

Error Diffusion on Meshes[☆]

Marc Alexa, Jan Eric Kyprianidis

TU Berlin, Germany

Abstract

We consider the problem of quantization for surface graphs. In particular, we generalize the process of error diffusion to meshes and then compare different paths on the surface when used for error diffusion. We suggest paths for processing mesh elements that lead to better distributions of available neighbors for error diffusion. We demonstrate the potential benefit of error diffusion at several mesh processing applications: quantization of differential mesh coordinates, including an extension to animated geometry, and vertex subset selection for mesh simplification. These applications allow us to compare different paths objectively. We find that the linear time solution results in excellent overall performance, outperforming other traversals taken from the literature. We conclude that the proposed path can be taken as a starting point for any application of error diffusion on meshes.

1. Introduction

Quantization is a fundamental operation in computer science: whenever quantities are encoded they are usually quantized; moreover, in many applications one has only a limited set of options, and choosing one of them is necessary. For Euclidean domains (i.e., images), the idea of diffusing the error resulting from quantization has been investigated for decades, greatly evolving over time [1, 2, 3, 4, 5, 6]. Interestingly enough, the subject has not been investigated for discrete surfaces in depth, despite a number of recent applications that would greatly benefit from a generalization of error diffusion [7, 8, 9, 10].

The fundamental idea of error diffusion for halftoning images (Figure 1(a)) is to spread the error resulting from quantization to neighboring image elements [11]. This requires that *at least some* neighboring elements have not been quantized (and, thus, fixed) when a certain element is processed. The pertinent research questions for error diffusion directly follow from this procedure: 1) In what order are the elements processed? 2) How is the error diffused to other elements? We note that both problems have significantly more structure on a regular grid in the plane compared to irregular surface meshes. In this work, we tackle the first problem, namely how to generate processing orders for surface meshes that have desirable properties.

A natural property for the processing order is that for most elements roughly half of the neighbors are unprocessed for each element. A path with this property may quite easily be computed for a regular grid and several such options are considered in the literature on error diffusion for images: scanline, serpentine paths, space filling curves, and others. Generating curves with similar properties on a mesh is theoretically very difficult (i. e., in NP) and approximations are still not easy to compute.

Our main contribution is the design and analysis of paths that have properties similar to the ones used in image processing. In particular, we consider and evaluate the following orders:

- The order of breadth-first traversal, or variants such as the one by Alliez et al. [12] used for remeshing.
- Paths generated by algorithms for the generation of triangle strips [13, 14], rendering [15], and mesh compression [16].
- An order inspired by the Warnsdorff rule [17] for the linear-time approximation of a Knight's tour through a chessboard (see Figure 1).
- Variants of the Warnsdorff rule based on locally or globally sorting the elements according to different criteria.

We show that our interpretation of the Warnsdorff rule is fast (linear in the vertices) and outperforms all other paths from the literature in terms of being similar to the properties of error diffusion orders for images.

Moreover, to make our analysis meaningful in terms of applications, we present two applications of error diffusion for mesh processing and show that the new paths again outperform all alternatives. The first mesh processing application is vertex quantization applied to the Laplacian of the mesh geometry [18], also known as high-pass quantization. In addition, we apply error diffusion to a similar technique for animated mesh data. Here, vertex trajectories are first transformed into a PCA-basis, before applying the Laplacian. A state-of-the-art coder [19] applies the parallelogram-rule [16] to this data. We show that quantizing the data directly while performing error diffusion improves on the entropy of the resulting code. See Section 4 for details.

What amounts to quantizing into black and white dots for images can be reinterpreted as a way to select vertices in a mesh. As an application of this idea we show how to subsample a

[☆]This is the authors' version of the work. The definitive version was published in *Computers & Graphics*, Vol. 46, 2015, pp. 336–344. doi: [10.1016/j.cag.2014.09.010](https://doi.org/10.1016/j.cag.2014.09.010).

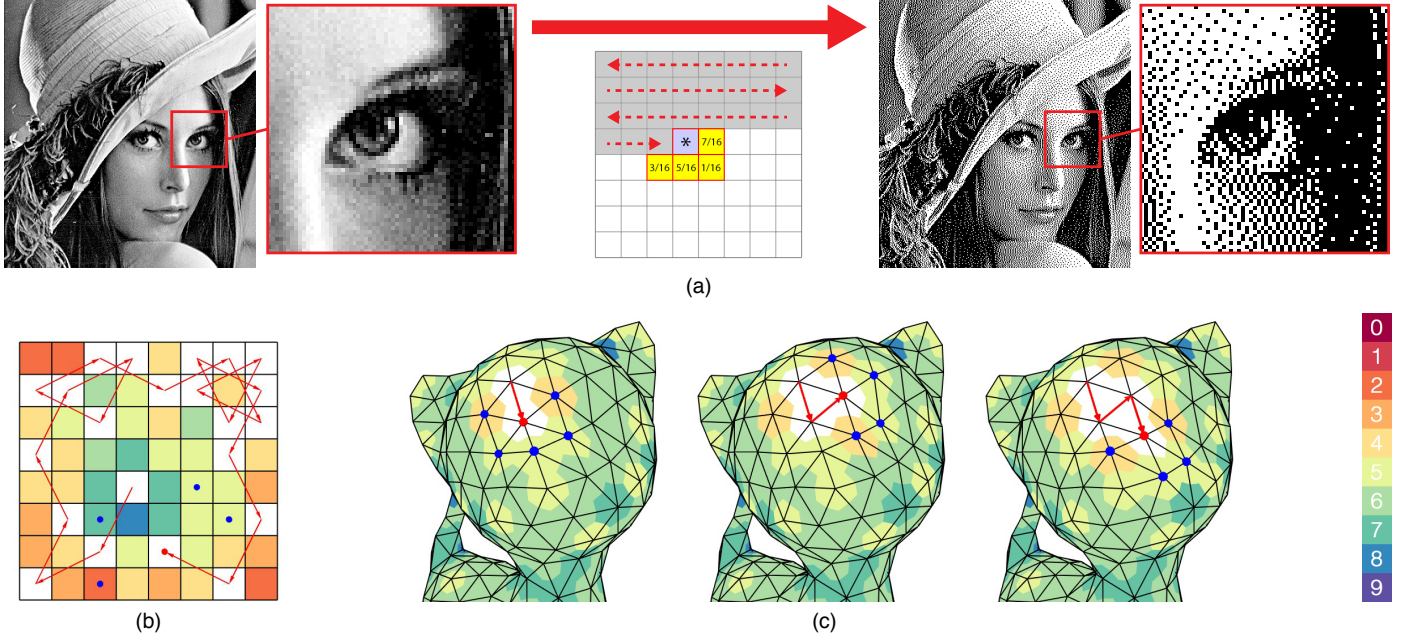


Figure 1: (a) Classical error diffusion for halftoning images converts grayscale images to black-and-white images by processing the input pixel-by-pixel. For each pixel, its grayscale value is converted to either black or white and the resulting error distributed to the neighboring unprocessed pixels. (b) Motivated by the Warnsdorff rule, which describes an algorithm for a Knight’s tour on a chessboard, we propose a processing order that enables error diffusion on surface meshes. (c) In contrast to processing orders popular in image processing, the Warnsdorff rule naturally generalizes to arbitrary surface graphs. Colors encode the number of legal moves, with white denoting already processed elements. Blue points mark legal moves at the current position.

mesh for simplification (see Section 5). The remaining vertices are then reconnected as triangles based on intrinsic graph distances, as in Boubekur and Alexa [20]. We show that we achieve better preservation of the original geometry in linear time for the same vertex count, thus providing a very fast mesh decimation technique with acceptable quality.

