Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2006)
M.-P. Cani, J. O'Brien (Editors)

Progressive Deforming Meshes based on Deformation Oriented Decimation and Dynamic Connectivity Updating

Fu-Chung Huang[†] Bing-Yu Chen[‡] Yung-Yu Chuang[§]

National Taiwan University

Abstract

We present a method for progressive deforming meshes. Most existing mesh decimation methods focus on static meshes. However, there are more and more animation data today, and it is important to address the problem of simplifying deforming meshes. Our method is based on deformation oriented decimation (DOD) error metric and dynamic connectivity updating (DCU) algorithm. Deformation oriented decimation extends the deformation sensitivity decimation (DSD) error metric by augmenting an additional term to model the distortion introduced by deformation. This new metric preserves not only geometric features but also areas with large deformation. Using this metric, a static reference connectivity is extracted for the whole animation. Dynamic connectivity updating algorithm utilizes vertex trees to further reduce geometric distortion by allowing the connectivity to change. Temporal coherence in the dynamic connectivity between frames is achieved by penalizing large deviations from the reference connectivity. The combination of DOD and DCU demonstrates better simplification and triangulation performance than previous methods for deforming mesh simplification.

1. Introduction

Today, more and more high-resolution animated 3D models, also called deforming meshes or time-varying surfaces, are widely used in many applications, such as games and movies. High-resolution models are required to present details and fine structures. However, some details might be unnecessary especially when viewing from a distance. Mesh simplification is a process of eliminating such unnecessary or redundant details from high-resolution 3D models by removing vertices, edges, or faces. By repeatedly applying this process, an animated model can be converted into a set of progressive mesh representing a sequence of 3D meshes with continuous level-of-details (LOD). Due to the removal of some primitives, this process usually distorts the original model. Hence, various metrics have been proposed to measure the deviation of the simplified model from the original one and many mesh simplification methods have been developed to minimize these metrics. However, most of these methods are designed for simplifying static meshes, but not time-varying meshes such animated 3D models or 3D metamorphosis. In this paper, we propose a new method for generating progressive deforming meshes.

To simplify deforming meshes, some previous methods focus on preserving the static connectivity, i.e., the connectivity of the deforming meshes remains unchanged for all frames. In these approaches, the mesh simplification process is only applied to one model, which aggregated features of all frames as a meta-mesh. However, such adaptations are inadequate and the results are often not satisfactory since they do not take time-varying deformation into consideration. Figure 1 shows the last frame of a simplified 3D morphing sequence, in which a horse is morphed to a man. The results of using previous approaches (Figure 1(c, d, e)) have obvious distortion in the hand area. It is because that the features of a horse are not necessarily the features of a man and our method has no such problem (Figure 1(f)).

On contrast to early work with static connectivity, a few recent methods change the connectivity adaptively and dynamically to improve the quality for each simplified mesh of the simplified deforming meshes. These methods start with the mesh of the first frame, and incrementally update the connectivity so that the mesh in the next frame is well ap-

[†] e-mail:jonash@cmlab.csie.ntu.edu.tw
[‡] e-mail:robin@ntu.edu.tw
[§] e-mail:cyy@csie.ntu.edu.tw

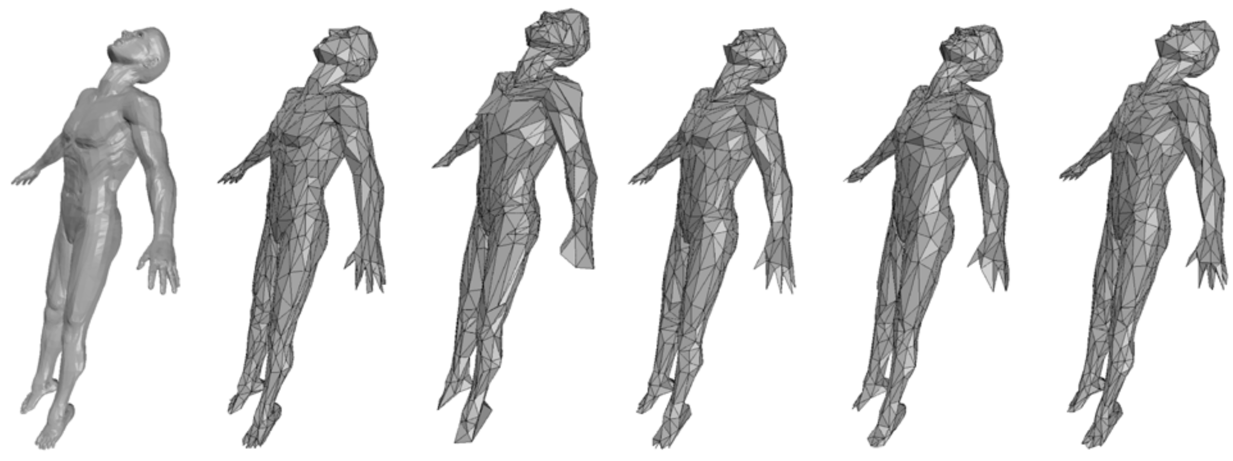


Figure 1: *The last frame of a simplified 3D morphing sequence. (a) The original model. (b) The result of an independent QEM approach. (c) The result of a static connectivity approach from the first frame. (d) The result of DSD. (e) The result of Kircher and Garland's method [KG05]. (f) Our result. Although independent QEM is the best approach to preserve the features, it has popping artifacts in animation. The result of our method preserves more details than pervious methods while maintaining temporal coherency of mesh connectivity.*

proximated. In order to generate good simplified deforming meshes, a great amount of updates is often inevitable, and hence resulting in popping artifacts. On the other hand, if consistency constraints on the connectivity of meshes from frame to frame are imposed, the results have better *temporal coherence*. However, geometric errors may be propagated and accumulated with these approaches. The fundamental problem is that the model in either the first or arbitrary frame does not necessarily represent a good compromise between the individual mesh distortion and connectivity updates for in-between meshes.

Our goal is *to minimize mesh distortion while maximizing temporal coherence* for deforming mesh decimation. In other words, our method attempts to find a sequence of meshes which well approximate the original meshes and the amount of frame-to-frame connectivity updates is minimized. We first introduce a deformation oriented decimation (DOD) metric to improve static connectivity approaches (Section 3.1). In addition to the geometric error metric, DOD metric uses the first order derivatives of the edge lengthes to measure the deformation degree of deforming meshes. This DOD scheme is used to obtain the initial progressive deforming meshes with static connectivity. Next, we introduce a dynamic connectivity updating (DCU) method which utilizes vertex trees to further reduce geometric distortion by altering connectivity of meshes (Section 3.2). Temporal coherence of connectivity between frames is achieved by penalizing large deviations from a reference connectivity. Results (Section 4) shows that the proposed method has better performance than previous methods for deforming mesh simplification.

