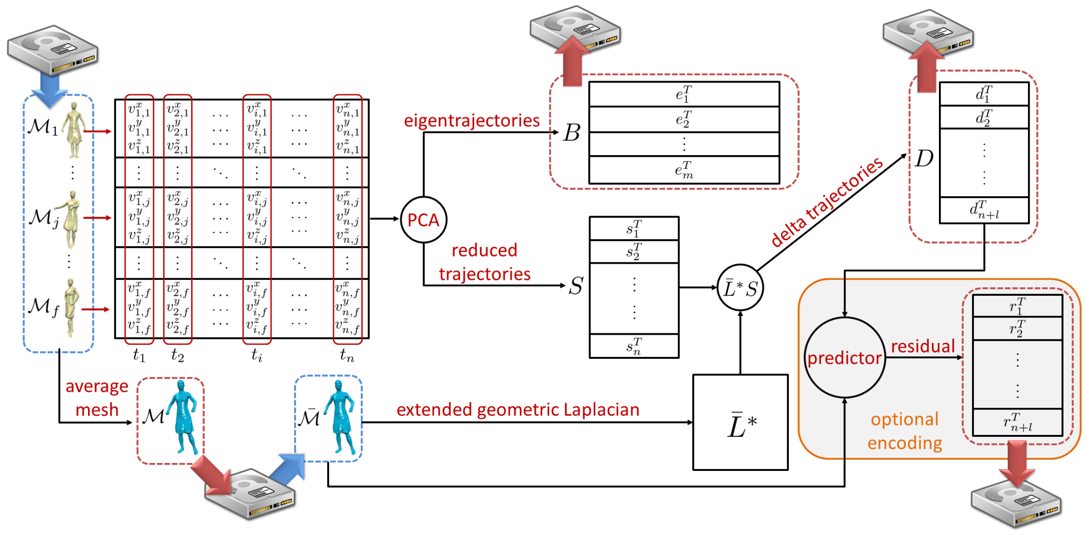

Figure 1: *Flowchart of the proposed compression algorithm (encoder). The predictor is shown in detail in Figure 5.*

temporally clustered and PCA is performed on each subsequence. However, the best reduction is achieved by the CODDYAC algorithm [VS07]. In this approach, instead of performing PCA on the geometry to find the key-frames, PCA is applied to the *trajectories* of the vertices in order to find a minimal number of significant trajectories characterizing the motion of the shape over the sequence. Each vertex trajectory is expressed as a combination of the basis trajectories and the algorithm encodes only the basis trajectory vectors and the coefficients. The same approach has been revised and improved in [VS09], proposing an alternative approach to compress the basis vectors named COBRA, and in [VP11], where the CODDYAC algorithm is combined with the Laplacian encoding technique by Sorkine et al. [SCOT03] to achieve the best compression rate so far with respect to a perceptual metric. Our proposed algorithm builds upon the CODDYAC approach and exploits the benefits of including geometric information in the Laplacian encoding.

3. Proposed algorithm overview

This section briefly sketches the algorithm, which is also summarized visually in Figures 1 and 2. In the following, symbols with a hat denote predictions of corresponding values, while symbols with a bar denote the corresponding values as reconstructed by the decoder, that is, different from the original values due to quantization and other sources of loss of precision.

We assume that our input is a sequence of f triangle meshes (frames) $\mathcal{M}_1, \ldots, \mathcal{M}_f$ characterized by the same underlying connectivity. We assume that the connectivity is encoded once, using any state-of-the-art algorithm. The position of the i-th vertex in the j-th frame is given by a vector of coordinates $v_{i,j} = (v^x_{i,j}, v^y_{i,j}, v^z_{i,j}), i = 1, \ldots, n, j = 1, \ldots, f$.

The proposed compression algorithm consists of several steps which are described in detail in the following sections. First, the dimensionality of the input data is reduced using the trajectory space PCA approach of the CODDYAC algorithm. After this step, the trajectory of each vertex is described by a vector of length m (user specified parameter). These vectors are then interpreted as m independent scalar-valued functions defined on the vertices of the mesh.

An average mesh $\mathcal{M}$ is computed for the input sequence in the so-called edge shape space [WDAH10] to avoid the usual pitfalls of mesh interpolation and averaging. This average mesh is encoded once using some standard approach for static mesh compression. It allows building a geometric discrete Laplace operator in the form of a matrix $\bar{L} \in \mathbb{R}^{n \times n}$ at both the encoder and the decoder.

The separate functions describing vertex trajectories are transformed by applying a discrete geometric Laplacian operator, yielding the so-called *delta trajectories*, which have a significantly smaller entropy than the original trajectory vectors. The decoder subsequently reverts this transformation by inverting the Laplacian matrix and reconstructs the trajectory vectors from the delta trajectories.

As an optional step, it is possible to compute the delta coordinates of the average mesh as well and use them to predict the delta trajectories.