2. Error Diffusion on Meshes

In this work, we will be primarily interested in the diffusion of vertex data. However, the same principles also apply to face data by considering the dual graph. Let us assume that a mesh is given together with a set of values associated with the vertices of the mesh. Associated values could be, for example, colors, vectors representing location, or material properties. Our aim is to quantize these values, i. e., to replace each value by a new value $\mathbf{q} \in Q$ from a finite set of appropriate type. This quantization process results in errors and the underlying idea of error diffusion is to distribute these errors to the yet unprocessed neighboring values.

For an arbitrary mesh we cannot make any assumptions on the regularity of its structure and therefore might have to account for irregularities during the distribution of the error. To this end, we can associate to each vertex a *mass* m_i that is deduced from the geometry. This is typically the area associated to a vertex (e. g., the Voronoi area [21], or the incident face areas based on vertex degrees [22]) but the term mass is common in the related literature on finite element methods. In some application it might be convenient and sufficient to assume that all masses are equal, i. e., $m_i = 1$.

Vertices are identified by an index i . Let $\mathbf{v}_i$ denote the value. When the vertex with index i is quantized, its assignment to a value in Q is chosen to minimize the quantization error:

$$\mathbf{v}_i \leftarrow \mathbf{q}_k, \quad \arg \min_k \|\mathbf{v}_i - \mathbf{q}_k\|, \quad \mathbf{q}_k \in Q. \quad (1)$$

Once the associated value of vertex i has been replaced by one of the values of Q , it becomes *unavailable*. The availability is reflected by a status variable $\mathbf{a} = \{a_i \in \{0, 1\}\}$. Initially set to $a_i = 1$ for all i , the status changes to $a_i = 0$ after the assignment.

Since we assume all initial values to be locally constant, the the total error for distribution is given by a multiplication with the element’s mass:

$$\mathbf{e}_i = m_i(\mathbf{v}_i - \mathbf{q}_k) \quad (2)$$

The distribution of the total error to available vertices depends on a mask of coefficients that encode distance and mass. We typically use a Gaussian mask¹ of the form

$$\tilde{c}_{ij} = \frac{m_j}{m_i} \exp\left(-\frac{1}{2\sigma} d_{ij}^2\right), \quad (3)$$

where d_{ij} is the distance between the vertices i and j . Possible choices are either the topological distance in the incidence graph of the mesh, the Euclidean distance, or the geodesic distance between the vertices. Weighing the elements by their

¹We note that this choice of coefficients is likely suboptimal for many applications. Yet, in analogy to error diffusion for images, the choice of coefficients is highly non-trivial, and we leave this for future research.

mass results in a mass-proportional distribution of the total error. As not all elements are available for the distribution of the error, the coefficients need to be normalized throughout the process. More precisely, using the sum

$$\Sigma_i = \sum_{j \in \{d_{ij} \leq r\}} a_j \tilde{c}_{ij} \quad (4)$$

yields the normalized mask values

$$c_{ij} = \frac{a_j \tilde{c}_{ij}}{\Sigma_i}. \quad (5)$$

The radius r defines the size of the mask and depends, as the standard deviation σ of the Gaussian weights, on the application. For topological distances in a triangle mesh, a reasonable choice is a mask of size $r = 1$.

The weights determine how much of the total error is distributed to each available vertex. Since the total error is an integral quantity but the vertices are assumed to carry a point-wise value, the proportional error must be divided by the mass of the receiving vertex when distributing the error:

$$\mathbf{v}_j \leftarrow \mathbf{v}_j + \frac{c_{ij}}{m_j} \mathbf{e}_i \quad (6)$$

When no vertices are available, the error cannot be diffused, adding to the global quantization error of the process. This setup for the masks does not imply a specific ordering. The mesh can be processed in *any order*, however, possibly yielding different results.

In case of a connected processing order an interesting feature can be included. From the literature on error diffusion for images it is known that it is advantageous if approximately half of the error is distributed to the next element in processing order [3, 23]. In our algorithmic setup, we therefore include this possibility in form of an control parameter $\zeta \in [0, 1]$. The pseudocode in Algorithm 1 summarizes this procedure, where we exploit that Σ_i is independent of j and thus can be pre-multiplied with $\mathbf{e}_i$.

3. Orders for Error Diffusion

The original error diffusion algorithm by Floyd and Steinberg [24] exploits the regularity of the underlying adjacency graph (i. e., a rectangular grid) and processes image elements in scan line order. Floyd-Steinberg error diffusion is known to produce artifacts (see [3] for a detailed discussion), which resulted in a number of different modifications to the original algorithm. In particular, different processing orders, such as Knuth's dot diffusion [1], the serpentine path [11], or space-filling curves [25] have been considered. In contrast to a random processing order, these processing orders have the important property that all image elements always have unprocessed neighbors for diffusing the error. By symmetry, this comes at the price of having not more than half of the neighborhood available. While it is clear that for any path through a graph every element would have this property *on average*, it is not clear how to generate paths that

Algorithm 1: Irregular error diffusion.

Data: Index i of the vertex to be quantized, index η of next vertex in processing order (if available), and fraction ζ of the error distributed to next vertex.

Result: Value of vertex i is replaced by an element of $\mathcal{Q}$ and the error is diffused to the available vertices in the neighborhood.

```

 $k_{\min} \leftarrow 0$ 
for  $\mathbf{q}_k \in S, k \geq 1$  do
    if  $|\mathbf{v}_i - \mathbf{q}_k| < |\mathbf{v}_i - \mathbf{q}_{k_{\min}}|$  then
         $k_{\min} \leftarrow k$ 
 $\mathbf{e} \leftarrow m_i(\mathbf{v}_i - \mathbf{q}_{k_{\min}}), \mathbf{v}_i \leftarrow \mathbf{q}_{k_{\min}}, a_i \leftarrow 0$ 
if  $\zeta > 0$  and  $\text{is\_valid}(\eta)$  then
     $\mathbf{v}_\eta \leftarrow \mathbf{v}_\eta + \zeta \mathbf{e}$ 
     $\mathbf{e} \leftarrow (1 - \zeta) \mathbf{e}$ 
if  $\Sigma_i > 0$  then
     $\mathbf{e} \leftarrow \mathbf{e} / \Sigma_i$ 
    for  $j \in \{d_{ij} \leq r\}$  do
        if  $a_j = 1$  then
             $\mathbf{v}_j \leftarrow \mathbf{v}_j + \tilde{c}_{ij/m_j} \mathbf{e}$ 

```

have the desirable feature of having at least half of the neighborhood available for a large number of elements.

Hamiltonian paths are a special and well-studied class in this setting: each element has at least one unvisited neighboring element. Yet, it is well known that Hamiltonian paths for general graphs are NP-hard [26] and thus infeasible to compute. Hence, we have to assume that several elements will not have available neighbors for error diffusion. Our problem is more general, however, in that we like to minimize the number of elements with only few available neighbors, since then more neighbors are available for distributing the error. Nonetheless, we take our inspiration from fast approximation approaches for finding Hamiltonian paths.

