DOI: 10.1111/j.1467-8659.2011.02018.x

Eurographics Symposium on Geometry Processing 2011
Mario Botsch and Scott Schaefer
(Guest Editors)

Volume 30 (2011), Number 5

Optimising Perceived Distortion in Lossy Encoding of Dynamic Meshes

L. Váša[1] and O. Petřík[1]

[1]Department of Computer Science and Engineering, University of West Bohemia, Czech Republic

Abstract

Development of geometry data compression techniques in the past years has been limited by the lack of a metric with proven correlation with human perception of mesh distortion. Many algorithms have been proposed, but usually the aim has been to minimise mean squared error, or some of its derivatives.
In the field of dynamic mesh compression, the situation has changed with the recent proposal of the STED metric, which has been shown to capture the human perception of mesh distortion much better than previous metrics. In this paper we show how existing algorithms can be steered to provide optimal results with respect to this metric, and we propose a novel dynamic mesh compression algorithm, based on trajectory space PCA and Laplacian coordinates, specifically designed to minimise the newly proposed STED error. Our experiments show that using the proposed algorithm, we were able to reduce the required data rate by up to 50% while preserving the introduced STED error.

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 goal of lossy compression is to encode input data into a file as small as possible, from which the data can be reconstructed with the smallest possible error. The data rate and resulting error are controlled by algorithm parameters, such as quantisation roughness. Advanced algorithms usually have multiple parameters, all of which have to be set accurately in order to obtain the best possible performance of the encoder.

An important aspect of lossy compression is the metric used to evaluate the distortion introduced by the compression. A straightforward sum of (squared) distances between original and decoded vertex positions is relevant in technical applications; however, it is more usual to employ lossless compression schemes when such precision is required. On the other hand, compression of audio data reached a breakthrough when perception-correlated metrics of distortion have been employed, and compression of images and video is currently in the process of adopting similar perception-related error measures [WLB04, WBSS04].

In geometry data compression, there are many compression algorithms, most of which focus on minimising the Mean Squared Error (MSE). Most researchers agree that this error measure does not capture human perception of distortion well, and some alternative metrics have been proposed. However, until recently there have been no rigorous experiments carried out to support any of the metrics.

With the proposal of the Spatio-Temporal Edge Difference (STED) error metric [VS11], the situation has changed, at least in the field of compression of dynamic meshes (i.e. animated models). Now we have a metric which captures the perceived distortion better than MSE, and we can focus on:

1. finding parameters of existing algorithms so that we obtain optimal performance with respect to STED error,
2. designing new algorithms specifically focused on minimising STED error,
3. improving the error measure: identifying its drawbacks by including the results of STED-optimised compression algorithms into future perception experiments.

In this paper we demonstrate that current compression algorithms such as Frame-based Animated Mesh Compression (FAMC) and Coddyac can be reconfigured to optimise the

STED error, and we propose a general framework based on the principle of equal slopes, which finds the optimal configurations much faster than exhaustive search. This principle is well-known to the compression community and is frequently used for rate-distortion optimisation with independent parameters. We have adapted this method to handle dependent compression parameters by using an iterative approach. Finding the optimal configurations allows us to perform a fair comparison of the performance of the state-of-the-art algorithms.

Subsequently, we propose a new algorithm based on describing the trajectory of each vertex by a feature vector (trajectory-space Principal Component Analysis - PCA) and encoding these vectors using the notion of Laplacian coordinates, i.e. encoding each value in the context of its neighbouring values, which nicely matches the aims of the STED measure.

The rest of the paper is organised as follows: In section 2, we briefly describe the previous works in the field of dynamic mesh compression with special focus on the state-of-the-art algorithms FAMC and Coddyac and the principle of the STED error measure. In section 3, we show how to reconfigure the existing algorithms and evaluate their performance with respect to STED. Identifying a weak spot in the existing approaches, we propose a new method in section 4, and show its results in section 5. Finally, conclusions and future research topics are presented in section 6.

2. Related work

Many algorithms have been proposed in the past for compressing static or dynamic triangular meshes. In this paper we focus on the case of dynamic meshes. A dynamic mesh is a sequence of static meshes (frames) with common connectivity, while the vertex positions change in each frame.

The task of lossy compression of such data seeks to find a compact binary representation from which the original can be roughly reconstructed, i.e. we do not require the decompressed vertices to lie at exactly the same positions as in the original data. However, we aim to minimise the difference. This topic has been developing rapidly in recent years. The first attempt is attributed to Lengyel's algorithm [Len99], which divides the mesh into parts of rigid movement.

Subsequently, a multitude of algorithms followed, based on various techniques, such as predictive encoding [IR03, SO06], wavelet decomposition [PA05] or shape-space PCA [AM00]. All of these techniques have focused on minimising the MSE error.

Another approach to the problem has been proposed by Zhang and Owen [ZO04], and further developed by Müller *et al.* [MSK*06]. The authors proposed representing each frame of the animation by a set of motion vectors, which describe the motion of each vertex with respect to the previous frame. The space of each frame is subdivided using an octree structure, and motion vectors in each cell of the octree are approximated by a linear interpolation of motion vectors associated with corners of the cell. If the approximation turns out to be too imprecise, then the octree cell is further subdivided.

Mamou *et al.* [MZP06] have proposed a different approach, based on skinning [JT05]. Their algorithm first decomposes the mesh into clusters of similar motion, where each cluster is assigned a transformation for each frame. The transformation is optimised so that it describes the motion of the cluster between the first and current frame. Each vertex has assigned a vector of weights, which is used to combine the cluster transformation matrices to obtain a prediction of vertex position in each frame. Finally, prediction residuals are encoded either directly, or using Discrete Cosine Transform (DCT) or PCA.

Applying PCA orthogonally, i.e. in the space of trajectories rather than shapes, Váša and Skala have proposed the Coddyac algorithm [VS07]. The approach finds a new decorrelated basis for the space of vertex trajectories, and subsequently the trajectory of each vertex is expressed in the new basis, allowing a radical dimensionality reduction. The same authors have also proposed an efficient algorithm for encoding the PCA basis [VS09]. Using the proposed techniques, they have achieved the best compression ratios so far.

The previously mentioned algorithms have always presented the results in the form of rate-distortion curves. The most frequently used metric for error evaluation is the Mean Squared Error (MSE), computed as average squared size of vertex dislocation caused by compression. Some papers have used the metric proposed in [KG04], called KG-error, which is in fact a normalised version of MSE, while others have used various versions of the Hausdorff distance, which is related to the magnitude of the largest distortion in the mesh. Finally, some of the papers have used the $D_a - error$, proposed in [JKJ*04], which measures the area of virtual ribbons created by error vectors in time.