4. Principal component analysis

The first step of the algorithm reduces the dimensionality of the data by using PCA. We follow the trajectory-based

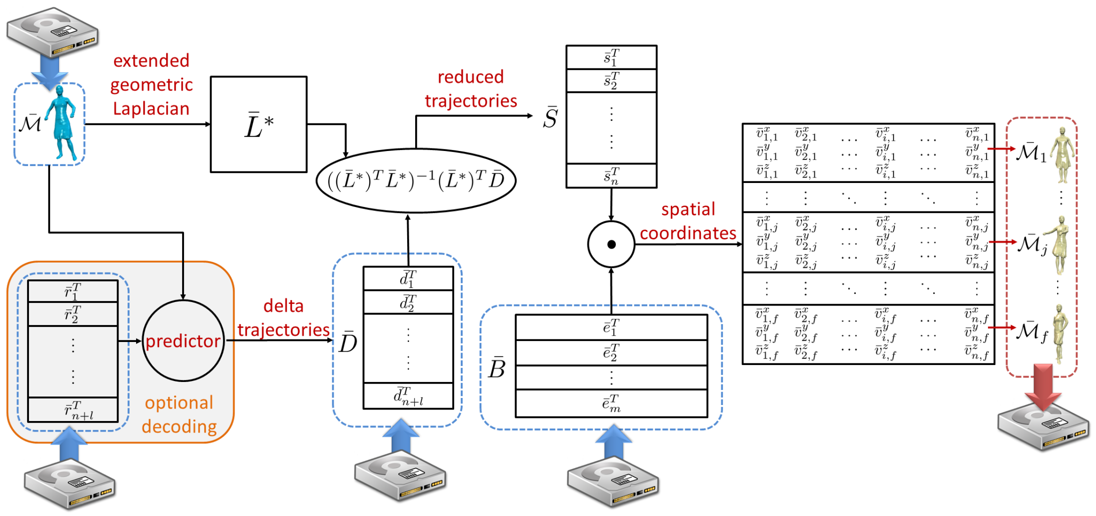

Figure 2: *Flowchart of the proposed compression algorithm (decoder). The predictor is shown in detail in Figure 5.*

approach of the CODDYAC algorithm [VS07]. First, the trajectory of each vertex is expressed as a column vector $t_i = (v^x_{i,1}, v^y_{i,1}, v^z_{i,1}, v^x_{i,2}, v^y_{i,2}, v^z_{i,2}, \ldots, v^x_{i,f}, v^y_{i,f}, v^z_{i,f})^T \in \mathbb{R}^{3f}$. An optimized basis of the space of trajectories is then found using the eigenvalue decomposition of the autocorrelation matrix of the trajectory vectors. This basis is given by the orthonormal *eigentrajectories* $e_i \in \mathbb{R}^{3f}$, $i = 1, \ldots, 3f$ and uses the average trajectory $t = (t_1 + \cdots + t_n)/n \in \mathbb{R}^{3f}$ as the origin. Collecting the m most important eigentrajectories in the matrix $B = (e_1, e_2, \ldots, e_m)^T \in \mathbb{R}^{m \times 3f}$, the trajectory of the i-th vertex is then described by the shorter column vector $s_i = B(t_i - t) \in \mathbb{R}^m$. Usually, the number of used basis vectors m is much smaller than the number of frames, allowing for a substantial reduction of the data rate.

To encode the data, it is necessary to encode the matrix B, the average trajectory t, and the vectors s_i. We use the COBRA algorithm [VS09] to encode B, while the rest of the paper deals with the encoding of the vectors s_i. During decoding, each trajectory t_i is reconstructed with some loss of precision as $\bar{t}_i = \bar{B}^T \bar{s}_i + \bar{t}$.

5. Computing the average mesh

An *average mesh*, which captures the common inter-vertex relationships, is used in our algorithm. The easiest way to compute it is to align all the frames, for example using an iterative closest point based approach, and then to compute the average position of each vertex. This approach, although computationally inexpensive, may lead to unsatisfactory results (see Figure 3). The average shape may exhibit artefacts, such as shrinking effects and self-intersections, leading to an invalid and visually implausible configuration of the mesh.

To overcome these limitations, we compute the average

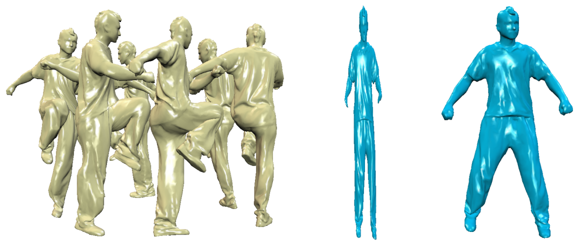

Figure 3: *The choice of the shape space severely affects the computation of the average mesh. In the* March 2 *data set* (a)*, the mesh describes a circular trajectory. A simple average on vertex coordinates results in a mesh where the shrinking effect is clearly visible* (b)*. Computing the average shape in edge space* (c) *avoids this effect and leads to a physically and visually plausible average mesh which captures the real structure of the deforming shape.*

mesh in the *edge shape space* [WDAH10], where each frame is expressed in terms of edge coordinates, that is, by *edge lengths* and *dihedral angles* between adjacent faces. For each edge, we compute its average length and its average dihedral angle over the whole sequence. However, it is not guaranteed that these values form a valid mesh configuration. Therefore, we project the average mesh back to the original vertex shape space by solving a non-linear optimization problem that searches for the mesh whose edge coordinates are *as close as possible* to the prescribed values. The problem can be tackled in a hierarchical fashion [WDAH10] or by iteratively solving a global non-linear system [FB11]. The resulting shape is finally placed in the original $\mathbb{R}^3$ space by tracking the trajectory of a reference face and computing its average position over time. In our experiments, we employ the technique presented in [WDAH10] to compute the average mesh.

