Intersection-free mesh decimation for high resolution cloth models

Ursula Derichs*, Martin Misiak*†, Arnulph Fuhrmann*

* TH Köln
Computer Graphics Group
ursula.derichs@th-koeln.de
arnulph.fuhrmann@th-koeln.de

† Universität Würzburg
HCI Group
martin.misiak@th-koeln.de

Abstract: We present an approach to reduce high-resolution polygonal clothing meshes for Mixed Reality (VR/AR) scenarios. Due to hardware limitations, current mobile devices require 3D models with a strongly reduced triangle count to be displayed smoothly. A particular challenge for mesh reduction of clothing models is that these models usually consist of several fabric layers, which are spatially tightly together, touching in many places. Conventional decimation approaches and tools result in models with a lot of intersections between these layers. In this paper, we evaluate an incremental mesh decimation algorithm with the constraint of intersection-free decimations in each step on high resolution clothing models. A half-edge mesh data structure enables topological correct decimations and a clustered Bounding Volume Hierarchy (BVH) of the mesh triangles accelerates spatial neighborhood searches that are required for intersection tests. We demonstrate the performance of our approach on several real-world clothing models originating directly from a CAD application.

Keywords: Mesh decimation, intersection-free simplification, collision detection, augmented reality, virtual clothing, BVH

1 Introduction

Mesh decimation algorithms have been explored ever since polygon meshes are used in computer graphics [SZL92]. Today, a variety of tools and algorithms are available to reduce polygonal meshes. Despite the large body of work in this field, the simplification of clothing models received little attention.

Cloth simulation systems can handle hundred of thousands [TWL+18] to millions of triangles [SSIF09]. Clothing models used in the garment industry for virtual prototyping often consist of about half a million triangles. Moreover many garments have several layers of fabrics. To render fabrics with real cloth thickness, each layer consists of a front- and a back-side. The distance between them can be as low as 0.1 mm for thin fabrics and up to several millimeters for thicker ones. The two sides can end up penetrating each other during conventional reduction as they are in extreme close proximity. In addition, if the clothing is presented on an avatar, its surface can also penetrate the clothing layers (cf. Figure 1, left). Both type of intersections are visually undesirable and time consuming to resolve manually.

Veröffentlicht durch die Gesellschaft für Informatik e.V. 2020 in B. Weyers, C. Lürig, D. Zielasko (Hrsg.):
GI VR / AR Workshop 2020, 24. - 25, September 2020, Trier.

http://dx.doi.org/10.18420/vrar2020_32

Figure 1: Intersections (left), patterns (middle), boundary gaps (right)

A possible workaround would be to introduce a safety distance between all fabric layers. However, this does not completely avoid intersections and can lead to unrealistic looking clothing. The complete removal of intermediate layers is also not an option, as transparencies play a significant part in the perception of clothes.

Another characteristics which needs to be considered during a reduction is that clothes are made of pattern pieces, which are mapped to different mesh parts, that do not share vertices at their edges (cf. Figure 1, middle). A reduction must maintain the border line of all mesh parts, and must ensure that mesh parts stay together and do not drift apart and show gaps between the patterns (cf. Figure 1, right).

Combining all of these considerations, it is evident that cloth simplification is no trivial task. We tested many commercially available mesh reduction tools, however none provided satisfactory results. Therefore, we devised and implemented an approach for the simplification of complex cloth meshes. Our approach ensures no penetrations between the different fabric layers or the avatar take place, while preserving the borders of individual patterns and their topology.

2 Related Work

Mesh simplification is a large and well researched field. Existing algorithms can be classified into vertex clustering algorithms, incremental decimation algorithms, resampling algorithms and mesh approximation algorithms [BKP+10].

A very common and well evaluated method is the iterative edge collapse [Hop96]. Many variants have been proposed and studied. They mainly differ in the order the edges are picked for collapses. This order can be based on different geometric as well as perceptual error metrics. The most influential one is the Quadric Error Metric (QEM) [GH97], as it can be efficiently computed and naturally handles multiple attributes per vertex. Many decimation approaches use it as a foundation for more sophisticated error metrics based on perceptual features [LVJ05, TZCL16]. Bahirat et al. [BLMP18] presented a simplification algorithm for non-manifold meshes targeted at large reduction rates. The key observation for

maintaining an acceptable quality was to preserve boundaries located at surface curvatures.