None of these metrics has been tested for correlation with human perception, and in fact, Váša and Skala [VS11] have shown that none of the metrics actually correlates with results of subjective testing. In the case of minimising the perceived error, it is therefore incorrect to minimise any of the previously mentioned metrics. On the other hand, the newly proposed metric STED [VS11] has been shown to correlate with subjective results much better (Pearson correlation coefficient of 0.91-0.97 in the performed experiments).

2.1. STED error measure

This metric is based on measuring the change of edge lengths caused by the compression, and takes into account the temporal properties of the dynamic mesh by including virtual (temporal) edges connecting a vertex to it's position

in the next frame. The parameters of the metric have been set so that the metric results match the results of subjective experiments as closely as possible.

The metric first works with each frame separately, computing the spatial part of the error, called $STED_{spat}$. Having the original length of each edge e denoted $l(e)$, and the distorted length $\overline{l(e)}$, the algorithm computes the relative change of length for each edge $d(e) = (\overline{l(e)} - l(e))/l(e)$. Subsequently, the algorithm investigates the behaviour of this quantity in the topological neighbourhood of each vertex by computing standard deviation within each neighbourhood (assuming that high deviation relates to high perceived error, whereas low deviation indicates low perceived error). Finally, the deviations of $d(e)$ are averaged over all vertices in all the frames of the animation.

Subsequently, the algorithm computes the temporal part of the metric $STED_{temp}$ using a similar approach, only this time taking into account the virtual edges connecting positions of each vertex in subsequent frames. A relative change of the length of these edges is computed and averaged over all the temporal edges of all pairs of subsequent frames.

The overall error is evaluated as a hypotenuse of $STED_{spat}$ and $STED_{temp}$, using a weighting constant to relate the spatial and the temporal error. The algorithm has several parameters, such as the width of the used topological neighbourhood, or the aforementioned weighting constant. These parameters have been set empirically so that the results of the metric fit the results of a large subjective testing experiment conducted by the authors of the STED metric. For details see [VS11].

2.2. FAMC algorithm

Based on their skinning approach [MZP06], Mamou et al. presented a compression method called Frame-based Animated Mesh Coding [MZP*08], which was later standardised by the Motion Picture Experts Group as a part of the MPEG-4 standard [Int09]. The main idea of this approach is to divide the animated mesh into clusters of vertices, whose motion from the first frame to each of the succeeding frames can be described accurately using a single affine transform, and to only store this transform for each cluster and frame. In the beginning, the first frame of the animation is compressed using a static mesh compression scheme, such as 3DMC [Int01], or Touma and Gotsman [TG98]. Having F the number of frames of the dynamic mesh, a succession of global translation vectors $\tau_f,\ f = 1 \dots F$ is calculated by averaging the change in vertex position between the first and f-th frame over all vertices of the mesh. These vectors are then subtracted from the vertex positions.

The clustering process, which is performed next, is referred to as *Motion-based Segmentation* and comprises a series of half-edge collapse steps $hc(i,j) : v_i^* := v_i^* \cup v_j^* \cup \{v_j\}$, initially $v_i^* = \{\},\ i = 1 \dots V$, where V is the number of vertices. In every step, each edge (v_i, v_j) is assigned a cost Ω_{ij}:

$$\Omega_{ij} = \sum_{f=1}^{F} \left(\sum_{u \in \vartheta_{ij}} \left\| M_f^{ij} u_1 - u_f \right\|^2 \right), \tag{1}$$

$$\vartheta_{ij} = \{v_i, v_j\} \cup v_i^* \cup v_j^*, \tag{2}$$

$$M_f^{ij} = \arg\min_M \left(\sum_{u \in \vartheta_{ij}} \left\| M u_1 - u_f \right\|^2 \right) \tag{3}$$

The edge with the lowest cost is then collapsed to one of its endpoints and the process is repeated until the lowest cost exceeds a given threshold. In the end, each vertex v_k of the K remaining vertices defines a cluster $\pi_k = \{v_k\} \cup v_k^*$ and the transformation matrix for cluster π_k and frame f is calculated as:

$$M_f^{\pi_k} = \arg\min_M \left(\sum_{u \in \pi_k} \left\| M u_1 - u_f \right\|^2 \right) \tag{4}$$

By only storing one transformation matrix (12 coefficients) per cluster and frame, a significant reduction of data rate can be achieved. However, this process introduces a fairly large amount of error into the geometry of the mesh, especially at the cluster borders. To reduce this error, a technique named *Skinning-based Motion Compensation* is implemented. The predicted position $\hat{v}_f^i$ of i-th vertex in f-th frame is obtained using not only its own cluster's transform, but also using the transforms of neighbouring clusters. A vector of weights ω_i is assigned to each vertex, determining the combination ratio of transforms to be used to obtain the prediction of the position $\hat{v}_f^i$. Finally, a prediction residue is calculated and stored for each vertex and frame:

$$\varepsilon_f^i = v_f^i - \hat{v}_f^i \tag{5}$$

The most efficient version of the FAMC algorithm uses a Discrete Cosine Transform (DCT) for encoding of the transformation matrices as well as the global translations and prediction residues. All the DCT coefficients and the animation weights are uniformly quantised and entropy encoded using the CABAC [MWS03] encoder.

A separate input parameter is used to specify the coarseness of the quantisation process for each the transformed prediction residues, the global translations, the transform matrices and the animation weights. Yet another parameter controls the encoding of the first frame, which gives a total of five input parameters that have to be precisely set in order to obtain an optimal configuration. For more details on the FAMC compression algorithm, see [MZP06, Int09, MZP*08].

2.3. Coddyac algorithm

A different approach to compression is the Coddyac [VS07] algorithm. It is based on interpreting the input dynamic mesh

as a set of trajectories of individual vertices. The trajectory of the i-th vertex is described by a vector T_i of length $3F$, consisting of XYZ coordinates of the given vertex in all the frames. For dense meshes, the trajectory vectors of neighbouring vertices are likely to be similar to one another. In other words, the trajectory vectors are not distributed evenly in the space of dimension $3F$. Instead, they are roughly located in a subspace of much **lower dimension**.

This observation yields the first step of the Coddyac algorithm: finding this subspace and expressing the vertices in it. The Coddyac algorithm uses the PCA tool of linear algebra to find an optimised basis of the space, and each trajectory vector T_i is expressed in this basis by a *feature vector* C_i. The feature vectors are truncated to length B, where B is a user selected number of preserved basis vectors.