2. Related work

In this section, we review related work in three groups: multiresolution meshes, simplification of deforming meshes and compression of deforming meshes.

2.1. Multiresolution meshes

There are a plethora of papers on generating multi-resolution meshes, notably through re-meshing and simplification. Re-meshing approaches [CSAD04, EDD*95] attempt to obtain a good re-sampling over the original mesh surface. These methods can not be directly applied to time-varying surfaces, since they do not take the temporal coherence into account. Simplification techniques choose a primitive causing the least distortion measured by some metrics, and conduct primitive elimination operations such as vertex-removal [SZL92], vertex-clustering [CVM*96], and edge-collapsing [GH97, Hop96, LT98]. For edge-collapsing architecture, the sequence of simplification process forms a tree structure, called vertex tree. A vertex tree can be used for selective refinement or coarsening for arbitrary view-dependent multiresolution model [XV96, Hop97, LE97]. Traditional mesh simplification algorithms work fine on a single static model. However, as re-meshing, it does not consider temporal coherence and hence can not be directly used for deforming meshes.

2.2. Simplification of deforming meshes

Recently, some mesh simplification methods have been proposed for deforming meshes. Mohr and Gleicher [MG03] proposed a deformation sensitive decimation (DSD) method, which adapts the QEM [GH97] algorithm, to construct a meta-mesh by summing the quadric errors incurred by all frames. The meta-mesh uses the aggregated error as the guide to find an optimal primitive elimination sequence to simplify itself. DeCoro and Rusinkiewicz [DR05] proposed a method of weighting possible configuration of poses with probabilities. With articulated meshes, skeleton transformation is incorporated into standard QEM algorithm, and users must specify the probability distribution for each joint. Shamir and Pascucci [SP01] also used the QEM algorithm as a base simplification module. By extracting high frequency transformation, simplification is applied to the base mesh and the results are acquired by an inverse transformation on the simplified base mesh.

Keeping static connectivity for all frames produces a moderate approximation. With this adaptation, a very simple data structure is used for the representation and the quality of the simplified animation is satisfactory in general. However, because this approach makes a compromise among all frames, it may prefer to preserve a less prominent feature over a more important feature in some frame because the less prominent feature is more important in other frames. This is especially common in 3D metamorphosis animations. Hence, important features at certain frames could be sacrificed by this compromise. The phenomena can be seen in Figure 1.

Since methods with static connectivity can not preserve all features when more primitives are removed, an alternative way is to change the connectivity dynamically during the animation. Shamir *et al.* [SBP00, SP01] designed a scheme for simplifying deforming meshes while changing the connectivity dynamically. In their method, Time-dependent Directed Acyclic Graph (TDAG) is introduced by merging each individual simplified model of each frame into a unified graph. TDAG is a data structure that stores the life time of a vertex, which is queried for the connectivity updating.

Kircher and Garland [KG05] used edge-swap operations to dynamically update the connectivity. The simplified model for the next frame is obtained by a sequence of edge-swap operations from the simplified model of the current frame. Inadequate error propagation found in Shamir's approach [SBP00] is overcome by applying only valid and *beneficial* edge-swap operations. However, for extremely simplified deforming meshes, there might be a huge amount of connectivity updates which cause popping artifacts.

2.3. Compression of deforming meshes

Another way to reduce the size of the sequence of deforming meshes is through data compression. Lengyel [Len99] proposed a compression method for deforming meshes by decomposing the time-varying geometries to **SVG** matrices. Alexa and Müller [AM00] used Singular Value Decomposition (SVD) to find the principle components of deforming meshes. Then, the lossless or lossy compression of the deforming meshes can be controlled by either preserving all components or discarding some less important components. Ibarria and Rossignac [IR03] showed how to use predictors, ELP and Replica, to compress deforming meshes. As an extension of the geometry image [GGH02], Briceño *et al.* [BSM*03] used the RGB channels as the XYZ coordinates to compress the time-varying deforming meshes as compressing a video sequence. Hence, many techniques for video compression could be applied. James and Twigg [JT05] developed a method to automatically find the bones and vertex weights which are used for efficient hardware rendering and excellent mesh data compression.

3. Algorithm

Our algorithm consists of two components: *Deformation Oriented Decimation (DOD)* and *Dynamic Connectivity Updating (DCU)*. In this section, we first describe our DOD method for deforming mesh decimation with a static connectivity. Next, we use DCU algorithm to reduce errors by allowing adaptive connectivity while maintaining temporal coherence. Finally, some implementation details are discussed.

3.1. Deformation oriented decimation

Our DOD algorithm is based on Garland and Heckbert's QSlim algorithm which has been proven efficient and effective for static mesh decimation. QSlim iteratively selects an edge $(\mathbf{v}_i, \mathbf{v}_j)$ with the minimum contraction cost to collapse and replace this edge with a new vertex $\mathbf{u}$ which minimizes the contraction cost. To measure the contraction cost for an edge, Qslim utilizes the quadratic error metric (QEM) [GH97], which measures the total squared distance of a vertex to the two sets of planes $\mathbf{P}(\mathbf{v}_i)$ and $\mathbf{P}(\mathbf{v}_j)$ adjacent to $\mathbf{v}_i$ and $\mathbf{v}_j$ respectively. A plane can be represented with a 4D vector $\mathbf{p}$, consisting of the plane normal and the distance to the origin. Hence, the squared distance of a vertex $\mathbf{v}$ to a plane $\mathbf{p}$ equals $\mathbf{v}^{\mathrm{T}}(\mathbf{p}\mathbf{p}^{\mathrm{T}})\mathbf{v}$. The QEM error function Δ_{ij} for a vertex $\mathbf{v}$ to replace the edge $(\mathbf{v}_i, \mathbf{v}_j)$ is

$$\begin{aligned}\Delta_{ij}(\mathbf{v}) &= \sum_{\mathbf{p}\in\mathbf{P}(\mathbf{v}_i)} \mathbf{v}^{\mathrm{T}}(\mathbf{p}\mathbf{p}^{\mathrm{T}})\mathbf{v} + \sum_{\mathbf{p}\in\mathbf{P}(\mathbf{v}_j)} \mathbf{v}^{\mathrm{T}}(\mathbf{p}\mathbf{p}^{\mathrm{T}})\mathbf{v} \\ &= \mathbf{v}^{\mathrm{T}}\mathbf{Q}_i\mathbf{v} + \mathbf{v}^{\mathrm{T}}\mathbf{Q}_j\mathbf{v}. \qquad (1)\end{aligned}$$