To find an intersection between two moving triangle meshes, it is sufficient to test all edges against each other as well as to test all vertices against the faces. This can be solved by using algorithms proposed for continuous collision detection (CCD), see e.g. [HF07]. But the elementary tests involved in CCD have been proven to be prone to numerical issues [TTWM14] and solutions need considerably implementation effort. By concentrating on the special cases arising during mesh simplification, Gumhold et al. [GBK03] were the first to describe how edge collapses can be performed efficiently and numerically stable while retaining an intersection-free mesh.

While the use case of Gumhold et al. was also the intersection-free reduction of clothing models, our work differs in a few important aspects. The set of intersection tests presented by the authors can be significantly reduced, resulting in faster run times. In addition, we employ a clustered BVH to accelerate the spatial neighborhood searches to determine possible intersections. In contrast to the regular grid used by Gumhold et al., our data structure scales much better with mesh complexity, as we demonstrate by reducing clothing models acquired directly from a professional CAD software.

3 Mesh Decimation Implementation

The mesh decimation approach we are mainly following here is based on successive edge collapses with the main constraint, that an edge collapse is not performed if an intersection would occur [GBK03]. Mesh simplification algorithms based on edge collapse are flexible and widely used for reducing the resolution of very detailed meshes. QEM is used as an error metric due to its efficiency and accuracy for the selection of the edges to be collapsed [GH97]. The implementation of our basic decimation algorithm is based on the OpenMesh[1] decimation framework along with its half-edge data structure.

An edge collapse can be performed with different placement strategies for the position of the merged vertex. A simple, but fast and robust operation is known as halfedge collapse, which pulls one of the vertices (v_0) onto the other (v_1) (cf. Figure 4). When an edge collapses, all faces connected to v_0 will be moved to v_1 and change their shape. The halfedges between v_0 and v_1 and the faces connected to those halfedges will be removed from the mesh, and v_1 and the other connected faces to v_1 will not change.

Other placement strategies for the new vertex position have been proposed. Previous research work has shown, that a placement of the new vertex along the edge between v_0 and v_1 does not yield better results in quality [LN17]. However an optimal vertex placement as suggested in [GH97] could improve the quality of the simplified mesh. For a high-resolution model an additional optimization step for the vertex placement would have a considerable run time impact, and also the complexity for intersection tests is increasing considerably.

During decimation it is first checked for an edge if a collapse is topologically correct, then

[1] https://www.graphics.rwth-aachen.de/software/openmesh/

the quadric error for a collapse is calculated and the edge is inserted into a priority heap. There, the edge with the lowest error contribution is always on the top. One by one, the edges are picked from the heap and collapsed. Beside the QEM metric, other criteria can be taken into account, because an edge collapse could lead to an undesired visual artifact, despite of a low geometric error contribution. In the following we call these criteria constraints.

3.1 Constraints

3D models have different characteristics and need different criteria to be checked during the reduction to achieve a visually convincing result. For cloth models boundary preservation and intersection prevention are important constraints. It also turned out beneficial to prevent normal flipping and to assure a healthy aspect ratio for the triangles.

The order in which the constraints are checked is influencing the overall performance. If a fast check already prevents the collapse, slower and more complex checks like the intersection check do not need to run. Furthermore, it is an advantage to put aspect ratio tests before other tests, because this already eliminates collapses leading to degenerated faces, which could cause numeric problems in later checks. When an edge collapses, the faces connected to the deleted vertex will be moved to the new vertex and can intersect with other faces of the mesh, especially in a layered or folded mesh. As the intersection constraint is the main focus of this work, this constraint is described in more detail in chapter 4.

For the preservation of mesh boundaries [GH97] have introduced perpendicular penalty planes for the border edges, to artificially increase the error and so preserve the boundaries. For many models this works well, but for the cloth models this approach has proven to be insufficient to ensure that the borders are kept. A more strict boundary check had to be implemented, to prevent that edge collapses lead to a fringed boundary or reveal gaps between cloth patterns. It is obvious that collapses from a boundary vertex to an inner vertex must be prevented (cf. Figure 2 a)). If both vertices of the edge are on the boundary, then there are two cases. The edge connected to the contracting vertex has a different direction as the contracting edge (cf. Figure 2 b)), which would lead to a changed border shape. In the second case, the edge connected to the contracting vertex is collinear with the contracting edge (cf. Figure 2 c)), where the boundary shape does not change and the collapse is valid.

Another effect, that is to be prevented, is a face folding over after an edge collapse (cf. Figure 3). In that case the face's normal will point in another direction than before. This is called a normal flip. As this can lead to shading artefacts and also to self intersections within the same mesh layer, the collapse can not be allowed. The criterion takes into account the angular deviation between each face normal before and after the collapse. The collapse will pass the test, if the deviation is below a given threshold.