To store the dynamic mesh, it is necessary to store the selected subset of the basis (a matrix of size $3F \times B$), the feature vectors (matrix C of size $V \times B$) and the vector D representing the average trajectory. Details on how to efficiently encode the basis matrix can be found in [VS09].

The other key observation of the Coddyac algorithm is that the PCA step can be interpreted as a simple change of basis, and therefore it should not have any influence on the results of linear operators. This feature is employed for prediction of the values C_i^j at the decoder. In static mesh encoding, a very common prediction method is based on the parallelogram rule [TG98, ČS05]. The idea is that the mesh is traversed progressively by growing an area of processed vertices by adding one adjacent triangle (with one adjacent vertex) at a time [Ros99]. The newly added triangle together with its neighbouring known triangle form a quad q, illustrated in figure 1.

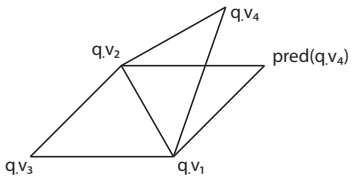

Figure 1: *Indices used to denote vertices of an expansion quad. Vertex v_4 is not known to the decoder.*

The XYZ coordinates of the new vertex $q.v_4$ are predicted to lie at the top of a projected parallelogram formed by the three known vertices $q.v_1$, $q.v_2$ and $q.v_3$. The coordinate prediction is then expressed as:

$$\begin{aligned} pred(v^X_{q.v_4}) &= \overline{v^X_{q.v_1}} + \overline{v^X_{q.v_2}} - \overline{v^X_{q.v_3}} \\ pred(v^Y_{q.v_4}) &= \overline{v^Y_{q.v_1}} + \overline{v^Y_{q.v_2}} - \overline{v^Y_{q.v_3}} \\ pred(v^Z_{q.v_4}) &= \overline{v^Z_{q.v_1}} + \overline{v^Z_{q.v_2}} - \overline{v^Z_{q.v_3}} \end{aligned} \quad (6)$$

where $\overline{v^X_{q.v_1}}$ denotes the X coordinate of the vertex $q.v_1$ as known to the decoder (which may be different from the original coordinate due to quantisation).

Since the feature vectors are in fact linearly transformed trajectory vectors, it is possible to use the same formula also for the elements of feature vectors:

$$pred(C^j_{q.v_4}) = \overline{C^j_{q.v_1}} + \overline{C^j_{q.v_2}} - \overline{C^j_{q.v_3}}, j = 1..B \quad (7)$$

The Coddyac algorithm traverses the mesh, adding one triangle at a time, performs the prediction according to equation (7) and transmits the quantised prediction residuals. Although any integer coding scheme can be used, it is reasonable to expect the Laplace distribution with zero mean of the prediction residuals, and therefore it is efficient to use an arithmetic entropy coder together with exp-Golomb binarisation.

3. Reconfiguring existing algorithms

Having a new error measure, a natural question to be asked is "which of the algorithms performs best with respect to the new metric". The authors of the STED metric have already shown that the new metric is fundamentally different from MSE and its derivatives, and therefore using it has an important impact on algorithm configuration – a configuration which is optimal with respect to MSE might be suboptimal with respect to STED.

This is also the reason why it is difficult to compare existing algorithms in the new metric: we cannot use the results reported in the respective papers, and even if we had the resulting meshes, we would not be able to provide a fair comparison, because the results were obtained with configurations optimised for MSE. Instead, we need an implementation of each algorithm, and we need to optimise their configurations for the STED error measure.

Unlike the usual optimisation problem, which focuses on finding extreme values of a single quantity defined on the input data, here we need to perform optimisation with respect to two quantities – bitrate r and distortion d. The task is trivial for compression algorithms with only one input parameter, or with input parameters that can be independently optimised (i.e. they have independent influence on the compression result). Unfortunately, this is not the case for most of the dynamic mesh compression algorithms.

An approach still commonly used among authors of dynamic mesh compression algorithms is an exhaustive search. In this case, a set of values is defined for each of the input parameters and the compression is then carried out with all possible combinations of these values. Finally, the combinations producing the best results are selected. While this method works for every compression algorithm and every distortion measure, it is very computationally expensive, especially with the growing number of optimised parameters. For this reason, we are proposing a new approach to the rate-distortion (RD) optimisation in dynamic mesh compression. The key task of this approach is to keep the number of compressor invocations as low as possible while making the

method as universal as possible in both the compression algorithm and the distortion measure used.

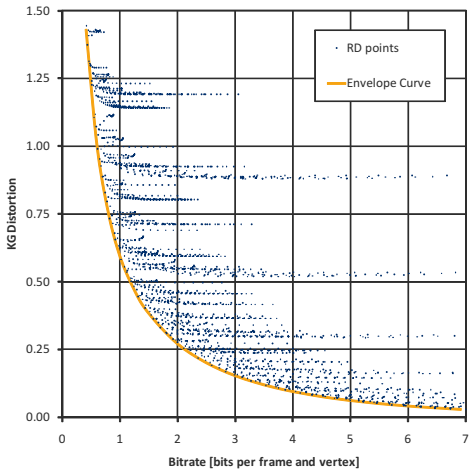

Figure 2: *RD points for various configurations used in an exhaustive search located above the envelope RD curve obtained through the proposed optimisation scheme using the Cow mesh sequence compressed by the Coddyac algorithm.*

We have based our optimisation method on the *Principle of Equal Slopes*. This principle exploits the shape of RD curves in order to find the optimal parameter values. For a dynamic mesh compression algorithm Γ with q input parameters and an input dynamic mesh μ, we can define an *RD chart* $H = \{\Gamma(P,\mu); \forall P\}$, which is the set of rate-distortion values for all possible parameter configurations $P = [p_1 \dots p_q]$ (assuming that p_i are real or integer numbers). This chart is confined at the bottom by an *envelope RD curve* O, which contains the optimal compression results (see figure 2). For a specified configuration $P^j = [p_1^j \dots p_q^j]$, an i-th *parameter RD curve* H_i^j contains results for all values of parameter p_i with the rest of the parameters fixed in P^j:

$$H_i^j = \{\Gamma(P_i^j,\mu); P_i^j = [p_1^j \dots p_{i-1}^j, p_i, p_{i+1}^j \dots p_q^j]; \forall p_i\} \quad (8)$$

Parameter RD curves are a subset of H, hence they are also lower-bound by curve O, and if they touch O, the point of touch is a result of an optimal configuration. Since they cannot intersect O, the parameter curves running through each such optimal point have the slopes of their tangents equal to the tangent slope of O in this point. Slope equality is a necessary condition for configuration optimality; however, in our experiments, where the parameter RD curves as well as the envelope RD curve have a convex and decreasing shape, it also works as a sufficient condition. Thus, whenever all the parameter curves running through a particular point have their slopes equal to each other in that point, the parameter configuration in this point is optimal. For illustration see figure 3.