Garland also suggests using an area-weighted quadric error metric for better results [Gar99] and defines the QEM error function as:

$$\Delta_{ij}(\mathbf{v}) = \mathbf{v}^{\mathrm{T}}(w_i\mathbf{Q}_i + w_j\mathbf{Q}_j)\mathbf{v} = \mathbf{v}^{\mathrm{T}}\mathbf{Q}_{ij}\mathbf{v}, \qquad (2)$$

where w_i is the total area of triangles adjacent to $\mathbf{v}_i$ and w_j is defined similarly. Hence, the QEM cost QEM_{ij} for contracting an edge $(\mathbf{v}_i, \mathbf{v}_j)$ is defined as $\Delta_{ij}(\mathbf{u}_{ij})$, in which $\mathbf{u}_{ij}$ is the vertex minimizing $\Delta_{ij}(\mathbf{v})$. QSlim simplifies a mesh by iteratively finding the edge $(\mathbf{v}_i, \mathbf{v}_j)$ with the minimum QEM_{ij}, performing an edge-collapse operation to replace

$(\mathbf{v}_i, \mathbf{v}_j)$ with a new vertex $\mathbf{u}_{ij}$ and updating the edge contraction costs related to $\mathbf{u}_{ij}$ until the desired vertex count m is reached.

To extend QEM to handle deforming meshes, a naïve way is to use QEM to obtain an edge-collapse sequence for the first frame, and then apply this sequence to all frames. Since all frames use the same edge-collapse sequence, they have the same connectivity. The disadvantage of this approach is obvious. Features of other frames might be removed if they are not features in the first frame. The deformation sensitive decimation (DSD) algorithm addresses this problem by summing QEM costs across all frames [MG03]. The DSD contraction cost for an edge $(\mathbf{v}_i, \mathbf{v}_j)$ is defined as

$$\mathsf{DSD}_{ij} = \sum_{t=1}^{f} \mathsf{QEM}_{ij}^t = \sum_{t=1}^{f} \mathbf{u}_{ij}^{t\ \mathrm{T}} \mathbf{Q}_{ij}^t \mathbf{u}_{ij}^t, \tag{3}$$

where $\mathbf{u}_{ij}^t$ minimizes the QEM cost QEM_{ij}^t for the edge $(\mathbf{v}_i, \mathbf{v}_j)$ at frame t. Hence, DSD tends to preserve edges that are geometric features more often in the animation.

The QEM error metrics used by DSD only concerns with features of geometry in the spatial domain but not features of deformation in the temporal domain. As a result, geometric features are often well preserved but regions with high deformation may not. To address DSD's ignorance to deformation information, we propose a deformation oriented decimation (DOD) metric which incorporates a deformation cost into the DSD metric.

We design our deformation cost ξ_{ij} so that an edge is less likely to be contracted if it has larger deformation and belongs to deformation areas more often. We measure the deformation of an edge by its average edge length change. The average edge length change for an edge $(\mathbf{v}_i, \mathbf{v}_j)$ is defined as

$$\overline{\Delta l}_{ij} = \sum_{t=1}^{f-1} \Delta l_{ij}^t, \tag{4}$$

where $\Delta l_{ij}^t = |l_{ij}^{t+1} - l_{ij}^t|$ is the edge length change from frame t to $t+1$ and l_{ij}^t is the edge length for the edge $(\mathbf{v}_i, \mathbf{v}_j)$ at frame t. An edge with larger $\overline{\Delta l}_{ij}$ implies that this edge deforms more often in the animation. Therefore, if we contract it, then the deformation performed by this edge will be lost. Hence, we should prefer to contract edges with smaller $\overline{\Delta l}_{ij}$ first.

To be consistent with area-weighted QEM, we assign higher weights to the edges belonging to the triangles of larger areas. Hence, we use the average area $\overline{A}_{ij}$ as the weight,

$$\overline{A}_{ij} = \sum_{t=1}^{f} A_{ij}^t, \tag{5}$$

where A_{ij}^t is the sum of areas of the two triangles sharing edge $(\mathbf{v}_i, \mathbf{v}_j)$ at frame t. Thus, we define the deformation cost ξ_{ij} for an edge $(\mathbf{v}_i, \mathbf{v}_j)$ as follows:

$$\xi_{ij} = w_{ij} * \overline{A}_{ij} * (\overline{\Delta l}_{ij})^2, \tag{6}$$

where w_{ij} is another weight defined as

$$w_{ij} = \frac{\max_t \Delta l_{ij}^t - \overline{\Delta l}_{ij}}{\sqrt{\frac{1}{f-1} \sum_{t=1}^{f-1} (\Delta l_{ij}^t - \overline{\Delta l}_{ij})^2}}. \tag{7}$$

The denominator of the above equation is the standard deviation of length changes. This weight represents the normalized maximum deformation and helps to preserve the edge with irregular and sudden large deformation. The introduction of this weight is to reflect the fact that we usually pay more attention to unexpected and sudden deformation.

Finally, We define the DOD error metric for contracting an edge $(\mathbf{v}_i, \mathbf{v}_j)$ as the sum of its DSD cost DSD_{ij} and its associated deformation cost ξ_{ij}:

$$\mathsf{DOD}_{ij} = \mathsf{DSD}_{ij} + \xi_{ij}. \tag{8}$$

The decimation process is similar to DSD algorithm except that DOD metrics are used instead. Once the edge $(\mathbf{v}_i, \mathbf{v}_j)$ has been selected to be contracted, the new vertex $\mathbf{u}_{ij}^t$ is found using the same approach as QEM to minimize geometric error for every frame t.