Faces also shall not degenerate during the decimation process. We have used the aspect ratio criterion, that requires that the edges of a triangle have similar length. For each collapse it is required that the aspect ratio for all affected faces is either better than before the collapse or better than a specified threshold.

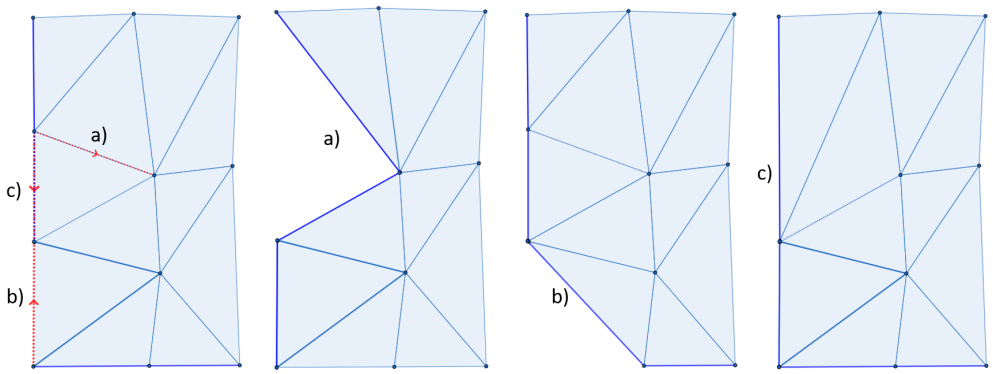

Figure 2: Illegal collapses a) and b), legal collapse c)

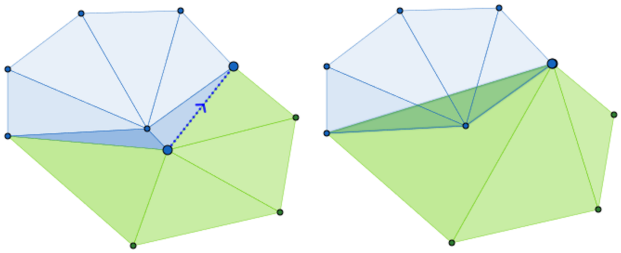

Figure 3: The dark blue face folds over when the edge collapses

4 Intersection Tests

4.1 Intersection Cases

The examination of intersection cases is based on the precondition that the mesh to be reduced is triangular, manifold and intersection-free from the start. The output of most cloth simulation systems is intersection-free. If this is not the case, the output can be untangled [BRB+19].

At each step of the decimation only intersection-free collapses are allowed, that means that we keep the mesh always intersection-free during the decimation. For this we follow the approach of Gumhold et al. [GBK03], but with a crucial distinction, as we use the halfedge collapse operator, which only moves one vertex at a time to the position of the vertex at the other end of the edge. This simple, but efficient operation reduces the number of intersection tests to be implemented considerably.

Let's assume, an edge shall be contracted from v_0 to v_1. The contraction operation can be described as a parameterized movement over time $v(t) = v_0 + t(v_1 - v_0)$ with $t \in [0, 1]$.

The vertex v_0 and all edges and faces connected to v_0 are the simplices that are moved over time and could potentially collide with the other non moving (stationary) simplices of the mesh. In Figure 4 the contraction is illustrated for $t = 0$ (case a)), $t = 0.5$ (case b)), $t = 1$ (case c)) and shows some *stationary simplices* of the mesh colored in blue, *moving simplices* in red, and *contracting simplices* in grey. The blue colored simplices, which are a subset of the stationary simplices, play a special role and are called the outer vertex ring of

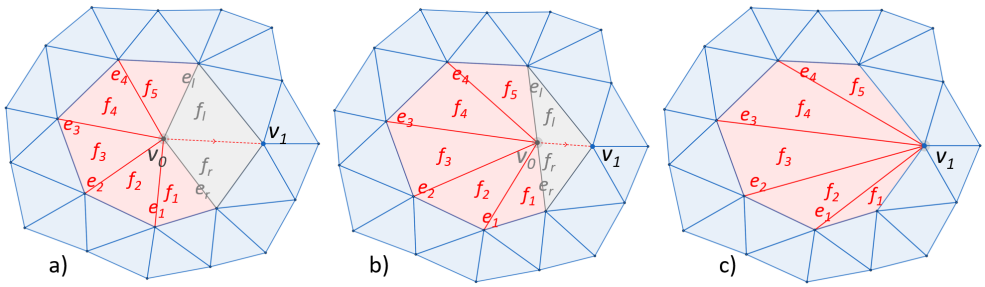

Figure 4: Edge collapse as an operation over time (from left to right)

v_0 in the following explanation.