Using the approach described above, we can perform an iterative parameter optimisation. Each iteration is initialised with an input parameter configuration. For the first iteration, this configuration is specified as a parameter of the optimisation process. An RD point is sampled in this configuration by invoking the compressor and two more points are sampled on each of the parameter RD curves, one to the "left" and one to the "right" of the current value of the parameter. From this set of points, the local behaviour of slope s_i of i-th parameter RD curve depending on parameter p_i can be approximated as:

$$\begin{aligned} s_i(p_i) &= s_i(p_i^{left}) + 2\frac{s_i(p_i^{right}) - s_i(p_i^{left})}{p_i^{right} - p_i^{left}}(p_i - p_i^{left}), \\ s_i(p_i^{left}) &= \frac{d^{current} - d(p_i^{left})}{r^{current} - r(p_i^{left})}, \\ s_i(p_i^{right}) &= \frac{d(p_i^{right}) - d^{current}}{r(p_i^{right}) - r^{current}}, \end{aligned} \quad (9)$$

and the local behaviour of rate and distortion can be approximated by relative increments:

$$\begin{aligned} r &= r^{current} + \sum_{i=1}^{q} \Delta r_i (p_i - p_i^{current}), \\ d &= d^{current} + \sum_{i=1}^{q} \Delta d_i (d_i - d_i^{current}), \\ \Delta r_i &= \frac{r(p_i^{right}) - r(p_i^{left})}{p_i^{right} - p_i^{left}}, \\ \Delta d_i &= \frac{d(p_i^{right}) - d(p_i^{left})}{p_i^{right} - p_i^{left}} \end{aligned} \quad (10)$$

The optimal configuration is then estimated by solving a system of equations, which consists of $q-1$ equations from the slope equality condition:

$$s_i(p_i) = s_{i+1}(p_{i+1}), \quad i = 1 \dots q-1 \quad (11)$$

and one equation defined by one of four additional constraints:

$$\begin{aligned} \text{slope:} \quad & s_1(p_1) = s^{target} \\ \text{rate:} \quad & \sum_{i=1}^{q} \Delta r_i (p_i - p_i^{current}) = r^{target} - r^{current} \\ \text{distortion:} \quad & \sum_{i=1}^{q} \Delta d_i (p_i - p_i^{current}) = d^{target} - d^{current} \\ i\text{-th parameter:} \quad & p_i = p_i^{target} \end{aligned} \quad (12)$$

Each constraint exactly determines the resulting configuration by specifying one of four properties of the result – either a target slope of the curves, a target bitrate or distortion, or a fixed value of one of the parameters.

By solving the system of equations defined above, a new

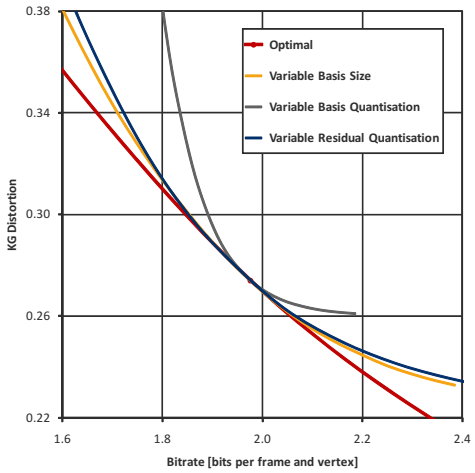

Figure 3: *Optimal RD-curve and three parameter RD-curves.*

(possibly optimal) configuration is obtained. Since we only use approximations of the parameter curves to define the equations, the result is likely to be suboptimal to some degree. Therefore, the process is iteratively repeated until convergence. An overall deviation from optimum is calculated in each iteration based on the slope differences and the additional constraint. Once this deviation falls under a specified threshold, the iteration process is stopped.

configuration				error	
QR	QB	BV	bpfv	STED	KG
0.265	18.01	74	0.50	1.09E-01	0.553
1.616	19.36	124	1.00	5.83E-02	0.269
2.465	20.28	166	1.49	3.84E-02	0.170
3.063	20.94	205	1.99	2.85E-02	0.121
3.553	21.48	238	2.48	2.22E-02	0.093
3.957	21.87	272	3.00	1.81E-02	0.074
4.289	22.17	301	3.49	1.53E-02	0.062
4.587	22.43	330	3.99	1.32E-02	0.053

Table 1: *Result of optimisation of KG error. The parameters: QR: quantisation of feature vector residuals, QB: quantisation of PCA basis, BV: number of used basis vectors.*

configuration				error	
QR	QB	BV	bpfv	STED	KG
2.424	16.89	36	0.50	4.71E-02	0.991
3.292	18.32	70	0.99	3.02E-02	0.474
3.863	19.11	102	1.49	2.20E-02	0.283
4.424	19.83	126	1.99	1.71E-02	0.204
4.736	20.33	154	2.46	1.41E-02	0.149
5.136	20.81	178	3.00	1.18E-02	0.117
5.429	21.19	200	3.48	1.02E-02	0.096
5.688	21.55	223	3.98	8.96E-03	0.080

Table 2: *Result of optimisation of STED error. The parameters: QR: quantisation of feature vector residuals, QB: quantisation of PCA basis, BV: number of used basis vectors.*

Using this process we are able to obtain rate-distortion curves for various error metrics using a single optimisation method. Tables 1 and 2 show the RD performance of the Coddyac algorithm with the jump sequence, together with the configurations used, for both STED error and KG error optimisation, on the same range of data rates. Notice that the configurations for matching bitrates differ between the target error metrics, and also the performance in both metrics changes considerably depending on the used target metric.

As mentioned before, one of the aims of the proposed optimisation method is to reduce the number of compressor invocations. Experiments show that with our method, the number of invocations depends linearly on the number of parameters, while the previously used exhaustive search required an exponential number of compressor invocations. For example, to produce the eight optimal configurations shown in table 1, the optimiser needed to run the compression module 724 times in order to obtain the configurations at specified bitrates. An exhaustive search with a precision of 2 vectors in the number of basis vectors and 0.1 bits in both the quantisation parameters would require about 267,000 compressor runs and could not be controlled to give results at particular bitrates.

4. Laplacian coordinates encoding

The results presented so far show that optimising STED error instead of MSE for the Coddyac algorithm leads to configurations with finer quantisation of feature vector residuals and lower numbers of basis vectors. In order to improve the performance, we should therefore focus on the encoding of feature vectors. We need to either lower the required data rate, or encode this data so that the character of the introduced error is more acceptable for the STED metric.