The average mesh $\mathcal{M}$ is then compressed using any of the common techniques for static mesh encoding. In our implementation we use the high-pass encoding by Sorkine et al. [SCOT03].

6. Discrete geometric Laplace operator

In high-pass encoding of meshes [SCOT03] and in Laplacian-based encoding of dynamic meshes [VP11], a discrete combinatorial Laplace operator is applied to data associated with the vertices of a triangle mesh. Two variants of the operator are used respectively: the Kirchhoff Laplacian and the Tutte Laplacian. As argued in [VP11], the Tutte Laplacian is better suited for mesh compression, since it corresponds to uniform quantization of vertex-associated data (see [WMKG07, Zha04] for more details on discrete Laplacians).

Using the Tutte Laplacian, the Laplace transformation is equivalent to encoding the difference between the actual position of a vertex and the average of its one-ring neighbours. These difference vectors are usually referred to as *differential coordinates*. The original coordinates can be reconstructed from them if at least one so-called *anchor point* is explicitly stored for each connected component of the mesh.

Alternatively, the Laplace operator applied to the coordinate function can also be interpreted as the *mean curvature normal*. However, this interpretation is only valid if the operator possesses the linear preservation property, that is, if it is zero for flat neighbourhoods. Unfortunately, this property does not hold for any combinatorial Laplacian, where the result of the operator becomes a combination (vector sum) of the mean curvature normal (curvature component) and a tangential shift (sampling component) of the given vertex with respect to the centre of mass of its neighbours. In turn, the presence of the tangential part negatively affects the entropy of the delta coordinates.

The problem is even more pronounced in the case of dynamic meshes, where the tangential component is commonly very similar in all frames of the sequence. Even after dimensionality reduction, there remains a substantial amount of redundant information that is contained in the components of the vectors that describe the vertex trajectories. This tangential information is then transmitted multiple times.

Unlike in static mesh encoding, the task of dynamic mesh compression allows to distribute the cost of storing the information needed for constructing a geometric Laplacian over all meshes in the sequence. Having the average mesh computed and transmitted, as described in the previous section, allows us to construct, both during encoding and decoding, a geometric Laplacian $L \in \mathbb{R}^{n\times n}$ that is much closer to the linear preservation property than the combinatorial Laplacian. The entries of L are

$$L_{ii} = 1, \qquad L_{ij} = -w_{ij} \Big/ \sum_{k\neq i} w_{ik}, \quad i \neq j,$$

with weights w_{ij} computed according to the well-known cotangent formula [PP93] or the MV formula [Flo03].

We finally choose l *anchor points* [SCOT03] $v_{k_1}, \ldots, v_{k_l}$ and add l rows, where the i-th additional row contains only one non-zero element of value 1 in the k_i-th column. We refer to the resulting matrix $L^* \in \mathbb{R}^{(n+l)\times n}$ as the *extended Laplacian*. Note that L^* has full column rank and is hence invertible in the least squares sense. In our experiments, the MV Laplacian provides better results than the cotangent Laplacian. Section 8 presents a comparison between the two choices for the operator.

The encoder uses the extended geometric Laplacian to compute the delta trajectories. To this end, the vectors that describe the vertex trajectories in the reduced space are rearranged into a matrix $S = (s_1, \ldots, s_n)^T \in \mathbb{R}^{n\times m}$ and the matrix of delta trajectories $D = (d_1, \ldots, d_{n+l})^T \in \mathbb{R}^{(n+l)\times m}$ is computed as

$$D = \bar{L}^* S,$$

where $\bar{L}^*$ is the extended geometric Laplacian computed from the *decoded* average mesh, so that the encoder and the decoder work with the same values. The values in the matrix D have a much smaller entropy than the values in S and using the geometric Laplacian instead of the Tutte Laplacian provides a further entropy reduction. Finally, the values in the matrix D are uniformly quantized and encoded into the output stream using an arithmetic coder. Optionally, these values can also be predicted as described in Section 7.

The decoder solves the sparse and overdetermined system of linear equations $\bar{L}^* S = \bar{D}$ in the least squares sense, yielding the reconstructed trajectories

$$\bar{S} = \left((\bar{L}^*)^T \bar{L}^*\right)^{-1} (\bar{L}^*)^T \bar{D}.$$

A decomposition of the normal matrix can be precomputed in order to solve the problem efficiently, one column of $\bar{S}$ at the time.

7. Encoding delta trajectories

A further compression improvement can be achieved by efficiently encoding the matrix D. To this end, we interpret the rows $d_i^T \in \mathbb{R}^m$ of this matrix as vectors associated with each vertex and propose to encode them during a mesh traversal that follows the border expansion strategy of the Edgebreaker [RSS01] algorithm. In this scenario, a part of the mesh is known to the decoder (initially one triangle), and in each step a prediction of the data associated with a vertex that is adjacent to the known part is computed by the