A fold-over of faces connected to the moving vertex is excluded with a preceding normal flip test. Therefore we can safely move the red simplices without a collision with an edge or vertex from the outer vertex ring.

We can also exclude that contracting simplices do collide with other simplices. The Vertex v_0 will move along the edge between v_0 and v_1, and as this edge is intersection-free, the move is collision-free. The same applies for the edges e_l and e_r, they will move along the surface of the faces f_l and f_r during contraction. As the faces are intersection-free, the edge moves are collision-free.

So it only remains to be checked if the group of moving simplices collide with any of the other stationary simplices. Collision detection algorithms will be applied on the moving vertices. In Figure 4, the moving simplices are the edges $e_1, ..., e_4$ and faces $f_1, ..., f_5$.

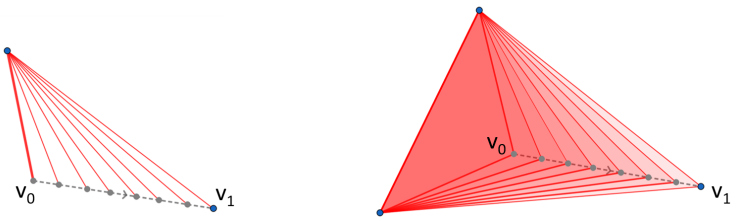

Figure 5: Time Sweeps for collision detection of contracting edges (left) and faces (right).

The movement of an edge over time corresponds to a triangle (cf. Figure 5), which needs to be collision tested against all stationary edges in the neighborhood. If there is no collision hit between the edge time sweep and another edge, then the face time sweep, which corresponds to a tetrahedron (cf. Figure 5) must be collision checked against all stationary vertices in the neighborhood. These two collision tests are sufficient to find all potential intersections. We query the neighbouring edges and vertices via a BVH with fast bounding box intersection tests. The two collision cases can be realized with an elementary geometric segment-triangle intersection test and a tetrahedron-point inclusion test.

We have added an ϵ value to slightly blow up the bounding volume of all faces to prevent numeric imprecisions and to not overlook any intersections. Degenerated time sweeps can

arise, which are not trivial to handle numerically, but fortunately these cases can be disregarded. The time sweep of an edge could theoretically be a line or a very needle-like triangle. In that case the edge is moved along the stationary edge itself and the collapsed edge, in this line no collision can arise. The other time sweep is a tetrahedron, which could be coplanar or even collinear. The collinear case can be excluded for the same reason as the collinear edge time sweep. In the coplanar case, the face would move along the intersection-free surface of the face itself and a neighbouring face, which is also collision free.

The edges directly connected to the contraction affected faces (coloured in red and grey in Figure 4) cannot collide with the moving faces. As edge and face time sweeps would touch these edges at their vertices, this kind of intersection must be excluded. A simple solution is to tag all the edges that shall be excluded from the collision check.

4.2 Bounding Volume Hierarchy

A clustered Bounding Volume Hierarchy (BVH) data structure is used for the spatial sorting of all mesh triangles to enable fast intersection detection of nearby simplices during edge collapses. The clustering of triangles has proven as an efficient strategy for collision detection within topological changing models [Gar09, HOEM15].

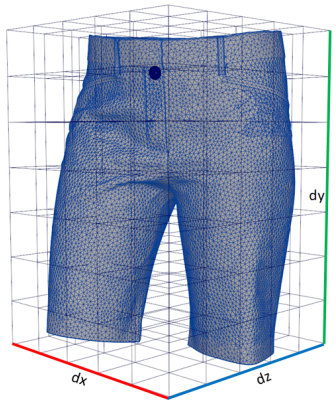

Figure 6: Uniform grid for BVH clusters

The BVH is constructed top-down as a binary tree, recursively splitting the faces along the longest axis of their bounding box. After each collapse the bounding volumes in the BVH must be adapted, because faces have been moved, changed their dimensions and some have been deleted. To accelerate the traversal and updates of the BVH, the triangles are clustered into spatial regions. With a regular grid, the mesh is partitioned into cuboids (cf. Figure 6) and all faces, which have their centers in the same grid will be in the same cluster. With the clustering, there is an additional layer in the BVH, consisting of a cluster tree. As for triangles, the clusters are recursively splitted into a binary tree according to their bounding volumes, which yields a tree with the clusters as leaves. Underneath each cluster there is the corresponding BVH subtree of all faces contained within the cluster. This is very efficient for the adaptation of the bounding volumes, as the affected faces are usually located within proximity in the same cluster. Therefore, the update is in most cases limited to only one cluster.