The character of STED error indicates that relative position of a vertex with respect to its neighbours is more important than its absolute position in space. It is therefore natural that if we encode relative positions rather than absolute ones, we will obtain a better reconstruction. For more support on why we expect relative position encoding to perform better, refer to Appendix A.

One way to express relative positions of vertices is to transform each vertex v into a coordinate system with origin located in the average position of the neighbourhood of the vertex:

$$\widehat{v_i} = v_i - \frac{1}{\|n(i)\|} \sum_{j \in n(i)} v_j \qquad (13)$$

The result of such transformation is known as *differential coordinates*. Transformation of all vertices at once can be described by a sparse matrix L, known as the Tutte Lapla-

cian:

$$L_{i,j} = \begin{cases} 1 & i = j \\ -\frac{1}{\|n(i)\|} & j \in n(i) \\ 0 & \text{otherwise} \end{cases} \quad (14)$$

which is closely related to the Kirchhoff Laplacian, defined as:

$$K_{i,j} = \begin{cases} \|n(i)\| & i = j \\ -1 & j \in n(i) \\ 0 & \text{otherwise} \end{cases} \quad (15)$$

For more details on discrete Laplacians, see Zhang [Zha04]. We argue that although Kirchhoff Laplacian has been used previously for mesh compression [SCOT03, CCOST05], it is in fact better to use the Tutte Laplacian for this purpose. If a mesh is very uniform, i.e. all vertices have the same valence, then L and K are equivalent up to a scalar multiplier. Since the resulting transformed coordinates will be uniformly quantised, it can be interpreted as changing the quantisation constant, which has no effect on the global rate-distortion curve. However, if the mesh vertices degree deviates from its mean significantly, then transforming by Kirchhoff Laplacian is equivalent to changing quantisation for each vertex depending on its valence. This may later impair the performance of entropy coding of the transformed data, because the entropy of the data will be influenced by both geometric entropy and the entropy of vertex valences.

Neither L nor K is invertible, and therefore it is not possible to reconstruct the original vertex positions v_i from the transformed vertex positions $\widehat{v}_i$, simply because shifting the whole model produces the same differential coordinates. It is therefore necessary to use additional *anchors*, i.e. vertices whose position will be stored directly. At least one such anchor is required for each connected component of the model. Having N anchors with indices $a_1, a_2, \ldots, a_N$, the extended (rectangular) Tutte Laplacian takes the following form:

$$\widetilde{L} = \begin{pmatrix} L \\ A \end{pmatrix} \quad (16)$$

where A is a $N \times V$ submatrix where $A_{i,j} = 1$ when $a_i = j$, $A_{i,j} = 0$ otherwise.

Notice that $\widetilde{L}$ only depends on mesh connectivity (it is a so-called combinatorial Laplacian), and it is applied consecutively on each coordinate, i.e. we first create a vector $x = [{}_x v_1, {}_x v_2, \ldots, {}_x v_V]$ of X coordinates of all vertices, transform it into $\widehat{x} = \widetilde{L}x$, and then similarly treat the y and z vectors. The transformed vectors $\widehat{x}$, $\widehat{y}$ and $\widehat{z}$ of length $V+N$ are then quantised and encoded. The decoder first decodes the connectivity of the mesh, constructs the rectangular Tutte Laplacian $\widetilde{L}$, and then solves the overdetermined systems of equations $\widetilde{L}x = \widehat{x}$, $\widetilde{L}y = \widehat{y}$ and $\widetilde{L}z = \widehat{z}$. In order to solve the three systems, it is possible to precompute the normal matrix of the system $\eta = \widetilde{L}^T.\widetilde{L}$, compute its LU decomposition and then use it to solve the three systems.

For more details on how to use the Kirchhoff Laplacian-based approach for static meshes, see Sorkine *et al.* [SCOT03]. Our proposal is to apply the Tutte Laplacian L on dynamic meshes. For this purpose, the PCA representation provides an elegant way to employ the Laplacian encoding technique, only instead of using XYZ coordiantes, we apply it on PCA coefficients associated with each vertex.

We can form the Tutte Laplacian L of the shared connectivity using equation (14), and having selected N anchor points, we can construct the rectangular transformation matrix $\widetilde{L}$ using equation (16). Having a matrix C of trajectory feature vectors of size $B \times V$, we can perform the forward transformation as matrix multiplication $\widehat{C} = (\widetilde{L}.C^T)^T$.

By forward transformation, we obtain a matrix $\widehat{C}$ of size $B \times (V+N)$, where the first V columns represent the feature vectors in differential coordinates for each vertex, while the last N columns are the feature vectors of the anchor vertices.

All elements of $\widehat{C}$ can be interpreted as coordinates of the same scale (in contrast to previously used Kirchhoff Laplacian), and therefore can be quantised using a single quantisation constant. For quantisation, we use a quantisation step related to average edge length, which seems to be a more intuitive setting (in relation to perceived distortion) than using mesh diagonal length. Having the average length $\widehat{l}$ of all the edges in all the frames, we quantise each floating point value e into an integer value $i = round\left((e.2^k)/\widehat{l}\right)$, where k is a user-specified quantisation roughness, which can be interpreted as "number of bits per average edge length". Note that this way we can better estimate the visual effect of quantisation based on the used parameter; however, the value of the parameter can be negative, which may seem counterintuitive.

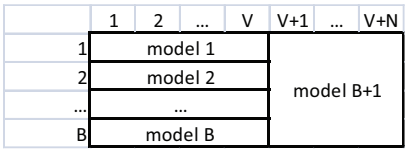

Figure 4: *Different statistic models used for encoding of elements of the $\widehat{C}$ matrix.*

After the quantisation, the elements of $\widehat{C}$ are entropy coded. The distribution of the values varies significantly depending on the position in the matrix, and it is therefore useful to use multiple statistical models. We choose to use a separate statistical model for each row of the matrix, and an additional one for the anchor points (see figure 4). For encoding, we use context-adaptive arithmetic coding in an implementation similar to [MWS03].

5. Experimental results

We have tested both rate-distortion performance and timings for the proposed Laplacian coordinates-based encoder, and

we are comparing the results with current state-of-the-art algorithms FAMC and Coddyac. Note that we are using re-implementations of the existing algorithms, which allow us to perform optimisation of parameters for the STED metric. Therefore, the results should provide a fair comparison of performance. All the data rates are in bits per frame and vertex (bpfv).