QEM measures area-weighted geometric distortions while the deformation cost measures area-weighted deformation amount in the animation. Hence, QEM preserves geometric features (DSD_{ij}) while the deformation cost (ξ_{ij}) preserves highly deformed areas, and the combination of the two terms gives simplification that preserves both spatial and temporal features simultaneously. Note that our deformation cost (Equation 6) has a consistent form with QEM

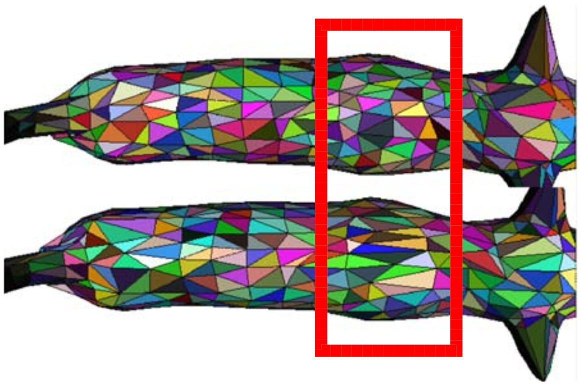


Figure 2: *The simplified walking dog animation. The top row shows the simplified mesh with our DOD method. The bottom row is the result using the DSD method. The highlighted area demonstrates that our DOD method preserves more triangles on the area having more deformation around dog's neck.*

cost. Hence. both costs have similar scales and the relative weight between them is set to 1 as shown in Equation 8.

Figures 2 show comparisons of our DOD method and the DSD method for a walking dog animation. In this animation, the area around dog's neck has more deformation and needs more triangles to represent it faithfully. Our DOD method does preserve more necessary connectivity than the DSD method.

3.2. Dynamic connectivity updating

The DOD algorithm in the previous section simplifies a deforming mesh with static connectivity. Hence, only the vertex positions can be changed across frames but not connectivity. For skinning animations, it is typically sufficient to use a fixed static connectivity. However, for animations with extremely non-rigid deformation, such as 3D metamorphosis animations, decimation using static connectivity is often inappropriate. In 3D metamorphosis, a source mesh is transformed into a target mesh. Both meshes are usually very different and the features of these two meshes are often located at different places. Hence, a static connectivity often fails to capture features across frames. For this situation, adaptive connectivity is preferred. However, at the same time, we do not want to have frequent connectivity change which might incur popping artifacts. To cope with the problem, the idea is to minimize the geometric error while maintaining smooth temporal connectivity changes.

The edge-collapse operations in the previous section implicitly form a binary tree structure since each edge-collapse operation merges two vertices $\mathbf{v}_i$, $\mathbf{v}_j$ into a new vertex $\mathbf{u}_{ij}$. Hence, we can record the sequence of edge collapses to construct a tree called *vertex tree*. Figure 3 gives an example of vertex tree. In a vertex tree, all leaf nodes together represents the full-resolution model. Any cut through the intermediate nodes forms a mesh at a specific resolution. We call such a cut *vertex front*, as illustrated in Figure 3. Hence, a vertex front $\mathbf{F}$ is a set of nodes $\mathbf{n}$ of the vertex tree. For the static mesh decimation problem, we want to find a vertex front of size m with the minimum geometric error. Hence, we can formulate the mesh decimation problem as the following optimization problem:

$$\min_{\mathbf{F}\in\Phi_m} \sum_{\mathbf{n}\in\mathbf{F}} \eta(\mathbf{n}), \tag{9}$$

where Φ_m is the set of all possible vertex fronts whose size are m and $\eta(\mathbf{n})$ is the cost associated with the node $\mathbf{n}$, i.e., , the contraction cost for the edge collapse to form the vertex corresponding to $\mathbf{n}$.

To extend the above optimization problem to handle an animation sequence, independent QEM approach solves the optimization for each frame individually without considering the temporal coherence in connectivity. This strategy however causes popping artifacts and many connectivity updates. As stated earlier, our goal is to minimize geometric

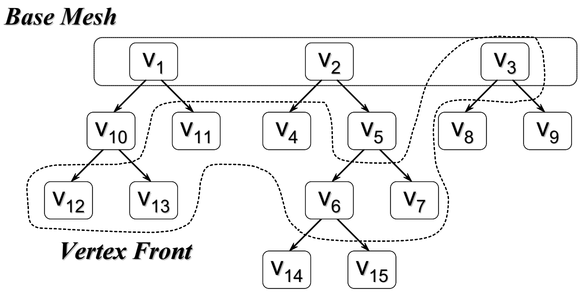


Figure 3: *Illustration of the vertex tree adopt from Hoppe's paper [Hop97]. The roots of the vertex forest collectively represent a base mesh. Any cut through the intermediate nodes forms a decimated mesh.*

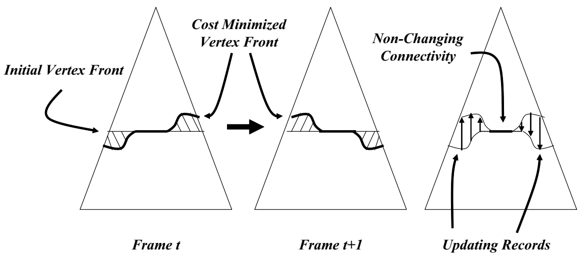


Figure 4: *Vertex front transformed from frame t to frame $t+1$. Because our method seeks for a different vertex front for each frame, the correspondence must be built such that we can update the connectivity. On the right most, two portions are computed: non-changing connectivity and necessary updating records.*

distortion while maintaining the temporal coherence. Thus, given a distance function $D(\mathbf{F}_1, \mathbf{F}_2)$ that measures connectivity discrepancy between two frames, we attempt to find a sequence of meshes with low geometric distortion as well as low connectivity discrepancy between consecutive frames. This problem can then be formulated as an optimization problem:

$$\min_{\mathbf{F}^1,\ldots,\mathbf{F}^f\in\Phi_m} \left(\sum_{t=1}^{f} \sum_{\mathbf{n}\in\mathbf{F}^t} \eta(\mathbf{n}) + k \sum_{t=1}^{f-1} D(\mathbf{F}^t, \mathbf{F}^{t+1}) \right), \tag{10}$$

where k is a user-specified constant to control the degree of connectivity smoothness between consecutive frames.