The graph we are considering for our approaches is defined as follows: an edge exists between elements i and j when $d_{ij} \leq r$, i. e., the coefficient $c_{ij} > 0$ and, thus, allows for error diffusion. Hence, in case of the topological distance, if A is the adjacency matrix of the original graph, we are looking for paths inside the graph with adjacency matrix A^r . As a baseline for a path, we consider a random ordering of elements. In the following we discuss several choices for the construction of a good path and evaluate these choices purely based on the graph. Further evaluation, considering specific applications, then follow in the next sections.

3.1. The Warnsdorff Path

While Hamiltonian path problems are usually NP-complete, there exist a variety of polynomial solutions for planar rectangular graphs. We use a particularly simple heuristic initially suggested by Warnsdorff [17] for computing a Knight's tour on a chessboard (Figure 1(b)): from the available moves, pick the one that leaves the smallest number of subsequent moves; ties

are broken arbitrarily. Several heuristics for the tie breaker are known to improve the situation. For example, Pohl [27] proposed to break ties by summing the number of moves available at the next level and picking the one yielding a minimum. This results in a simple, elegant, and *linear* algorithm for computing a path. The constant for this linear algorithm depends on the size of the masks. Yet, since error diffusion also needs to iterate over the elements in the mask to diffuse the error, a smaller constant cannot be achieved in any case.

An alternative to summing the number of available moves is to compute the sum of the available mask coefficients $\tilde{c}_{ij}$ (see Eq. (5)). For $r = 1$ this is equivalent up to scaling to computing the number of available moves. For $r > 1$, however, primitives with fewer available moves on the inner rings will be preferred. The sum of available mask coefficients is given by Eq. (4) and the sum of available mask coefficients on the next level is given by:

$$\Sigma'_i = \sum_{j \in \{d_{ij} \leq r\}} a_j \Sigma_j \quad (7)$$

There is a minor tweak to achieve guaranteed linear complexity: whenever the current primitive has no available neighbors the algorithm needs to search for a new primitive to start from. We do this by iterating over the availability a_i of primitives. Once we found an index u with $a_u = 1$ we store the index. In a subsequent search we start at index $u + 1$, since we know all primitives with smaller index are unavailable (including u that now has been processed). In total, this adds a single sweep over the primitives a_i , i. e., another linear term with a comparatively small constant. The pseudo-code in Algorithm 2 gives the details for this algorithm.

Algorithm 2: Next primitive using the Warnsdorff rule.

Data: Index i of current primitive, a global index u such that $a_{u'} = 0 \ \forall u' < u$

Result: Index l of next primitive

$l \leftarrow -1$

$\Sigma_{\min}, \Sigma'_{\min} \leftarrow \infty$

for $j \in \{d_{ij} \leq r\}$ **do**

if $a_j > 0$ **then**

if $\Sigma_j < \Sigma_{\min} \vee (\Sigma_j = \Sigma_{\min} \wedge \Sigma'_j < \Sigma'_{\min})$ **then**

$l \leftarrow j$

$\Sigma_{\min} \leftarrow \Sigma_j$

$\Sigma'_{\min} \leftarrow \Sigma'_j$

if $l < 0$ **then**

$l \leftarrow u + 1$

while $a_l = 0$ **do**

$l \leftarrow l + 1$

$u \leftarrow l$

3.2. Global Variants of the Warnsdorff Rule

The Warnsdorff path discussed in the previous section tries (by construction) to minimize the number of connected com-

ponents. For the application to error diffusion, however, a connected path is not strictly necessary. Therefore, we have experimented with a number of other approaches to define the next primitive globally. The main idea behind the heuristics we tested is this: try to avoid that neighborhoods fill up by preferring primitives that already have many unavailable neighbors.

Global. Among different choices, we found it best to consider the number of available neighboring primitives, as suggested by Warnsdorff and Pohl for the Knight’s path. To this end, we again used the sum of available coefficients on the current level, and the sum of available coefficients on the next level to break ties. Given an implementation for priority queues with the possibility to update the priority of elements in the queue (we use the relaxed heap in Boost²), all required is to supply an operator for the comparison of two elements. This operator is simply based on the comparison of the Σ - and Σ' -values of the primitives. This requires maintaining a heap for all primitives, adding an additional $O(\log n)$ term to the asymptotic computational complexity.

Tree. As a further alternative, we considered allowing to add primitives to the path whenever they are a neighbor of an already processed primitive. Compared to the Warnsdorff path algorithm, this extends the set of possibilities beyond the current node, while limiting the set of processed nodes in comparison to the global approach and enforcing more continuity. Moreover, the set to be sorted is reduced. This approach is equivalent to a breadth first search over the graph: a list of nodes is active, if they have already been processed but are not completely surrounded by other processed nodes. We always pick the “best” node from the set of active nodes. Because of this similarity to graph traversal algorithms, the list of nodes will essentially form a (spanning) tree. For surface graphs, the number of active elements tends to be on the order of $\sqrt{n}$, where n is the total number of primitives, as it is the circumference of the currently conquered primitives. This reduces the factor for the $O(\log(n))$ -term necessary to keep the heap data structure sorted. In our setting, we define the “best” element to pick in the same way as before, namely, as the one with the smallest number of available neighbors based on the sum of available mask coefficients.

3.3. Analysis of the Different Paths

The characteristics of the resulting paths can be compared independently of any application, since the criteria for the different path constructions are based on the availabilities a_i and the mask coefficients $\tilde{c}_{ij}$, which depend only on the geometry of the graph. For quantitative comparison, we compute the number of available neighbors along the path as well as runtimes, without considering the actual diffusion of error, which is identical for all algorithms.

Figure 2 shows the distribution of available primitives relative to the primitive count averaged over several meshes. With the exception of the paths generated by the triangle strip and

²<http://www.boost.org>

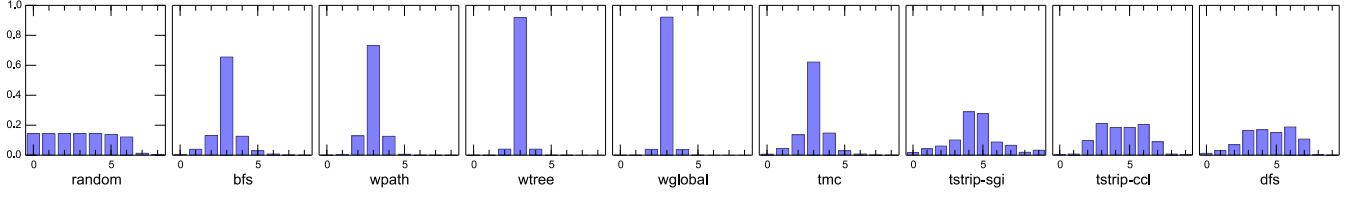


Figure 2: Average relative neighborhood frequencies of the available faces for different paths. The global paths (wtree, wglocal) are performing best, with very similar distributions. The Warnsdorff path (wpath) still shows a very good distribution.