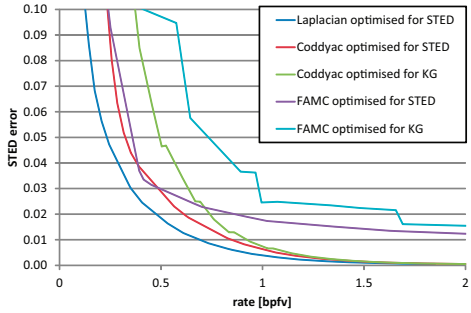

Figure 5: *Rate-distortion curve for the **dance** sequence.*

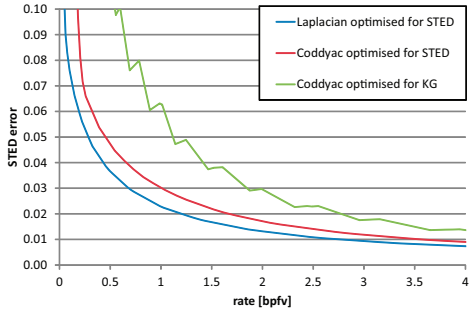

Figure 6: *Rate-distortion curve for the **jump** sequence.*

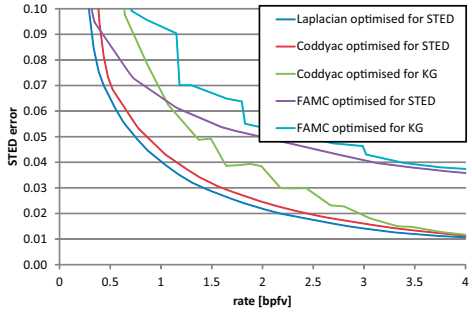

Figure 7: *Rate-distortion curve for the **cow** sequence.*

5.1. Rate-distortion performance

We have implemented the technique described in section 4 and we have compared the results with the state-of-the-art algorithms. Figures 5, 6, 7 and 8 show the rate-distortion curves obtained for some popular datasets used for testing

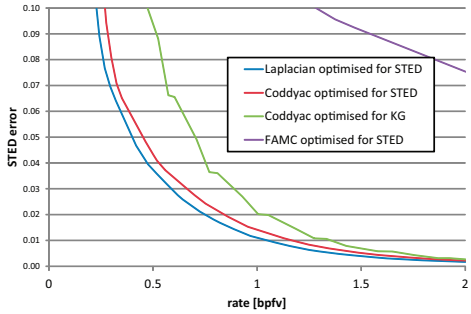

Figure 8: *Rate-distortion curve for the **chicken** sequence.*

the performance of dynamic mesh encoding. The figures demonstrate that using the Laplacian-based encoding, we were able to reduce the required data rate by up to 50% with respect to the next best encoder, which is the Coddyac approach configured for STED error.

We are only comparing our results against the FAMC and Coddyac algorithms, because these algorithms have been previously shown to provide the best RD performance. In some figures, the results of the FAMC are missing; this is because the results were too poor to fit in the RD chart.

We are also providing visual results in figure 9. The figure shows the result of a rather rough compression to 0.5 and 0.3 bpfv; however, it makes the difference between optimising KG and STED error easy to observe. Note that the difference between STED-optimised Coddyac, STED-optimised FAMC and the proposed algorithm is more visible during playback of the animation. Similar results were obtained for other models.

			Coddyac		Proposed algorithm	
	vertices	frames	encode [ms]	decode [ms]	encode [ms]	decode [ms]
Cow	2700	204	4326	1602	4492	2511
Chicken	3030	400	7208	1610	7394	2558
Dance	7061	201	3746	1398	3962	2605
Jump	15830	222	8158	3104	10888	5382
ClothBall	46598	94	10744	7234	12776	10358

Table 3: *Timings for the compression and decompression steps of the algorithm.*

5.2. Timings

Our implementation of the encoder uses an off-the-shelf linear algebra toolkit from Bluebit software, which is used for solving the sparse system of linear equations and also the eigenvalue decomposition necessary for PCA of trajectories. Using this library, we are able to achieve faster computation times than previously reported for the Coddyac algorithm and its derivatives. Table 3 shows timings for the Coddyac

algorithm, and for the proposed modification using Laplacian coordinates. Note that the actual time needed for encoding a particular dataset strongly depends on the chosen accuracy, which in turn influences the number of used basis vectors. The table shows computation times for an STED error of 0.02, at which the compressed mesh is usually almost indistinguishable from the original. For the testing, we have used a PC with Intel Core i7 CPU @ 2.67GHz.

The table shows that using Laplacian coordinates causes variable slowdown of 3-33% on compression and 43-86% on decompression, depending on the complexity of the datasets. We are using random placement of anchor points, which greatly speeds up the compression at a small cost in rate-distortion performance. In cases when top RD performance is required, it is possible to place anchors more rigorously by adding one at a time to a position of largest error. Such an approach, however, requires solving the sparse set multiple times, which slows down compression considerably. Decompression times are not affected by the selected anchor point placement scheme.

6. Conclusion

We have presented two ways to reduce the perceived error introduced by compression of dynamic meshes. Our contributions are:

- showing a general approach which configures existing algorithms to reduce the STED error (and possibly many other "well behaved" error measures),
- proposing a novel approach based on PCA in the space of trajectories and Laplacian coordinates, efficient in both RD performance and computational complexity,
- providing fair comparison under the new STED error measure.

We have demonstrated that reconfiguration of existing algorithms has a significant impact on performance, and it is therefore necessary in order to obtain a fair comparison of algorithms. By reconfiguring the Coddyac algorithm, we were able to gain a 20-30% performance improvement in our experiments.

Furthermore, we have shown that when we focus on a particular step of a compression algorithm (and parameter optimisation gives useful clues about which part to optimise), it is possible to further improve the performance. In our case, we have gained another 20-30% by replacing parallelogram encoding of feature vectors by using Laplacian coordinates.

Our paper also provides evidence that the recently proposed STED error metric does indeed capture perceived error better than previous metrics. So far we have not found that optimising this metric brings any new visible artifacts which are not addressed by this error measure.

In the future it might be interesting to consider using a derivative of the STED error measure for static meshes and test the performance of existing compression algorithms with respect to it. The results of the proposed compression algorithm should also be included in future experiments regarding perceived distortion of dynamic meshes.

7. Acknowledgements

This work has been supported by the Ministry of Education, Youth and Sports of the Czech Republic under the research program LC-06008 (Center for Computer Graphics).

The chicken character was created by Andrew Glassner, Tom McClure, Scott Benza and Mark Van Langeveld. This short sequence of connectivity and vertex position data is distributed solely for the purpose of comparison of geometry compression techniques.