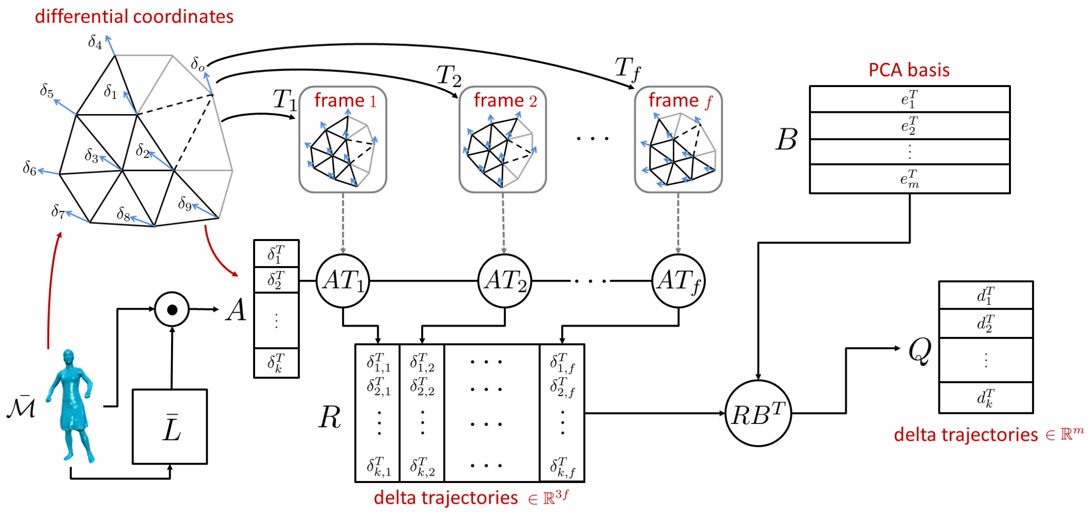

Figure 4: *Relationship between the data used for predicting delta trajectories. Note that this figure only illustrates the derivation of the predictor. The flowchart of the prediction itself is shown in Figure 5.*

encoder and the decoder. The encoder then encodes the so-called *residual*, that is, the difference between the actual data value and the prediction. In turn, the residual allows the decoder to correct the prediction and to expand the known part of the mesh by one vertex.

Suppose that the vectors $\bar{d}_i$ associated with several vertices are already decoded and thus known to the decoder. The task at hand is to predict a yet unknown vector d_o, associated with the vertex v_o, as accurately as possible. The key to this prediction is the availability of the average mesh $\bar{\mathcal{M}}$ at the decoder side, which allows us to construct the geometric Laplacian $\bar{L}$. This operator is used for transforming the trajectory vectors s_i, but it can also be applied to the coordinates of the average mesh itself in order to find its *differential coordinates*. There is a strong relation between these differential coordinates (known for each vertex) and the delta trajectories d_i, which we use for the prediction. Note that the rows of D, which correspond to anchor vertices, do not get predicted at all.

The differential coordinates of the average mesh are computed as

$$\Delta = \bar{L}\bar{C},$$

where $\bar{C} \in \mathbb{R}^{n\times 3}$ contains the vertex coordinates of the decoded average mesh $\bar{\mathcal{M}}$, with the x, y, and z coordinates stacked in the first, second, and third column, respectively. The rows $\delta_i^T \in \mathbb{R}^3$ of Δ are the differential coordinates of $\bar{\mathcal{M}}$ and can also be interpreted as vectors associated with each vertex. Each of these vectors is in fact the difference between the weighted average of the neighbours of v_i and v_i itself.

The key observation of our predictor is that the delta trajectory d_o, associated with vertex v_o, can be interpreted as differential coordinates of v_o in all the frames of the animation, stacked into a vector of length $3f$ and then projected onto the reduced basis. Following this interpretation, we now want to predict the differential coordinates in each frame. A naive approach would be to predict them as being equal to the corresponding differential coordinates in the average frame, and to then express this constant delta trajectory in the reduced basis. Such approach fails because the differential coordinates are rotation dependent, hence they change in each frame with the movement of the mesh.

Instead, we may assume that there exists for each vertex a series of f (almost) rigid transformations $T_i \in \mathbb{R}^{3\times 3}$, $i = 1,\dots,f$, which transform the differential coordinates from the average frame to the differential coordinates in each frame (see Figure 4). Stacking these transformations into a single matrix $T = (T_1,\dots,T_f) \in \mathbb{R}^{3\times 3f}$, the prediction of the delta trajectory in $\mathbb{R}^{3f}$ can be written as

$$p_o^T = \delta_o^T T,$$

where $\delta_o \in \mathbb{R}^3$ are the differential coordinates of v_o in $\bar{\mathcal{M}}$. Expressing the prediction in the reduced basis is then equivalent to multiplication by the basis matrix B^T,

$$\hat{d}_o^T = p_o^T B^T = \delta_o^T T B^T. \qquad (1)$$

Remember that in this and the following equations symbols with a hat denote the prediction of the corresponding value.