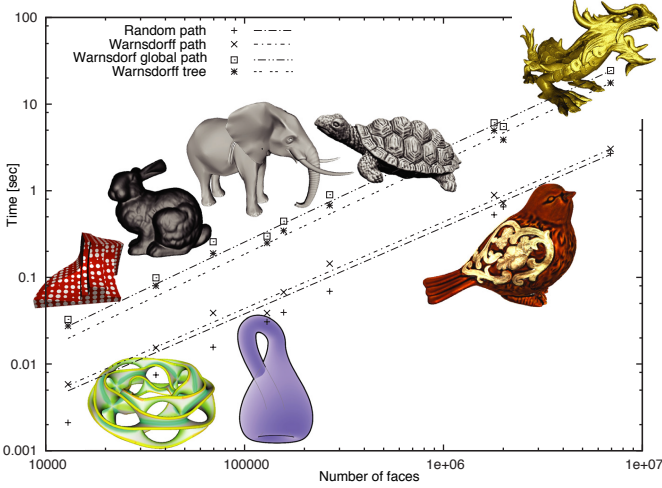


Figure 3: Relative execution times for computing the different paths (without actually diffusing error) for meshes with different number of faces on a log-log scale.

rendering algorithms, where faces have been used, all shown distributions have been computed using vertices as primitives. As can be seen, a random path generates a uniform distribution of available neighbors, which is highly undesirable, since low numbers of available neighbors are as likely as higher numbers. A straightforward breadth first search (BFS), as well as the ordering generated by the mesh compression algorithm (TMC) by Touma and Gotsman [16], provide a reasonable distribution of available neighbors but are out-performed by the specifically designed approaches based on the Warnsdorff rule. Triangle strip algorithms also generate paths on surfaces. We evaluated a variant of the original SGI algorithm [13] and a more recent approach by Yoon et al. [14]. Both approaches show a distribution that is close to uniform, making them unsuitable for the purpose of error diffusion. This is due to the different requirements. Regardless of how many neighbors of an element have been visited before, triangle strips are typically constructed to continue as long as possible or are optimized for cache coherence. Similar considerations apply for the DFS sequence from [15]. The Warnsdorff path, with second level tie breaking as proposed by Pohl, shows excellent performance. There is a strong peak for 2–4 neighbors. The two global approaches yield nearly similar results, both outperforming the Warnsdorff path in terms of the available neighbors statistic.

The necessary relative computation times are shown in Figure 3. This demonstrates that all techniques behave almost lin-

early in the number of vertices.

Note that simply laying out the mesh in the plane and then using standard techniques is not an option. Such an algorithm would be necessarily super linear, since it would require solving a sparse linear system and some form of sorting to obtain a path through the transformed primitives. Moreover, the practical run time and requirements on main memory would be significantly higher, in general, than the proposed approach. In addition, laying out mesh in the plane may suffer from numerical problems and may generate visual discontinuities across the boundary. If boundaries are identified, i. e., if the diffusion is performed across boundaries, the distribution of available neighbors may suffer. And even if all these issues can be addressed, not much is gained in the end, since the mesh is still *irregular*, though then living in the plane.

4. High-Pass Quantization

In the previous sections, we outlined how to perform error diffusion on arbitrary meshes. As an application we show in this section how our error diffusion framework can be applied for high-pass quantization of vertex coordinates. Most mesh compression schemes require at some stage the quantization of vertex coordinates. Quantizing vertex coordinates of meshes in Cartesian coordinates, however, introduces strong high frequency errors. By contrast, taking the differentials of coordinates and quantizing them (Figure 4(b)) moves the error into the low frequencies of the geometry, as shown in Figure 4(c). This approach is known as high-pass quantization and was pioneered by Sorkine et al. [18, 28]. In this section we will demonstrate that error diffusion completely avoids the large deformations that result from quantizing differentials (i. e., low frequency geometry error, potentially tackled by anchoring, also shown in Figure 4(c)), as error diffusion minimizes the global energy dissipation (Figure 4(d)).

4.1. High-Pass Quantization of Static Meshes

High-Pass Transform. To avoid the encoding of additional data such as edge lengths, we assume that the decoding side in a potential transmission scheme has no other information available than the combinatorics and therefore use the combinatorial (or umbrella) Laplacian. For details we refer the reader to a rich literature on Laplacian computations for meshes, such as [29]. For all vertices $\mathbf{v}_i \in \mathbb{R}^3$ we compute

$$\delta_i = \mathbf{v}_i - \frac{1}{|\mathcal{N}(i)|} \sum_{(i,j) \in \mathcal{E}} \mathbf{v}_j, \quad (8)$$

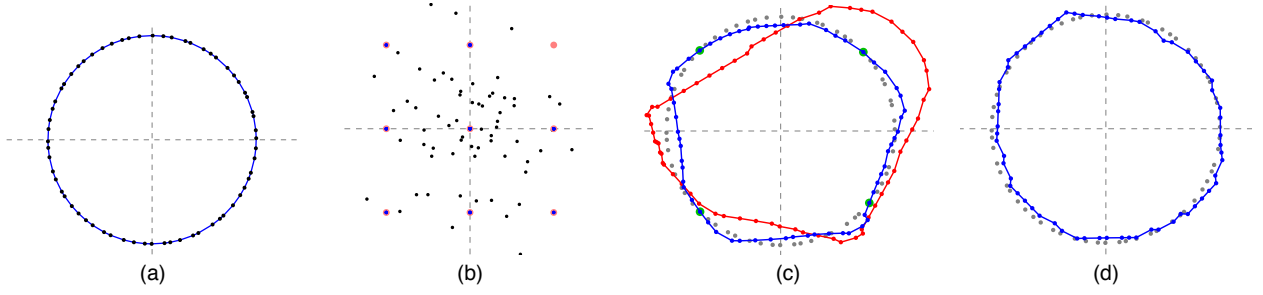


Figure 4: Illustration of high-pass quantization in two dimensions. (a) Points randomly distributed on a circle. (b) Differential coordinates of the points (black) and quantized coordinates (blue). (c) Reconstruction of the quantized coordinates without (red) and with anchor points (blue). (d) Reconstruction of differential coordinates quantized without any anchor points using error diffusion.

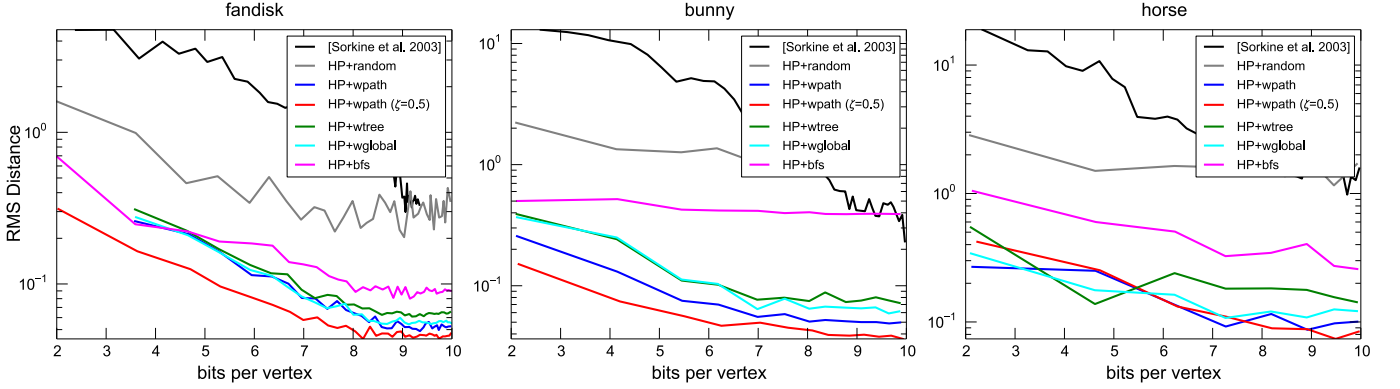


Figure 5: RMS distances, normalized by model diameter, of different meshes reconstructed from differential coordinates using different error diffusion schemes.

where $|N(i)|$ is the degree of the vertex i and $\mathcal{E}$ is the set of edges of the mesh. Equivalently, Eq. (8) can also be written in matrix notation as $\mathcal{D} = LV$, with L denoting the Laplacian operator and V denoting the Cartesian vertices of the mesh.