References

[AM00] ALEXA M., MÜLLER W.: Representing animations by principal components. *Computer Graphics Forum 19*, 3 (2000).

[CCOST05] CHEN D., COHEN-OR D., SORKINE O., TOLEDO S.: Algebraic analysis of high-pass quantization. *ACM Trans. Graph. 24*, 4 (2005), 1259–1282.

[ČS05] ČERMÁK M., SKALA V.: Polygonization of implicit surfaces with sharp features by edge-spinning. *Visual Computer 21*, 4 (2005), 252–264.

[Int01] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION: *ISO/IEC 14496 Part 2: Visual*. International Organization for Standardization, 2001.

[Int09] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION: *ISO/IEC 14496 Part 16: Animation Framework eXtension (AFX), amendment 2: Frame-based Animated Mesh Compression (FAMC)*. International Organization for Standardization, 2009.

[IR03] IBARRIA L., ROSSIGNAC J.: Dynapack: space-time compression of the 3d animations of triangle meshes with fixed connectivity. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2003), Eurographics Association, pp. 126–135.

[JKJ*04] JANG E. S., KIM J. D. K., JUNG S. Y., HAN M., WOO S. O.: Interpolator data compression for mpeg-4 animation. *IEEE Transactions on Circuits and Systems for Video Technology 14*, 7 (2004), 989–1008.

[JT05] JAMES D. L., TWIGG C. D.: Skinning mesh animations. *ACM Transactions on Graphics (SIGGRAPH 2005) 24*, 3 (Aug. 2005).

[KG04] KARNI Z., GOTSMAN C.: Compression of soft-body animation sequences. In *"Computers & Graphics 28, 1"* (2004), pp. 25–34.

[Len99] LENGYEL J. E.: Compression of time-dependent geometry. In *SI3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics* (New York, NY, USA, 1999), ACM Press, pp. 89–95.

[MSK*06] MULLER K., SMOLIC A., KAUTZNER M., EISERT P., WIEGAND T.: Rate-distortion-optimized predictive compression of dynamic 3d mesh sequences. *SP:IC 21*, 9 (October 2006), 812–828.

[MWS03] MARPE D., WIEGAND T., SCHWARZ H.: Context-based adaptive binary arithmetic coding in the h.264/avc video compression standard. *IEEE Trans. Circuits Syst. Video Techn. 13*, 7 (2003), 620–636.

[MZP06] MAMOU K., ZAHARIA T., PRETEUX F.: A skinning approach for dynamic 3d mesh compression: Research articles. *Comput. Animat. Virtual Worlds 17*, 3–4 (2006), 337–346.

[MZP*08] MAMOU K., ZAHARIA T., PRÊTEUX F., STEFANOSKI N., OSTERMANN J.: Frame-based compression of animated meshes in MPEG-4. In *ICME 2008: 2008 IEEE International Conference on Multimedia and Expo* (June 2008), pp. 1121–1124.

[PA05] PAYAN F., ANTONINI M.: Wavelet-based compression of 3d mesh sequences. In *Proceedings of IEEE ACIDCA-ICMI'2005* (Tozeur, Tunisia, november 2005).

[Ros99] ROSSIGNAC J.: Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics 5*, 1 (/1999), 47–61.

[SCOT03] SORKINE O., COHEN-OR D., TOLEDO S.: High-pass quantization for mesh encoding. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (2003), Eurographics Association, pp. 42–51.

[SO06] STEFANOSKI N., OSTERMANN J.: Connectivity-guided predictive compression of dynamic 3d meshes. In *Proc. of ICIP '06 - IEEE International Conference on Image Processing* (oct 2006).

[TG98] TOUMA C., GOTSMAN C.: Triangle mesh compression. In *Graphics Interface* (June 1998), pp. 26–34.

[VS07] VÁŠA L., SKALA V.: Coddyac: Connectivity driven dynamic mesh compression. In *3DTV Conference Proceedings* (2007).

[VS09] VÁŠA L., SKALA V.: Cobra: Compression of the basis for the pca represented animations. *Computer Graphics Forum 28*, 6 (2009), 1529–1540.

[VS11] VÁŠA L., SKALA V.: A perception correlated comparison method for dynamic meshes. *IEEE Transactions on Visualization and Computer Graphics 17*, 2 (2011), 220–230.

[WBSS04] WANG Z., BOVIK A. C., SHEIKH H. R., SIMONCELLI E. P.: Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing 13*, 4 (2004), 600–612.

[WLB04] WANG Z., LU L., BOVIK A. C.: Video quality assessment based on structural distortion measurement. *Sig. Proc.: Image Comm. 19*, 2 (2004), 121–132.

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

[ZO04] ZHANG J., OWEN C. B.: Octree-based animated geometry compression. In *DCC '04: Proceedings of the Conference on Data Compression* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 508–517.

Appendix A: Motivation for using differential coordinates

The aim of this section is to provide further theoretical support as to why differential coordinates perform better with respect to the STED measure.

Consider the one ring neighbourhood of a single vertex with original coordinates p^o. There are N neighbours of the vertex with coordinates $n_i^o, i = 1..N$, where N is usually 6 for regular meshes. We will first consider the usual absolute encoding of the positions, involving quantisation. The decoded position of the vertex is $p^a = p^o + \Delta$, where Δ is the error due to quantisation. Δ is a zero mean uniformly distributed variable spanning the quantisation range. Similarly, the decoded neighbouring vertex positions can be written as $n_i = n_i^o + \Delta_i$, where each Δ_i has the same properties as Δ.

Relative edge length differences of incident edges are then expressed as:

$$\begin{aligned} rl_i^a &= \frac{\|n_i - p^a\| - \|n_i^o - p^o\|}{\|n_i^o - p^o\|} \\ &= \frac{\|n_i^o + \Delta_i - p^o - \Delta\| - \|n_i^o - p^o\|}{\|n_i^o - p^o\|} \end{aligned} \tag{17}$$

Without loss of generality, we can split the errors Δ_i into a common part $k = 1/N \sum_{i=1}^{N} \Delta_i$ and neighbour specific parts $\hat{\Delta}_i = \Delta_i - k$. Note that $\sum_{i=1}^{N} \hat{\Delta}_i = 0$. Denoting $e_i = n_i^o - p^o$ the original lengths of incident edges, we can rewrite 17 as:

$$rl_i^a = \frac{\|e_i + k + \hat{\Delta}_i - \Delta\| - \|e_i\|}{\|e_i\|} \tag{18}$$