In order to evaluate the prediction, we now have to locally estimate the transformations that are stacked in T from the already decoded data. For an unknown vertex v_o, there are always at least three neighbouring vertices already known to the decoder (that is, their vectors $\bar{d}_i$ are available). However, for reasons of stability we use a larger number k of known

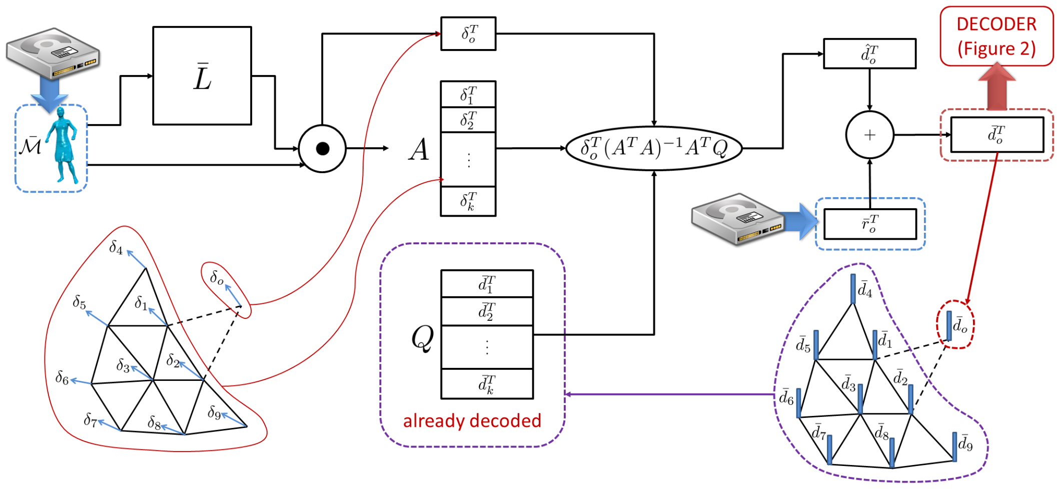

Figure 5: *One border expansion step, involving the prediction of d_o.*

neighbouring vertices, using a breadth-first search from the unknown vertex v_o. Collecting their differential coordinates in $\bar{\mathcal{M}}$ in the matrix $A = (\delta_1, \delta_2, \ldots, \delta_k)^T \in \mathbb{R}^{k\times 3}$, the delta trajectories of these vertices in $\mathbb{R}^{3f}$ are the rows of

$$R = AT,$$

and multiplying R with B^T transforms the data into the reduced basis,

$$Q = RB^T = ATB^T = (\bar{d}_1, \bar{d}_2, \ldots, \bar{d}_k)^T.$$

The matrix $Q \in \mathbb{R}^{k\times m}$ contains the decoded data associated with the neighbouring vertices and we can now solve for T in the least squares sense,

$$T = \left(A^TA\right)^{-1} A^TQB, \tag{2}$$

where we exploit the orthonormality of B, that is, $B^TB = I$. We finally plug the solution into the prediction (1) and obtain

$$\hat{d}_o^T = \delta_o^T \left(A^TA\right)^{-1} A^TQBB^T = \delta_o^T \left(A^TA\right)^{-1} A^TQ.$$

The prediction is evaluated by the decoder for each vertex during the traversal, as shown in Figure 5. Finally, instead of encoding the vector d_i itself, only the residual $r_i = d_i - \hat{d}_i$ is quantized and encoded. The decoder receives the residual $\bar{r}_i$ and reconstructs the vector $\bar{d}_i = \bar{r}_i + \hat{d}_i$. The more accurate this prediction is, the lower is the entropy of the residuals. Note that the actual prediction implementation involves neither the transformation matrices T_i nor the matrix R.

This kind of prediction has the advantage of requiring only the inversion of a single 3×3 matrix, and it works entirely in the reduced space of dimension m. Our experiments show that the system (2) tends to be unstable for k too small, and that the slightly higher computational cost of involving too many neighbouring vertices does not lead to further improvement in compression efficiency.

We tested the influence of k on the compression performance. Since the prediction has a negligible effect on the distortion (it is the quantization of the residuals that causes distortion, not the inaccuracy of the prediction), we focussed on the resulting data rates. For $k = 3$ we obtained data rates comparable or slightly worse than in the case without any prediction of delta coordinates. Increasing k leads to a reduction of the data rate in all datasets, with a rather flat minimum usually around $k = 30$. Thus, in our experiments we use $k = 15$ as a compromise between compression performance and computational cost.

8. Results

Traditionally, vertex-based error measures, such as root-mean-square or Karni-Gotsman (KG) error [KG00], are used for evaluating the amount of distortion caused by compression. Recently, however, researchers began to use *perceptual* metrics that capture the perceived distortion better than the traditional metrics. In the field of dynamic mesh compression, a perceptual metric STED has been proposed [VS11]. This metric is based on evaluating the changes in edge lengths, and it correlates well with the perceived distortion of dynamic meshes in user studies performed so far.

Additionally, traditional metrics tend to behave erratically when evaluating the results of Laplacian-based encoding. This behaviour stems from the relative instability of the result in locations far away from anchor points: even small changes in quantization may largely influence measures that observe the absolute dislocations of vertices. Figure 6 illustrates this effect. By focussing on local relations, the perceptual metrics provide a much more stable result that also matches the human perception. Therefore, we use the STED metric in our experiments.