Quantization. The differential coordinates $\mathcal{D} = \{\delta_i\}$ can now be quantized, resulting in a new set of differential coordinates $\mathcal{D}' = \{\delta'_i\}$ with $\delta'_i \in Q$. We use uniform quantization as in [18], however, with a small twist. Instead of simply computing the bounding box, we take the bounding box containing 99.9% of the differential coordinates, typically leading to significant improvement of the results. Error diffusion is incorporated by performing the quantization and distribution of the error as explained in Section 4, while processing the primitives in order of one of the paths from Section 3. Finally, the quantized differential coordinates $\{\delta'_i\}$ can be enumerated and encoded using, e.g., Huffman coding or a more advanced coding scheme.

Reconstruction. In order to recover the mesh, the quantized differential coordinates are first decoded. Then the Cartesian coordinates V' of the mesh can be recovered by solving the system

$$LV' = \mathcal{D}'. \quad (9)$$

In our implementation we use the sparse LU decomposition as provided by the Cholmod based solver of Eigen [30]. As long as the mesh is well-defined, the Laplacian L has only the constant functions in its kernel, and the solution is defined up to global

translation. As part of the compressed data we therefore also store the mean of the vertices of the mesh.

Results and Discussion. For the quantitative statements, we compare quantization based on error diffusion using the paths discussed in the previous section and classical quantization by measuring the root mean square (RMS) distance [31] of the reconstructed mesh to the original mesh. As can be seen in Figure 5, error diffusion significantly improves the situation across all quantization levels in comparison to the original approach by Sorkine et al. [18, 28], which does not diffuse the error. Moreover, it can be seen that the Warnsdorff path significantly improves over the random processing order and breadth-first-search. Furthermore, it can be observed that the global Warnsdorff-based paths achieve no improvement. In fact, the simple Warnsdorff path with $\zeta = 0.5$ outperforms all other processing orders. Figure 6 shows a visual comparison with the original approach, as well as results produced by our approach using random and Warnsdorff paths. As can be seen, the Warnsdorff path achieves a significant improvement over the other approaches.

We note that the RMS distance is a measure that is completely unrelated to this type of quantization. In addition, we have repeated this experiment based on a number of other distance measures, such as the Hausdorff distance and visual similarity (as used by Sorkine et al. [18]), and obtained similar results.

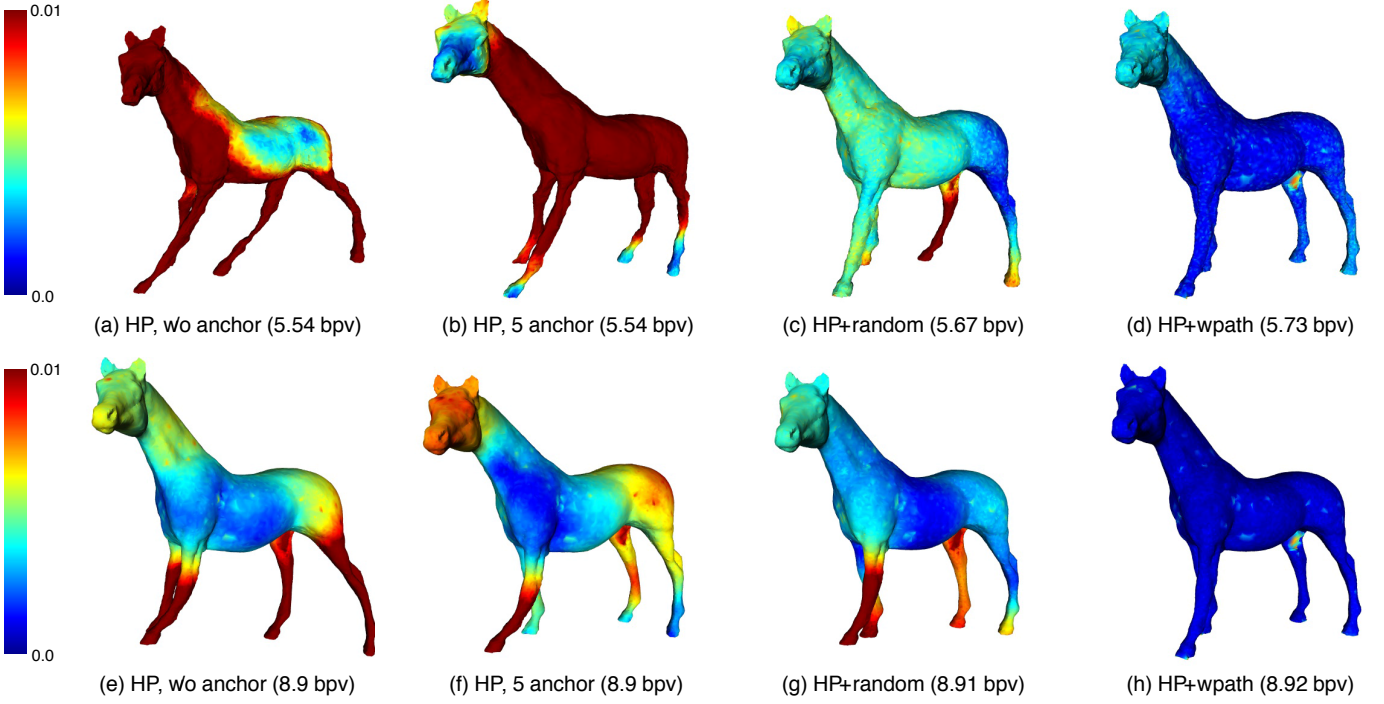


Figure 6: Comparison of high-pass quantization with and without error diffusion: (a),(b),(e),(f) High-pass quantization as proposed by Sorkine et al. [18]. (c),(d),(g),(h) Proposed method: High-pass quantization with error diffusion. Our approach using error diffusion does not use any anchor points. All methods use the same bounding box (containing 99.9% of the differential coordinates) for uniform quantization.