The distance function $D(\mathbf{F}_1, \mathbf{F}_2)$ is defined as what updating is necessary to transform from the vertex front $\mathbf{F}_1$ to the vertex front $\mathbf{F}_2$. Updating is composed of edge collapses and vertex splits, as illustrated in Figure 4. We first define the distance $D(n, \mathbf{F}_2)$ from a node n of $\mathbf{F}_1$ to the vertex front $\mathbf{F}_2$ as the quadratic height difference in the vertex tree from n to n^*; here, n^* is a node of $\mathbf{F}_2$, can reach n by edge collapses or vertex splits, and has the maximal height difference to n in the vertex tree. The quadratic term prevents the solu-

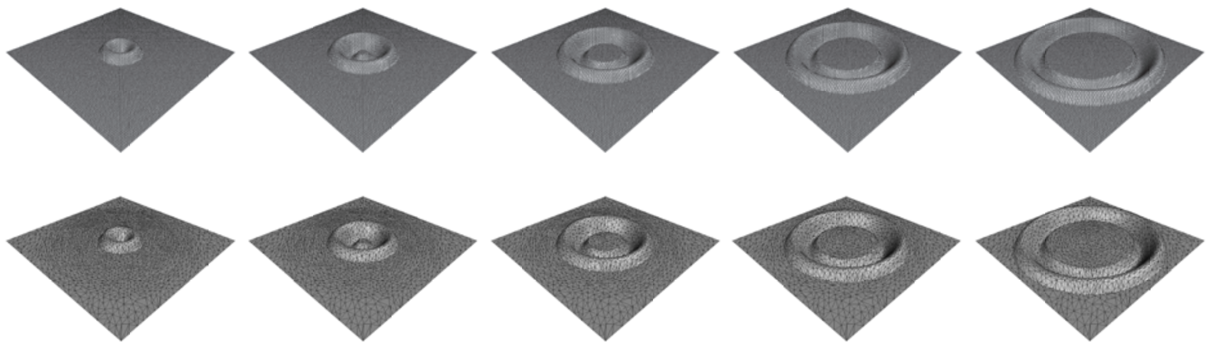


Figure 5: *A 50-frame* wave *animation spreading out from the center to the boundary. Top: The original sequence with 10,210 vertices and 20,000 faces. Bottom: 2,040-vertices and 3,982-faces approximation.*

tion from being overly coarsened or refined at a certain location of the mesh. Finally, the distance $D(\mathbf{F}_1, \mathbf{F}_2)$ is defined as $\sum_{n \in \mathbf{F}_1} D(n, \mathbf{F}_2)$.

However, the optimization defined in Equation 10 is a huge combinatorial optimization problem. We had attempted to solve it by a genetic algorithm but did not have satisfactory results. It is because solutions often get stuck in local minimums. Hence, we simplify Equation 10 and optimize the approximated problem instead. The idea is to first find a reference connectivity represented by a vertex front $\overline{\mathbf{F}}$ as the initial connectivity for all frames. Each frame is then optimized separately by altering $\overline{\mathbf{F}}$ to further reduce geometric error. However, to maintain temporal coherence, we do not like a vertex front too far away from $\overline{\mathbf{F}}$. Thus, we formulate the approximated minimization problem as:

$$\min_{\mathbf{F} \in \Phi_m} \left(\sum_{\mathbf{n} \in \mathbf{F}} \eta(\mathbf{n}) + k * D(\mathbf{F}, \overline{\mathbf{F}}) \right). \tag{11}$$

For this reduced optimization, we first perform DOD algorithm to obtain the optimal vertex front $\overline{\mathbf{F}}$ and set up a vertex tree structure. For each frame t, we generate a vertex tree with the tree structure that DOD algorithm builds. However, the cost associated with each node of the tree is calculated using QEM. That is, the collapse sequence is universally determined by DOD, but the costs are calculated individually for each frame's own vertex tree using QEM. After building the vertex tree for each frame, the optimal vertex front is found using a greedy method. Although our algorithm is only suboptimal, it works very well in practice and gives better results than previous methods.

An alternative approximate solution to Equation 10 would be to solve for the first frame by minimize geometric cost and to use the result as the reference connectivity to optimize for the next frame. Hence, for each frame, we minimize the geometric distortion and the connectivity discrepancy from the previous frame. Similar to Kircher and Garland's method [KG05], this strategy does not use the information of the entire sequence and has similar disadvantages.

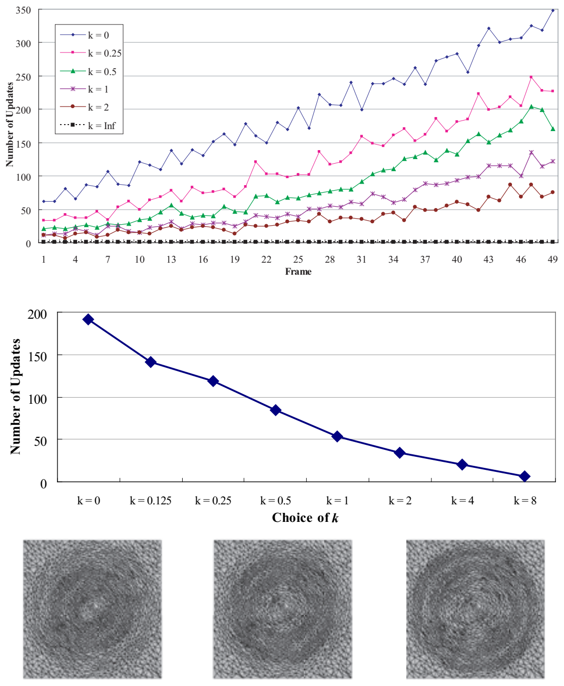


Figure 6: *The effects of different k for the spreading wave animation using the DOD+DCU approach. Top: The number of updates in each frame at different k. Middle: The average number of updates for every doubling k. Bottom: The illustration of the connectivity changes from frame to frame.*

The coefficient k in Equation 11 controls the degree of temporal smoothness; larger k gives a solution closer to DOD and smaller k gives a solution closer to QEM (not

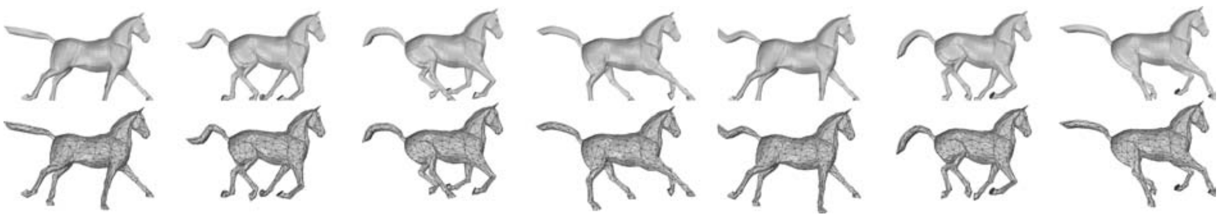


Figure 7: *A* horse-gallop *animation with 48 frames. Top: The original animation with 8,431 vertices and 16,843 faces. Bottom: 800-vertices and 1,588-faces approximation.*