The number of faces in each cluster varies, and there are obviously many empty clusters. To save storage, the clusters are maintained in a hash table. The distribution of faces according to a regular grid turned out simple and efficient for BVH updates. The grid dimensions are derived from a predefined cluster size c. Later on some analysis is presented for the selection of a suitable value for c.

4.3 Intersection Reevaluation during decimation

After each collapse operation, all moved and connected edges need to be checked again for constraints and the new quadric error, because the topology has changed. Reevaluated edges are reinserted in the heap, if they are not constrained. When it comes to intersections, it is however not sufficient to check only the connected edges, because also other edges need to be reevaluated, as they could become a valid intersection-free collapse target in a changed neighboring geometry. In this work it turned out, that a reevaluation does not sufficiently increase the reduction rate to justify the additional runtime. Another approach was proposed by Gumhold et al. [GBK03], who put discarded edge collapse operations in a FiFo queue. With a timer the discarded operations are periodically reconsidered.

5 Performance

5.1 Test Models

Four models are used to demonstrate the decimation approach. The Stanford bunny and the XYZ RGB dragon are used as reference models and shall demonstrate the general performance of the decimation algorithm, even if those meshes are not layered. The dragon has an under-title "XYZ - RGB" as detail, that the decimation must keep.

The main focus is to demonstrate the intersection-free decimation capabilities on real cloth models. One test model are woman shorts, which consist of double layered front and backsides, another waistband layer with straps, sewn-in pockets with an extra thick seam layer and a round button. Our second cloth model is a female avatar wearing a fleece jacket and a pant, to demonstrate also the capability to reduce a garment combination. Both models present a very challenging case for mesh reduction.

5.2 Results

Experimental results for the runtime behaviour and the resulting mesh quality are shown in Table 1. The results are for a fixed reduction to 30% of the original mesh. Of course some of the meshes could be reduced more aggressively, but for cloth models a reduction beyond 30% currently leads to undesired visual deviations. Testing has been performed with the following threshold values: 45° maximum allowed normal change angle, cluster size 100, minimum aspect ratio 0.2, 8° max. deviation angle for collinearity on boundaries. The times have been measured on an Intel i7 with 2.9 GHz.

The Hausdorff distance was sampled over all vertices of the original mesh and the reduced model using Meshlab [CCC+08]. It was measured as RMS value relative to the bounding box of the model, so that the figures are better comparable between models of different dimensions. Of course the discarding of collapses, that lead to an intersection, and instead performing collapses with a higher QEM, increase the overall error slightly.

In a mixed reality scenario, where cloth models are presented in a virtual environment, the

Models	Shorts	Garments+Avatar	Bunny	XYZ Dragon
# faces	99,928	631,142	69,451	7,219,045
# vertices	60,662	676,548	34,834	3,609,600
# faces (30%)	29,978	189,341	20,834	2,165,713
without intersection constraint				
Red. time in sec.	0.035	0.034	0.046	0.037
Hausdorff distance	0.00015	0.000045	0.00016	0.000021
with intersection constraint				
Red. time in sec.	0.14	0.14	0.09	0.22
Hausdorff distance	0.00027	0.000051	0.00016	0.000021
# prev. intersections	18,852	40,728	0	314

Table 1: Experimental results for a reduction to 30 %. Red. times are per 1000 edge collapses.

visual appearance and the perceived similarity between the original and the reduced model is however a more important aspect as the geometrical error. We have examined the results on a desktop system and in a standalone VR headset (Oculus Quest). The original and reduced models look similar, and especially for the garment combination it takes intensive inspection to find the deviations. See Figure 7 and 8 for a visual comparison.