Now we will consider the case when the differential coordinates are quantised instead of the absolute coordinates. For reasons of simplicity, we will again assume that the neighbouring vertices have been moved from their original positions n_i^o to quantised positions $n_i = n_i^o + \Delta_i = n_i^o + k + \hat{\Delta}_i$. The exact differential coordinates of the central vertex are $d^o = p^o - 1/N \sum_{i=0}^{N} n_i^o$. Quantised differential coordinates are $d = d^o + \Delta$, and the reconstructed position of the vertex can be written as:

$$p^r = \frac{1}{N}\sum_{i=1}^{N} n_i + d = \frac{1}{N}\sum_{i=1}^{N}(n_i^o + k + \hat{\Delta}_i) + p^o - \frac{1}{N}\sum_{i=0}^{N} n_i^o + \Delta \tag{19}$$

Since $\sum_{i=1}^{N} \hat{\Delta}_i = 0$, we can write $p^r = k + p^o + \Delta$. Finally, we can again express the relative edge length differences for the case of differential coordinates coding:

$$\begin{aligned} rl_i^r &= \frac{\|n_i^o + k + \hat{\Delta}_i - k - p^o - \Delta\| - \|n_i^o - p^o\|}{\|n_i^o - p^o\|} \\ &= \frac{\|e_i + \hat{\Delta}_i - \Delta\| - \|e_i\|}{\|e_i\|} \end{aligned} \tag{20}$$

The contribution of the neighbourhood to the STED error is the deviation of relative edge lengths within the given neighbourhood. Original edge lengths e_i are constant, and $\hat{\Delta}_i$, Δ and k are all zero mean random variables. Therefore, the deviation of relative edge lengths converges to zero as the deviations of $\hat{\Delta}_i$, Δ and k converge to zero. However, using differential coordinates we have eliminated k; thus we may expect that the contribution to STED error will be lower, even when the same quantisation intervals are used.

The above derivation is valid for three component XYZ vectors. However, it can be applied to feature vectors as well, since the feature vectors are in fact XYZ coordinates expressed in a different basis.

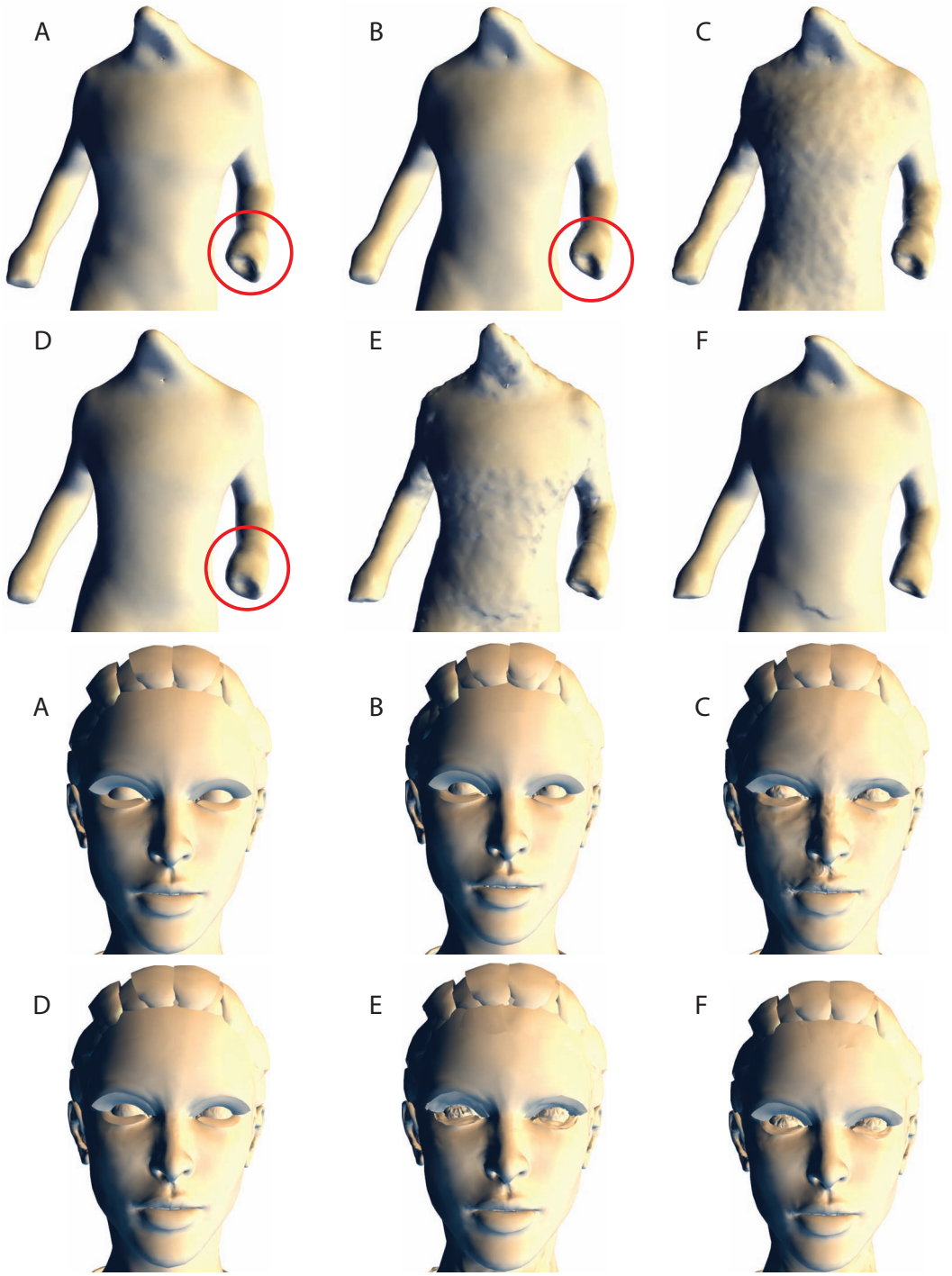

Figure 9: *Examples of a single frame from the jump and walk sequences, showing the original (A), compressed by the proposed Laplacian based algorithm (B), compressed by Coddyac optimised for KG error (C), compressed by Coddyac optimised for STED error (D), compressed by FAMC optimised for KG error (E) and compressed by FAMC optimised for STED error (F). Algorithms were configured to provide an optimal result for the data rate of 0.5 bpfv (jump) and 0.3 bpfv (walk). Optimisation for KG-error introduces obvious artifacts, not present in the other results. STED-optimised Coddyac partially avoids these artifacts at the cost of strongly smoothing the model. With the proposed approach, the smoothing effect is visibly smaller (see the hands of the jump model). STED-optimised FAMC produces some other types of artifacts, such as visible transitions between the clusters caused by rough quantisation of transformation matrices.*