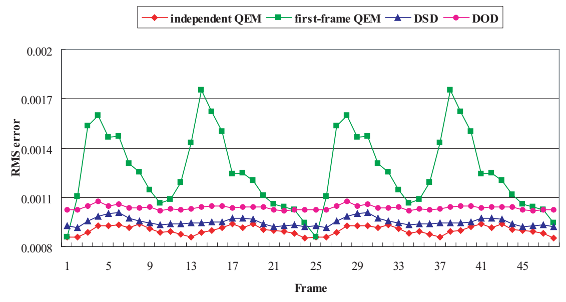


Figure 8: *Comparisons of independent QEM, first-frame QEM, DSD, and DOD for an extreme simplification for the horse-gallop animation.*

the same QEM because the structure of the vertex tree is from DOD but the node costs are from QEM). We use the wave spreading example in Figure 5 to illustrate the impact of k. Figure 6 shows the relationship between the choice of k and the number of average connectivity updates. The result shows that the relationship is approximately linear with halving or doubling k. This property holds for all examples we have tested.

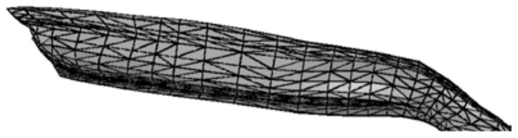
3.3. Implementation

Our algorithm can be summarized into the following steps:

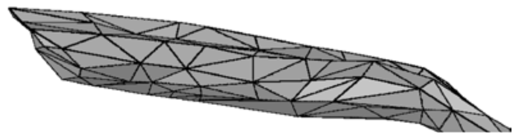
1. Use DOD to generate a sequence of edge-collapse operations, build the structure of the vertex trees and create the initial vertex front.
2. For each frame, build the vertex trees using DOD's structure and fill in the cost for each node using QEM's edge contraction cost.
3. Modify the initial vertex front for each frame to minimize Equation 11.

For Step 1, we modify QSlim 2.1 to incorporate the DOD metric and build the vertex trees. For each edge, we record its length and the areas of the two triangles sharing it. Then, we perform the following procedure:

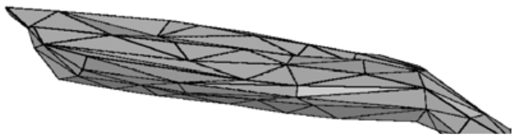
- Compute the DOD cost for every edge, and insert these edges into a priority queue.



(a) original mesh



(b) simplified mesh using DOD



(c) simplified mesh using DSD

Figure 9: *The tail part of the simplified horse animationin Figure 7, in which the tail keeps swinging up and down. Our DOD method gives better triangulation on the tail than the DSD method.*

- Iteratively select and contract an edge with the least DOD cost, update every frame, evaluate the new DOD cost, and insert these new edges back to the priority queue.
- Go to the previous step until reaching the base mesh.

At this point, we have a sequence of edge-collapses which can be used to build the vertex tree and compute the initial vertex front.

For Step 2, conceptually, we have to calculate the geometric cost for every node of the vertex tree for each frame. However, every edge-collapse operation in Step 1 has actually already recorded such information.

For Step 3, we want to modify the vertex front for each frame to minimize Equation 11. Similar to Hoppe's solution [Hop96], we use a priority queue to minimize the cost of vertex front modification. It dramatically speeds up our optimization.

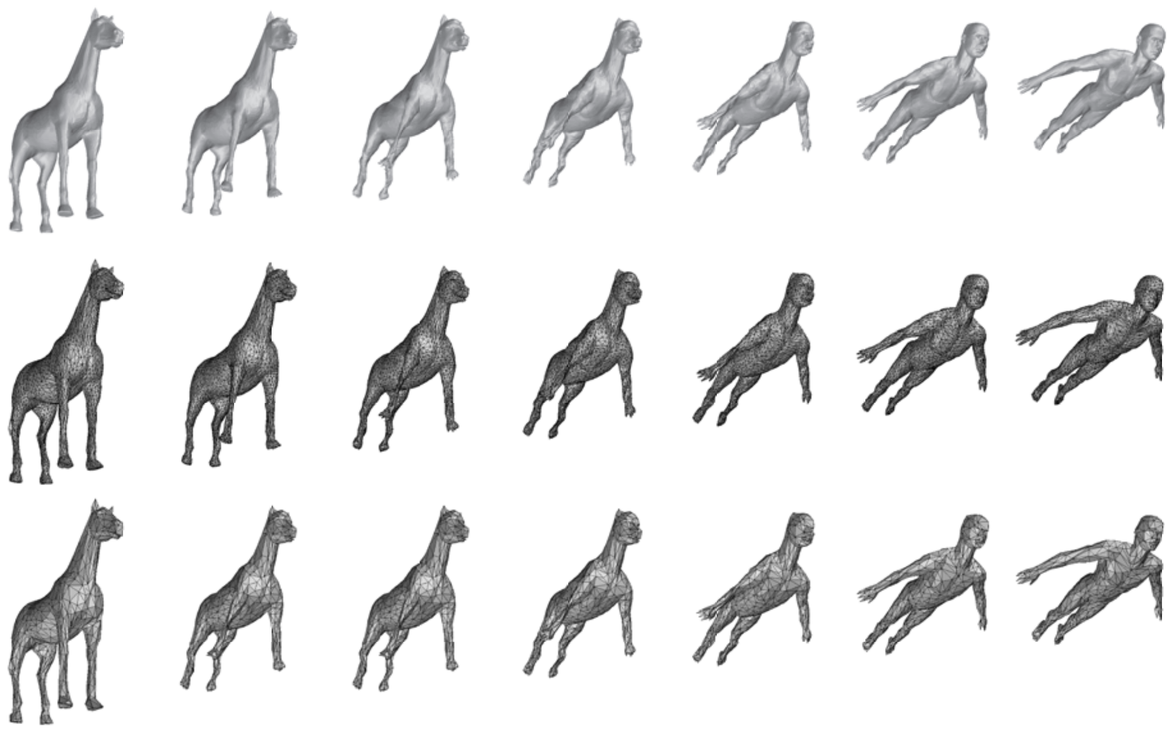


Figure 10: A horse-to-man *morphing animation with 200 frames. Top: The original sequence. Middle: 3,200-vertices approximation. Bottom: 800-vertices approximation.*