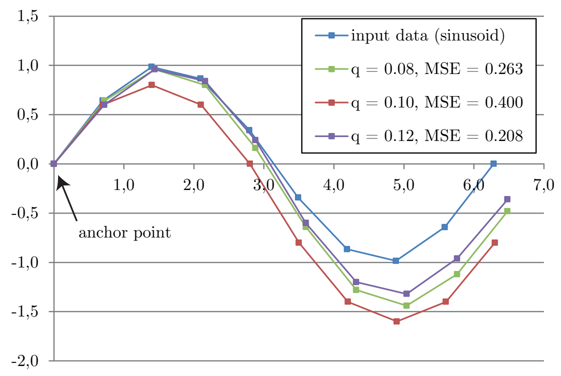

Figure 6: *Example reconstructions based on 2D Laplacians. The blue line represents the original 2D data points, the other lines are reconstructions from delta coordinates quantized using varying sizes of the quantization bin q. The mean squared error (MSE) behaves erratically with respect to the quantization, even though visually the reconstructions are similar.*

Figures 7 and 8 show typical rate-distortion (R-D) curves for our algorithm as well as the state of the art. We use a parameter optimization strategy similar to the one described in [VP11] to determine the quantization constants and the number of preserved basis vectors m. Using the MV Laplacian reduces the required data rate, measured in bits per frame per vertex (bpfv), by 22% on average, even without predicting delta trajectories. Including the prediction scheme for the latter further reduces the data rate by 30% on average with respect to compression with combinatorial Laplacians. For most of the measurements, we use the STED error metric [VS11], since it has been shown to correlate well with perceived distortion. The proposed algorithm, however, also brings improvement in compression performance with respect to traditional error measures, such as the KG error, as demonstrated in Figure 9. We refer to the accompanying video for a visual comparison of the resulting animations.

Table 1 shows the data rates, distortions, and residual entropies for various other models. It can be seen that our algorithm significantly improves the performance when there is a sufficient difference between the combinatorial and the geometric Laplacian of the given surface, that is, for meshes with non-uniform sampling. Meshes with more uniform sampling, such as the *dance* sequence, also benefit from using the geometric Laplacian, although the amount of improvement is smaller.

Finally, Table 2 shows the compression and decompression run times for our approach, compared to the run times of the scheme based on combinatorial Laplacians. If the delta trajectories are not predicted, the only source of slowdown is the computation and encoding of the average mesh, since the time required for solving the Laplacian system remains the same. The delta trajectory prediction requires additional time for evaluating the predictions, but the expense is rather small. It depends on the application, whether such additional expense is worth the reduced data rate, but the encoding and decoding times remain well within the limits of practical applicability.

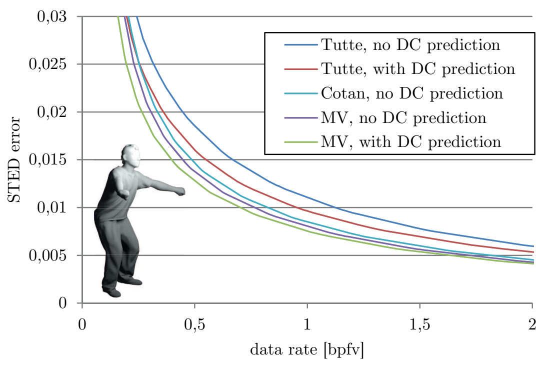

Figure 7: *R-D curves for the* squat2 *dataset [VBMP08], measured in STED error.*