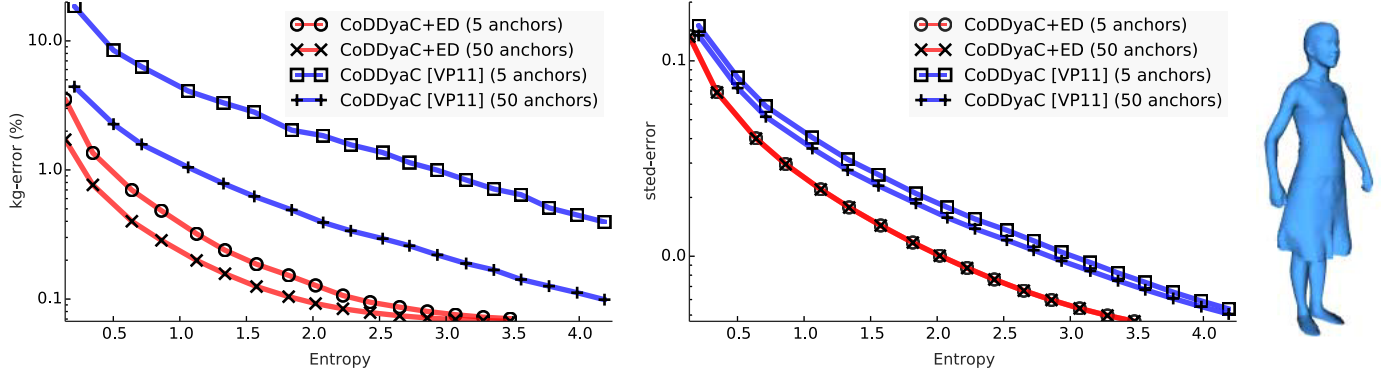


Figure 7: Comparison of our error diffusion utilizing CoDDyAC variant with the differential coordinate-based CoDDyAC variant [19]. Shown are entropy vs. model distortion of the quantized trajectory coefficients. In both cases a PCA basis of 150 vectors was used and the same set of randomly chosen anchors.

For all paths, we assumed that the masses m_i (i.e., the vertex areas) are the same. Early experiments that we conducted showed that using mass-weighted paths is not advantageous in this case. This is reasonable, since the computation of the differential coordinates and the reconstruction is based on the combinatorial Laplacian. In particular, the least-squares solver will perform compensation during the reconstruction only on basis of the combinatorial structure. Nevertheless, incorporation of metric information into the diffusion process may be important for other applications.

4.2. High-Pass Quantization of Dynamic Meshes

It is quite natural to consider our approach of coding geometry also for dynamic meshes. The combinatorial Laplacian stays

constant if the combinatorics do not change during an animation. Hence, the Laplacian only needs to be factored once and can then be used in the reconstruction of any frame of an animation without significant overhead. Therefore, after some overhead for the first frame, each following frame can be decoded on the fly, assuming that the quantized differential coordinates are easy to decode.

Similar to Váša and Petřík [19], we build upon the CoDDyAC algorithm [32] for dynamic mesh compression. The original CoDDyAC algorithm performs a principal component analysis (PCA) of the trajectories of the vertices. After truncating the PCA-basis and expressing the trajectories in the truncated basis, the coefficients of the trajectories are encoded with the EDGEBREAKER algorithm [33], utilizing the parallelogram rule

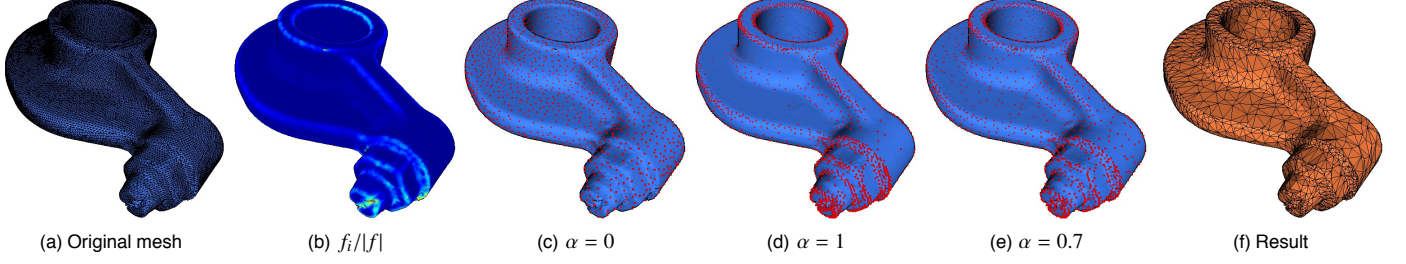


Figure 8: Intrinsic mesh simplification selects a subset of vertices by diffusing a feature measure: (a) Original mesh (40177 vertices). (b) Visualization of the feature measure. (c) Uniform selection. (d) Selection based on feature measure only. (e) Selection based on a mixture of uniform and feature measure. (f) Result (4018 vertices) obtained by connecting the selected vertices.

for prediction and quantization [16]. The differential coordinate-based CoDDyAC algorithm [19] simply applies the Laplacian operator to the trajectory coefficients before encoding them with the EDGEBREAKER algorithm.

We use the same approach, however, instead of utilizing EDGEBREAKER, we perform quantization and error diffusion on the trajectory coefficients, as described in the previous section for static meshes. Figure 7 compares our approach with the differential coordinate-based CoDDyAC variant. As in the case of classical high-pass quantization, our approach based on error diffusion significantly improves the situation. In particular, our approach enables using fewer anchor points. These are expensive to encode in this case, since encoding anchors corresponds to storing a full set of trajectory coefficients. A set of example animations is included with the supplemental material. An interesting observation is that there is a close relationship between prediction based on the parallelogram rule and error diffusion, since the former can be regarded as a simplistic form of error diffusion, where the total quantization error is distributed to a single neighbor, namely the next element in processing order.

5. Intrinsic Mesh Simplification

As a further application, we show in this section how our error diffusion framework can be applied to mesh simplification. The underlying idea is that identifying a subset among the vertices of a mesh is equivalent to performing a black and white coloring of the vertices.

More specifically, we base our approach on TopStoc [20]. This algorithm uses a stochastic approach to select the subset of vertices based on a geometric importance measure of the vertices. The stochastic selection makes the algorithm linear in the vertices. We derive a similar measure, attach it to the vertices, and then apply error diffusion. One may think of the measure as a grayscale value, with error diffusion coloring a few vertices ‘white’, meaning these vertices are selected to represent the geometry.

Algorithm. Let $\mathcal{V}$ denote the set of vertices and K the fraction of vertices to be selected. If K is assigned to each vertex, then error diffusion will produce $K|\mathcal{V}|$ white vertices (Figure 8(a)). Treating all vertices equally, however, is less than ideal, since more vertices than necessary are selected in regions

of low geometric detail, while information is lost in regions of high geometric detail, due to undersampling. Better results can be achieved by selecting the vertices based on a measure of their geometric importance [20]

$$f_i = \frac{1}{2} \left(1 - \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} n_i^T n_j \right), \quad (10)$$

measuring the deviation among normalized vertex normals in the one-ring neighborhood $\mathcal{N}(i)$ of a vertex i (Figure 8(b)). The values of this measure lie in the range $[0, 1]$, with $f_i = 0$ corresponding to flat regions and increasing $f_i > 0$ corresponding to increasingly stronger curved regions of the mesh. Initializing each vertex with $K f_i |\mathcal{V}| / |f|$, where $|f| = \sum_{i \in \mathcal{V}} f_i$ denotes the cumulative sum of the measures, and performing error diffusion then again produces $K|\mathcal{V}|$ white vertices, with vertices in curved regions being preferred (Figure 8(c)). Unfortunately, this is still not optimal. Since the feature function is a local measure, it cannot detect large curved regions that are locally flat. We avoid undersampling of such regions by combining uniform and feature-based selection, through the use of linear weighting, controlled by a parameter $\alpha \in [0, 1]$:

$$w_i(\alpha) = K \left((1 - \alpha) + \alpha \frac{f_i |\mathcal{V}|}{|f|} \right) \quad (11)$$

Once a subset of vertices has been selected, it gets triangulated using the topological clustering described by Boubekeur and Alexa [20].