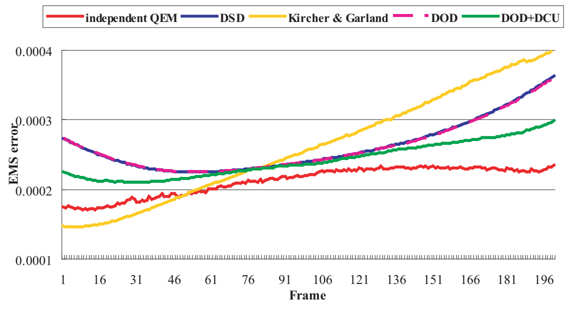


Figure 11: *Comparisons of geometric errors for the horse-to-man morphing sequence. The average geometric error is 2.610 for Kircher and Garland,2.626 for DSD,2.617 for DOD, and 2.433 for DOD+DCU, all in* 10^{-4}.

4. Results

We first compare our DOD method to first-frame QEM and DSD methods in terms of geometric errors for simplifying the horse-gallop animation sequence from 8,431 vertices to 800 vertices (Figure 7). The geometric errors were evaluated using Metro [CRS98]. Figure 8 shows the results. As a reference, the red curve shows the result of applying the QEM method independently to every frame. The first-frame QEM approach (green curve) performs worst. Our result (magenta curve) are much better than the first-frame QEM approach but slightly worse than the DSD method (blue curve). It is because that the DSD method solely focuses on minimizing geometric costs while our DOD method also pays attention to deformation that is not reflected in this experiment. Hence, purely in terms of geometric errors, our DOD method can be outperformed by the DSD method. However, DOD preserves more deformation and provides better triangulation. Note that, though the independent QEM has the best performance in terms of geometric error, the resulted animation has severe popping artifacts.

Figure 9 shows that our DOD algorithm provides better triangulation for horse's tail since it has more deformation. In practice, our DOD method also avoids sliver triangles because the introduced deformation cost penalizes the changes on long edges and more weights are assigned to non-sliver triangles.

Next, we compare DOD, DOD+DCU, DSD and Kircher and Garland's method [KG05] for a 3D metamorphosis example, *horse-to-man* (Figure 10). Figure 11 shows the results. Again, the independent QEM provides a reference

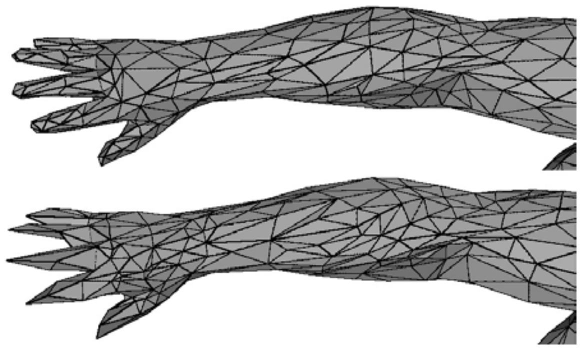


Figure 12: *Comparisons of DOD+DCU (top row) and Kircher and Garland's method (bottom row). In this figure, we illustrate the details for the last frame of the* horse-to-man *animation.*

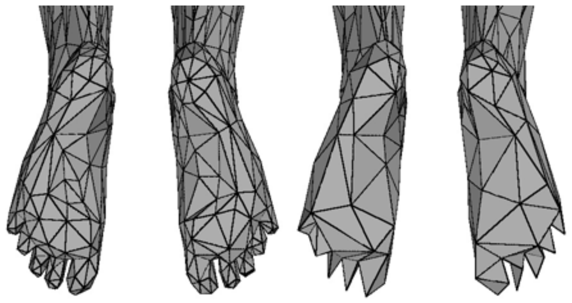


Figure 13: *Comparisons of DOD+DCU (left) and Kircher and Garland's method (right). This is the foot part of the last frame of the* horse-to-man *animation, in which our method successfully preserves more details.*

(red curve). Compared to the DSD method (blue curve), our DOD method (magenta curve) has almost identical performance with the DSD (blue curve) method in this example. However, the DOD+DCU approach (green curve) generally has a lower error than the DSD method. It shows that the DCU method does improve the geometric error since it allows the connectivity to adapt to features better. Kircher and Garland's method gives less distortion at the beginning, but the distortion gradually grows up (orange curve). It is because that it uses the first frame as the initial connectivity and repeatedly warps the connectivity of the current frame to the next frame.

Figures 12 and 13 show the difference of the last frame for the *horse-to-man* animation using our method and Kircher and Garland's method. Our method obviously adjusts to features better than their method. This is, however, partly because our method uses information of the entire sequence. One the other hand, Kircher and Garland's method only uses information from the next frame. Hence, their method is better suited for the situations when the deformation is updated incrementally such as interactive pose editing or physical simulation. However, when the whole sequence is known in advance, our method has better performance.

Figures 14 and 15 show more examples on various animations. Figure 14 demonstrates an example of simplifying a facial expression animation. Even after removing 95% of vertices, the simplified meshes can still be rendered faithfully to the rendering of original ones. Figure 15 shows another example for skinning animation.

The experiments were performed on a computer with an AMD64 2.0GHz CPU and 12GB memory. The computation time of calculating the edge cost is 54.31 seconds for the *horse-to-man* animation, which has 200 frames and 52,461 edges for each frame. Constructing the complete vertex tree takes 445.07 seconds for the 200 meshes in the animation. The minimization of Equation 11 takes 47.60 seconds.

5. Conclusions and future work

In this paper, we propose a method for progressive deforming meshes using the DOD and DCU methods. The DOD method extends the DSD formulation by augmenting an additional deformation cost. The results show a better simplification and triangulation than DSD. The DCU framework is proposed for adapting connectivity of the deforming meshes by utilizing the vertex tree originally designed for view-dependent multi-resolution mesh. Our method is easy to implement and generally has lower geometry error than previous methods.

There are several interesting research directions we want to explore. First, the position of the newly generated vertex after edge-collapsing is not the optimal. For the cases which have near planar surfaces during the animation, the vertex position may be drifted. Second, our DCU method is not an optimal one. We expect to find a way to optimize Equation 10. Third, we expect to extend the DOD algorithm to an incremental algorithm. When little information is known in advance about the deformation, we can incrementally change connectivity while still preserving the deforming area well. Finally, compression and hardware acceleration for progressive deforming meshes are interesting topics as well.