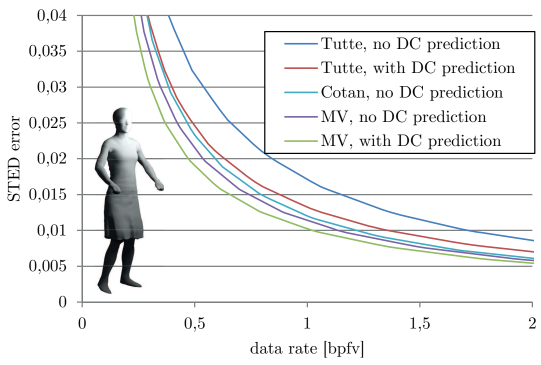

Figure 8: *R-D curves for the* samba *dataset [VBMP08], measured in STED error.*

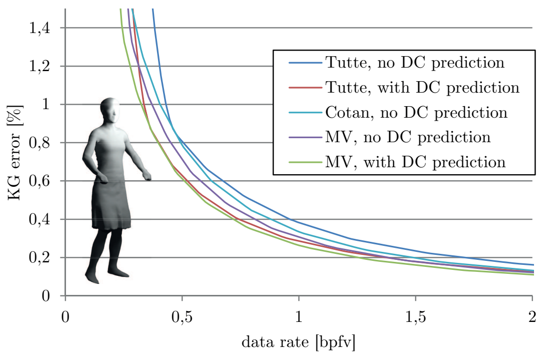

Figure 9: *R-D curves for the* samba *dataset [VBMP08], measured in KG error.*

		Tutte Laplacian		MV Laplacian				MV Laplacian + DC prediction			
model	dist.	rate	H	rate	Δrate	H	ΔH	rate	Δrate	H	ΔH
samba	0.02	0.837	2.497	0.535	**36.0%**	1.123	55.0%	0.470	**43.9%**	0.930	62.8%
9971V, 175F	0.01	1.722	2.893	1.119	**35.0%**	1.584	45.2%	1.019	**40.8%**	1.405	51.4%
squat2	0.02	0.435	3.099	0.293	**32.7%**	0.973	68.6%	0.253	**41.7%**	0.763	75.4%
10002V, 250F	0.01	1.122	3.011	0.816	**27.3%**	1.598	46.9%	0.751	**33.1%**	1.452	51.8%
humanoid	0.02	0.519	3.463	0.398	**23.2%**	1.942	43.9%	0.357	**31.2%**	1.680	51.5%
7646V, 154F	0.01	0.726	3.275	0.577	**20.5%**	2.120	35.3%	0.513	**29.3%**	1.882	42.6%
cowheavy	0.05	0.765	2.879	0.678	**11.4%**	2.046	28.9%	0.644	**15.8%**	1.780	38.2%
2904V, 204F	0.03	1.337	3.142	1.169	**12.5%**	2.397	23.7%	1.126	**15.8%**	2.204	29.9%
dance	0.02	0.470	2.154	0.397	**15.4%**	1.785	17.1%	0.344	**26.8%**	1.211	43.8%
7061V, 201F	0.01	0.688	2.858	0.606	**11.9%**	2.304	19.4%	0.527	**23.3%**	1.670	41.6%
march2	0.02	0.869	3.059	0.528	**39.2%**	1.066	65.2%	0.449	**48.3%**	0.908	70.3%
10002V, 250F	0.01	1.807	2.861	1.299	**28.1%**	1.447	49.4%	1.182	**34.6%**	1.342	53.1%
camel	0.03	1.071	1.674	0.978	**8.7%**	1.201	28.2%	0.861	**19.6%**	0.988	41.0%
21885V, 48F	0.01	2.572	2.318	2.296	**10.8%**	1.887	18.6%	2.089	**18.8%**	1.709	26,3%

Table 1: *Data rates and average entropies of residuals (H) for various models. The distortion is measured by the STED metric, the data rates are in bits per frame per vertex.*

	Tutte Laplacian		Average mesh			MV Laplacian without DC prediction		MV Laplacian with DC prediction	
model	encode	decode	build	encode	decode	encode	decode	encode	decode
samba	2521	790	1548	380	304	2543	815	2739	1013
squat2	3493	804	1581	216	173	3670	872	3680	1056
humanoid	1481	359	1192	179	143	1413	314	1671	454
cowheavy	1089	175	453	58	46	1031	141	1174	201
dance	2004	397	1132	138	110	1970	420	2031	460
march2	4044	1322	1599	215	172	4116	1305	4381	1546
camel	2240	1228	3531	482	386	2338	1159	2721	1580

Table 2: *Encoding/decoding times (in milliseconds) for various models. The total encoding time is the sum of times required for building and encoding the average mesh and the corresponding encoding time. Likewise, the total decoding time is the sum of the time required for decoding the average mesh and the decoding time.*

9. Conclusion

We presented a significant improvement to the CODDYAC algorithm that reduces the compression rate by removing the redundancy in a dynamic mesh. The choice of Laplacian weights which depend on the geometry of the shape, like those of the MV Laplacian, proves to be a better choice than uniform weights based on connectivity only, since it helps to reduce the tangential shift redundancy in the delta trajectories. The overhead introduced by computing and encoding the average mesh is balanced by a significantly better compression rate, while the computational cost of the encoding/decoding procedure is affected only marginally by the choice of weights. Encoding delta trajectories computed by geometric Laplace operators reduces the data rates by 22% on average in our experiments.

In addition to this improvement, we exploit the spatial and temporal coherence of vertex trajectories to develop a novel predictor approach, which, by assuming local rigidness of the motion, predicts the behaviour of the differential coordinates of a vertex by tracking the trajectories of the differential coordinates of the neighbouring vertices. This predictor allows us to reduce the bitrate by 30% on average in our experiments, while introducing only a negligible overhead in the process. The whole procedure is computationally cheap and outperforms state-of-the-art methods for compressing dynamic meshes.

A possible limitation arises when dealing with frames characterized by very regular spatial sampling. In these cases, the geometric and combinatorial Laplacians are quite similar and, as a consequence, the compression rates are almost equal, with the disadvantage that our algorithm requires additional storage to encode the average mesh. However, we observed that even in these cases a gain may be reached by employing the spatio-temporal predictor to compress the delta trajectories (see the last rows of Table 1).

Acknowledgements

This work was supported by the SNF under project number 200020-146764. The authors thank the anonymous reviewers for their valuable comments and suggestions.

References

[AM00] ALEXA M., MÜLLER W.: Representing animations by principal components. *Comput. Graph. Forum 19*, 3 (2000), 411–418. 2

[AS07] AMJOUN R., STRASSER W.: Efficient compression of 3D dynamic mesh sequences. *Journal of WSCG 15*, 1-3 (2007), 99–106. 2

[BSJ04] BOURGES-SEVENIER M., JANG E. S.: An introduction to the MPEG-4 animation framework extension. *IEEE Trans. Circuits Syst. Video Technol. 14*, 7 (July 2004), 928–936. 2

[CLL*13] CORSINI M., LARABI M. C., LAVOUÉ G., PETŘÍK O., VÁŠA L., WANG K.: Perceptual metrics for static and dynamic triangle meshes. *Comput. Graph. Forum 32*, 1 (2013), 101–125. 1

[FB11] FRÖHLICH S., BOTSCH M.: Example-driven deformations based on discrete shells. *Comput. Graph. Forum 30*, 8 (2011), 2246–2257. 4

[Flo03] FLOATER M. S.: Mean value coordinates. *Comput. Aided Geom. Des. 20*, 1 (Mar. 2003), 19–27. 1, 5

[JKJ*04] JANG E., KIM J., JUNG S. Y., HAN M.-J., WOO S. O., LEE S.-J.: Interpolator data compression for MPEG-4 animation. *IEEE Trans. Circuits Syst. Video Technol. 14*, 7 (2004), 989–1008. 2