Results and Discussion. The key parameters of our approach are the adaptivity α , the weight ζ that specifies the fraction of the error that is distributed to the next element in processing order, and the processing order used. Our experimental evaluation shows that α should not be chosen close to one or zero. Good choices typically lie in the range $\alpha \in [0.25, 0.75]$. For the sake of simplicity, we have taken $\alpha = 0.7$ for all our experiments. Next, we evaluated how the choice of the fraction ζ influences the quality of the approximation of the generated simplified mesh. To this end, we compute the root mean square (RMS) distance [31] between the original and the simplified model. We observed that ζ should be chosen close to one for small simplification ratios and close to zero for large simplification ratios. An optimal choice, however, is also depending on the given mesh. As a reasonable default, we have therefore

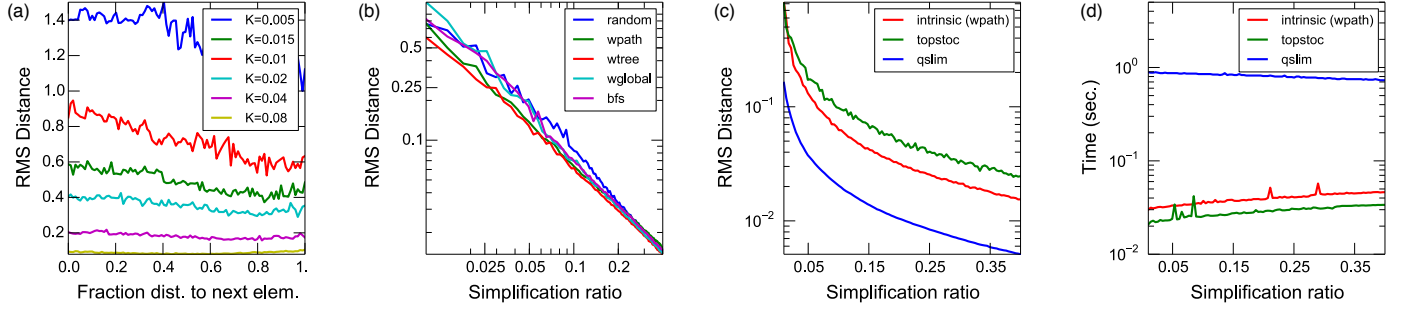


Figure 9: Evaluation of the proposed intrinsic mesh simplification approach. (a) Assessment of the impact of the different processing orders on the Hausdorff error of the simplified mesh. (b) Demonstration of the importance of diffusing the error, for small simplification ratios, partially to the next element in processing order. For different fixed simplification ratios, it is shown how the Hausdorff error changes depending on the fraction ζ of the error that is distributed to the next element. (c) and (d) Comparison with TopStoc [20] and QSLIM [34] in terms of RMS error and computation time.

taken $\zeta = 0.5$ for all our experiments. Figure 9(a) shows the Hausdorff error depending on ζ for the model shown in Figure 8. Evaluation of the different proposed processing orders showed that the paths based on the Warnsdorff-rule have a small advantage over the other orders.

To evaluate how our approach relates to other techniques, we compared it with two state-of-the-art techniques. The first is TopStoc by Boubekeur and Alexa [20], which is similar to our method but uses a stochastic approach to select the subset of vertices. As can be seen in Figure 9(c), our approach based on error diffusion clearly outperforms TopStoc’s stochastic approach. Another advantage is that error diffusion based on the Warnsdorff rule is deterministic: it always produces the same result and almost exactly the desired number of vertices—unlike TopStoc. This improvement comes at the price of a slightly increased processing time of about 30%.

We also compare our approach to QSLIM by Garland and Heckbert [34]. Although already dated, this technique still represents a state-of-the-art quality oriented simplification technique and as such an upper bound on the quality. As expected, it generates significantly better quality simplifications, using significantly more computation time (see Figure 9).

6. Conclusions

We have presented a systematic analysis of different processing orders for error diffusion on surface meshes. We have shown that several fundamental algorithms in mesh processing benefit from incorporating error diffusion. In all cases, adding error diffusion to the original algorithm is trivial and, assuming the linear time path is used, the additional computational complexity is negligible.

In particular, for high-pass quantization, the application of error diffusion changes the results by orders of magnitude, mostly making anchoring of vertices unnecessary. The application on quantizing the coefficients of trajectory data results in promising improvements. It appears these improvements transfer to the recent state-of-the-art versions of this compression scheme [35]. The modification of TopStoc improves the quality of the simplified geometry, at negligible extra cost. We note that

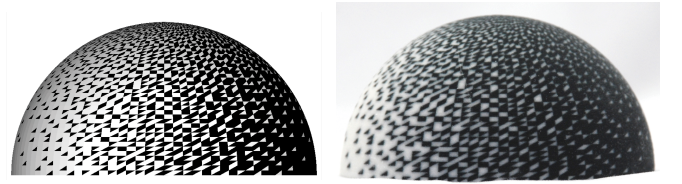


Figure 10: A half sphere mesh with a gray ramp processed by error diffusion on the faces. The images show a rendering and a photograph of a 3D print. Note the worm-like artifacts in the vertical direction.

the usage scenario of algorithms like TopStoc is naturally limited, and QSLIM still is the method of choice for off-line mesh simplification. All applications allow measuring the gain from error diffusion in an objective way and show that the linear time path based on the Warnsdorff rule performs well in practice.

We demonstrated that error diffusion can be useful as a tool to process mesh geometry. Another application is, of course, in computing the quantities needed for physical output of shapes, e.g., providing an optimized implementation of the *dithering* stage of the OpenFab pipeline [36]. With the recent advance in mass market 3D printing, the importance of this area will be hard to overestimate. Applying the ideas to volumetric (or any other type of graph) is also straightforward. This allows generating patterns from graded material properties inside a volume for creating physical output using a two-material printer.

Limitations. We know from error diffusion on image data that the coefficients are extremely important for the spectral properties and, thus, the visual quality of the result. Even in our applications, we can see this effect in the simplification scenario, where the distribution of the vertices is important. We can replicate the importance of distributing the bulk of the error into the element to be processed next. However, more optimization of the coefficients is necessary and an open problem.

To illustrate the problem of the current approach, we show the visual quality of error diffusion on a spherical mesh. Here, we have applied error diffusion to the faces and compensated for the face areas. Figure 10 shows a rendering as well as a photograph of a 3D print. Note the obvious worm-type artifacts along the vertical direction.

Future Work. Clearly, the most pressing future work is generating better distribution coefficients. Note that the optimization of the weights is significantly more involved for the irregular mesh case, because not only are there various different neighborhood combinatorics, but also the set of processed and unprocessed elements in the neighborhood varies (and considering only the combinatorial structure of the problems neglects the area and shape of the mesh elements, which might also prove relevant).

Acknowledgements

We would like to thank Felix Haase for helping us with the implementation and the anonymous reviewers for their comments and feedback. This work has been supported by the ERC through grant ERC-2010-StG 259550 (XSHAPE).