Acknowledgements

We wish to thank Robert W. Sumner and Jovan Popović for providing the horse-gallop animation, Alla Sheffer for the horse-to-man morphing data, and Li Zhang for the facial expression animation. We also would like to thank the reviewers for their pertinent comments. This work was supported in part by the National Science Council of Taiwan under Grant No. NSC93-2213-E-002-084 and NSC94-2213-E-002-097.

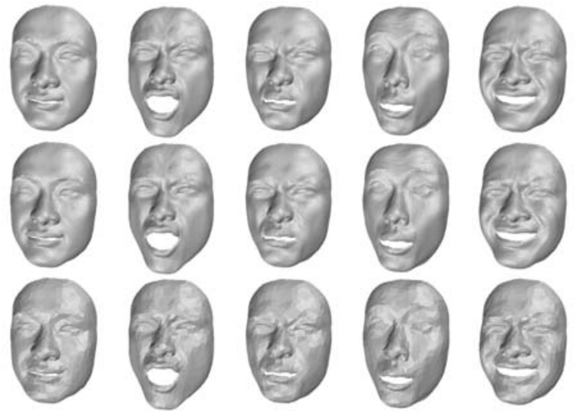


Figure 14: *A facial expression animation with 192 frames. Top: The original sequence with 23,725 vertices and 46,853 faces. Middle: 3,201-vertices and 5,825-faces approximation using DOD+DCU approach. Bottom: 1,200-vertices and 1,874-faces approximation.*

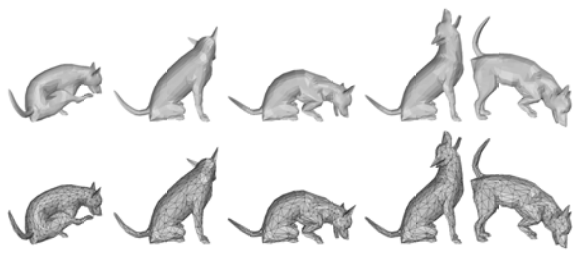


Figure 15: A dog *animation with 111 frames. Top: The original animation with 4,070 vertices and 8,136 faces. Bottom: 800-vertices and 1,596-faces approximation.*

References

[AM00] ALEXA M., MÜLLER W.: Representing animations by principal components. *Computer Graphics Forum (Eurographics 2000 Conference Proceedings) 19*, 3 (2000), 411–418.

[BSM*03] BRICEÑO H. M., SANDER P. V., MCMILLAN L., GORTLER S., HOPPE H.: Geometry videos: a new representation for 3D animations. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), pp. 136–146.

[CRS98] CIGNONI P., ROCCHINI C., SCOPIGNO R.: Metro: measuring error on simplified surfaces. *Computer Graphics Forum 17*, 2 (1998), 167–174.

[CSAD04] COHEN-STEINER D., ALLIEZ P., DESBRUN M.: Variational shape approximation. *ACM Transactions on Graphics (SIGGRAPH 2004 Conference Proceedings) 23*, 3 (2004), 905–914.

[CVM*96] COHEN J., VARSHNEY A., MANOCHA D., TURK G., WEBER H., AGARWAL P., BROOKS F., WRIGHT W.: Simplification envelopes. In *ACM SIGGRAPH 1996 Conference Proceedings* (1996), pp. 119–128.

[DR05] DECORO C., RUSINKIEWICZ S.: Pose-independent simplification of articulated meshes. In *Proceedings of Symposium on Interactive 3D Graphics and Games* (2005), pp. 17–24.

[EDD*95] ECK M., DEROSE T., DUCHAMP T., HOPPE H., LOUNSBERY M., STUETZLE W.: Multiresolution analysis of arbitrary meshes. In *ACM SIGGRAPH 1995 Conference Proceedings* (1995), pp. 173–182.

[Gar99] GARLAND M.: *Quadric-based polygonal surface simplification.* PhD thesis, Carnegie Mellon University, 1999.

[GGH02] GU X., GORTLER S. J., HOPPE H.: Geometry images. In *ACM SIGGRAPH 2002 Conference Proceedings* (2002), pp. 355–361.

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *ACM SIGGRAPH 1997 Conference Proceedings* (1997), pp. 209–216.

[Hop96] HOPPE H.: Progressive meshes. In *ACM SIGGRAPH 1996 Conference Proceedings* (1996), pp. 99–108.

[Hop97] HOPPE H.: View-dependent refinement of progressive meshes. In *ACM SIGGRAPH 1997 Conference Proceedings* (1997), pp. 189–198.

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

[JT05] JAMES D. L., TWIGG C. D.: Skinning mesh animations. *ACM Transactions on Graphics (SIGGRAPH 2005 Conference Proceedings) 24*, 3 (2005), 399–407.

[KG05] KIRCHER S., GARLAND M.: Progressive multiresolution meshes for deforming surfaces. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2005), pp. 191–200.

[LE97] LUEBKE D., ERIKSON C.: View-dependent simplification of arbitrary polygonal environments. In *ACM SIGGRAPH 1997 Conference Proceedings* (1997).

[Len99] LENGYEL J. E.: Compression of time-dependent geometry. In *Proceedings of Symposium on Interactive 3D Graphics* (1999), pp. 89–95.

[LT98] LINDSTROM P., TURK G.: Fast and memory efficient polygonal simplification. In *IEEE Visualization 1998 Conference Proceedings* (1998), pp. 279–286.

[MG03] MOHR A., GLEICHER M.: *Deformation sensitive decimation.* Tech. rep., University of Wisconsin, 2003.

[SBP00] SHAMIR A., BAJAJ C., PASCUCCI V.: Multi-resolution dynamic meshes with arbitrary deformations. In *IEEE Visualization 2000 Conference Proceedings* (2000), pp. 423–430.

[SP01] SHAMIR A., PASCUCCI V.: Temporal and spatial level of details for dynamic meshes. In *Proceedings of ACM Symposium on Virtual Reality Software and Technology* (2001), pp. 77–84.

[SZL92] SCHROEDER W. J., ZARGE J. A., LORENSEN W. E.: Decimation of triangle meshes. *ACM Computer Graphics (SIGGRAPH 1992 Conference Proceedings) 26*, 2 (1992), 65–70.

[XV96] XIA J. C., VARSHNEY A.: Dynamic view-dependent simplification for polygonal models. In *IEEE Visualization 1996 Conference Proceedings* (1996), pp. 327–334.