[KG00] KARNI Z., GOTSMAN C.: Spectral compression of mesh geometry. In *Proceedings of the 2000 ACM SIGGRAPH conference on Computer graphics and interactive techniques* (2000), pp. 279–286. 7

[KG04] KARNI Z., GOTSMAN C.: Compression of soft-body animation sequences. *Comput. Graph. 28* (2004), 25–34. 2

[LCS13] LUO G., CORDIER F., SEO H.: Compression of 3D mesh sequences by temporal segmentation. *Comput. Animat. Virtual Worlds 24*, 3–4 (2013), 365–375. 2

[LKT*07] LEE P.-F., KAO C.-K., TSENG J.-L., JONG B.-S., LIN T.-W.: 3D animation compression using affine transformation matrix and principal component analysis. *IEICE - Trans. Inf. Syst. E90-D*, 7 (July 2007), 1073–1084. 2

[MSK*05] MÜLLER K., SMOLIC A., KAUTZNER M., EISERT P., WIEGAND T.: Predictive compression of dynamic 3D meshes. In *Proceedings of the 2005 IEEE International Conference on Image Processing* (2005), pp. 621–624. 2

[MSKW06] MÜLLER K., SMOLIC A., KAUTZNER M., WIEGAND T.: Rate-distortion optimization in dynamic mesh compression. *Signal Processing: Image Communication 21*, 9 (2006), 812–828. 2

[PKJK05] PENG J., KIM C.-S., JAY KUO C. C.: Technologies for 3D mesh compression: A survey. *Journal of Visual Communication and Image Representation 16*, 6 (Dec. 2005), 688–733. 2

[PP93] PINKALL U., POLTHIER K.: Computing discrete minimal surfaces and their conjugates. *Exp. Math. 2*, 1 (1993), 15–36. 5

[RSS01] ROSSIGNAC J., SAFONOVA A., SZYMCZAK A.: 3D compression made simple: Edgebreaker on a corner-table. In *Proceedings of the 2001 Shape Modeling International Conference* (May 2001), pp. 278–283. 2, 5

[SCOT03] SORKINE O., COHEN-OR D., TOLEDO S.: High-pass quantization for mesh encoding. In *Proceedings of SGP '03* (2003), pp. 42–51. 1, 3, 5

[SLKO07] STEFANOSKI N., LIU X., KLIE P., OSTERMANN J.: Scalable linear predictive coding of time-consistent 3D mesh sequences. In *Proceedings of the 2007 3DTV Conference* (2007), pp. 1–4. 2

[SO06] STEFANOSKI N., OSTERMANN J.: Connectivity-guided predictive compression of dynamic 3D meshes. In *Proceedings of the 2006 IEEE International Conference on Image Processing* (2006), pp. 2973–2976. 2

[SSK05] SATTLER M., SARLETTE R., KLEIN R.: Simple and efficient compression of animation sequences. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer animation* (2005), pp. 209–217. 2

[TG98] TOUMA C., GOTSMAN C.: Triangle mesh compression. In *Proceedings of the 1998 Graphics Interface Conference* (1998), pp. 26–34. 1

[VBMP08] VLASIC D., BARAN I., MATUSIK W., POPOVIĆ J.: Articulated mesh animation from multi-view silhouettes. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 97:1–97:9. 8

[VP11] VÁŠA L., PETŘÍK O.: Optimising perceived distortion in lossy encoding of dynamic meshes. *Comput. Graph. Forum 30*, 5 (2011), 1439–1449. 2, 3, 5, 8

[VS07] VÁŠA L., SKALA V.: CODDYAC: Connectivity driven dynamic mesh compression. In *Proceedings of the 2007 3DTV Conference* (2007), pp. 1–4. 2, 3, 4

[VS09] VÁŠA L., SKALA V.: COBRA: Compression of the basis for PCA represented animations. *Comput. Graph. Forum 28*, 6 (2009), 1529–1540. 3, 4

[VS11] VÁŠA L., SKALA V.: A perception correlated comparison method for dynamic meshes. *IEEE Trans. Vis. Comput. Graph. 17*, 2 (2011), 220–230. 2, 7, 8

[WDAH10] WINKLER T., DRIESEBERG J., ALEXA M., HORMANN K.: Multi-scale geometry interpolation. *Comput. Graph. Forum 29*, 2 (May 2010), 309–318. 2, 3, 4, 5

[WMKG07] WARDETZKY M., MATHUR S., KÄLBERER F., GRINSPUN E.: Discrete Laplace operators: no free lunch. In *Proceedings of SGP '07* (2007), pp. 33–37. 5

[Zha04] ZHANG H.: Discrete combinatorial Laplacian operators for digital geometry processing. In *in SIAM Conference on Geometric Design, 2004* (2004), pp. 575–592. 5

[ZO04] ZHANG J., OWEN C.: Octree-based animated geometry compression. In *Proceedings of the 2004 Data Compression Conference* (2004), pp. 508–517. 2

[ZO05] ZHANG J., OWEN C.: Hybrid coding for animated polygonal meshes: combining delta and octree. In *Proceedings of the 2005 International Conference on Information Technology: Coding and Computing* (2005), vol. 1, pp. 68–73. 2