References

- [1] D. E. Knuth, Digital halftones by dot diffusion, *ACM Transactions on Graphics* 6 (4) (1987) 245–273.
- [2] K. T. Knox, Evolution of error diffusion, *Journal of Electronic Imaging* 8 (4) (1999) 422–429.
- [3] V. Ostromoukhov, A simple and efficient error-diffusion algorithm, in: *Proc. SIGGRAPH '01*, 2001, pp. 567–572. doi:10.1145/383259.383326.
- [4] B. Zhou, X. Fang, Improving mid-tone quality of variable-coefficient error diffusion using threshold modulation, *ACM Transactions on Graphics* 22 (3) (2003) 437–444. doi:10.1145/882262.882289.
- [5] J. Chang, B. Alain, V. Ostromoukhov, Structure-aware error diffusion, *ACM Transactions on Graphics* 28 (2009) 162:1–162:8. doi:10.1145/1618452.1618508.
- [6] H. Li, D. Mould, Contrast-aware halftoning, *Computer Graphics Forum* 29 (2) (2010) 273–280.
- [7] W. Matusik, B. Ajdin, J. Gu, J. Lawrence, H. P. A. Lensch, F. Pellacini, S. Rusinkiewicz, Printing spatially-varying reflectance, *ACM Transactions on Graphics* 28 (5) (2009) 128:1–128:9. doi:10.1145/1618452.1618474.
- [8] T. Malzbender, R. Samadani, S. Scher, A. Crume, D. Dunn, J. Davis, Printing reflectance functions, *ACM Transactions on Graphics* 31 (3) (2012) 20:1–20:11. doi:10.1145/2167076.2167078.
- [9] D. Chen, D. I. W. Levin, P. Didyk, P. Sitthi-Amorn, W. Matusik, Spec2Fab: A reducer-tuner model for translating specifications to 3D prints, *ACM Transactions on Graphics* 32 (4) (2013) 135:1–10. doi:10.1145/2461912.2461994.
- [10] M. Skouras, B. Thomaszewski, S. Coros, B. Bickel, M. Gross, Computational design of actuated deformable characters, *ACM Transactions on Graphics* 32 (4) (2013) 82:1–82:10. doi:10.1145/2461912.2461979.
- [11] R. Ulichney, *Digital Halftoning*, MIT Press, 1987.
- [12] P. Alliez, E. de Verdiere, O. Devillers, M. Isenburg, Isotropic surface remeshing, in: *Shape Modeling International*, 2003, pp. 49–58. doi:10.1109/SMI.2003.1199601.
- [13] K. Akeley, P. Haeberli, D. Burns, *tomesh.c: C program on SGI developer's toolbox CD* (1990).
- [14] S.-E. Yoon, P. Lindstrom, V. Pascucci, D. Manocha, Cache-oblivious mesh layouts, *ACM Transactions on Graphics* 24 (3) (2005) 886–893. doi:10.1145/1073204.1073278.
- [15] A. Bogomjakov, C. Gotsman, Universal rendering sequences for transparent vertex caching of progressive meshes, *Computer Graphics Forum* 21 (2) (2002) 137–149. doi:10.1111/1467-8659.00573.
- [16] C. Touma, C. Gotsman, Triangle mesh compression, in: *Graphics Interface*, 1998, pp. 26–34.
- [17] H. Warnsdorff, Des Rösselsprünge einfachste und allgemeinste Lösung, *Schmalkalden*, 1823.
- [18] O. Sorkine, D. Cohen-Or, S. Toledo, High-pass quantization for mesh encoding, in: *Proc. Symposium on Geometry processing (SGP)*, 2003, pp. 42–51.
- [19] L. Váša, O. Petřík, Optimising perceived distortion in lossy encoding of dynamic meshes, *Computer Graphics Forum* 30 (5) (2011) 1439–1449. doi:10.1111/j.1467-8659.2011.02018.x.
- [20] T. Boubekeur, M. Alexa, Mesh simplification by stochastic sampling and topological clustering, *Computers & Graphics* 33 (3) (2009) 241–249. doi:10.1016/j.cag.2009.03.025.
- [21] M. Desbrun, M. Meyer, P. Schröder, A. H. Barr, Implicit fairing of irregular meshes using diffusion and curvature flow, in: *Proc. SIGGRAPH '99*, 1999, pp. 317–324. doi:10.1145/311535.311576.
- [22] M. Alexa, M. Wardetzky, Discrete laplacians on general polygonal meshes, *ACM Transactions on Graphics* 30 (2011) 102:1–102:10. doi:10.1145/2010324.1964997.
- [23] J.-N. Shiao, Z. Fan, Set of easily implementable coefficients in error diffusion with reduced worm artifacts, *Proc. SPIE* 2658 (1996) 222–225. doi:10.1117/12.236968.
- [24] R. W. Floyd, L. Steinberg, An adaptive algorithm for spatial greyscale, *Proceedings of the Society for Information Display* 17 (2) (1976) 75–77.
- [25] L. Velho, J. d. M. Gomes, Digital halftoning with space filling curves, in: *Proc. SIGGRAPH '91*, 1991, pp. 81–90. doi:10.1145/122718.122727.
- [26] M. R. Garey, D. S. Johnson, *Computers and Intractability*, W. H. Freeman and Company, 1979.
- [27] I. Pohl, A method for finding hamilton paths and knight's tours, *Communications of the ACM* 10 (1967) 446–449. doi:10.1145/363427.363463.
- [28] D. Chen, D. Cohen-Or, O. Sorkine, S. Toledo, Algebraic analysis of high-pass quantization, *ACM Transactions on Graphics* 24 (2005) 1259–1282. doi:10.1145/1095878.1095880.
- [29] M. Wardetzky, S. Mathur, F. Kälberer, E. Grinspun, Discrete laplace operators: No free lunch, in: *Proc. Symposium on Geometry processing (SGP)*, 2007, pp. 33–37.
- [30] G. Guennebaud, B. Jacob, et al., *Eigen v3*, <http://eigen.tuxfamily.org> (2010).
- [31] N. Aspert, D. Santa-Cruz, T. Ebrahimi, MESH: measuring errors between surfaces using the hausdorff distance, in: *IEEE International Conference on Multimedia and Expo*, 2002, pp. 705–708. doi:10.1109/ICME.2002.1035879.
- [32] L. Váša, V. Skala, CODDYAC: connectivity driven dynamic mesh compression, in: *3DTV Conference*, 2007, pp. 1–4. doi:10.1109/3DTV.2007.4379408.
- [33] J. Rossignac, Edgebreaker: Connectivity compression for triangle meshes, *IEEE Transactions on Visualization and Computer Graphics* 5 (1) (1999) 47–61. doi:10.1109/2945.764870.
- [34] M. Garland, P. S. Heckbert, Surface simplification using quadric error metrics, in: *Proc. SIGGRAPH '97*, 1997, pp. 209–216. doi:10.1145/258734.258849.
- [35] L. Váša, S. Marras, K. Hormann, G. Brunnett, Compressing dynamic meshes with geometric Laplacians, *Computer Graphics Forum* 33 (2).
- [36] K. Vidimč, S.-P. Wang, J. Ragan-Kelley, W. Matusik, OpenFab: a programmable pipeline for multi-material fabrication, *ACM Transactions on Graphics* 32 (4) (2013) 136:1–12. doi:10.1145/2461912.2461993.