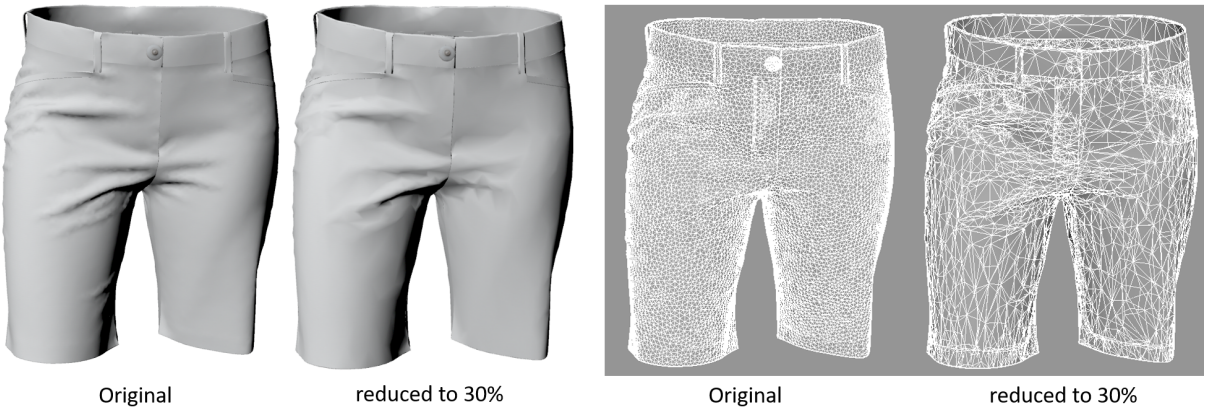

Figure 7: Original and reduced shorts model

The XYZ RGB dragon model could be reduced without destroying the lettering and within reasonable time, although it contains over 7 million triangles. A few intersections were found during the reduction, these could be numeric imprecisions or real self-intersections, as the surface is vaulted. For the bunny no intersection was found. The selected cluster size has a major impact on the runtime. Hence, we measured the runtime for different cluster sizes (see Figure 9) and found 100 to be optimal.

5.3 Discussion and Limitations

We aimed at presenting virtual clothes on standalone VR/AR Head Mounted Displays (HMDs). As a consequence of the limited graphic performance of such HMDs, the frame

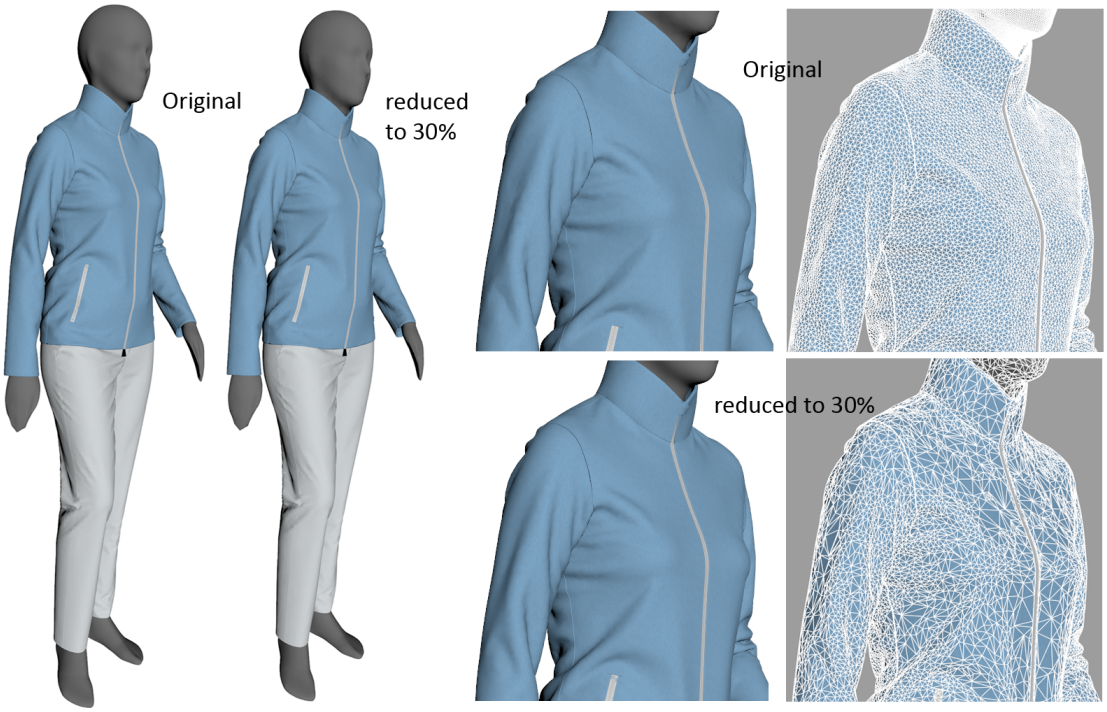

Figure 8: Original and reduced garment combination on female avatar

rate drops significantly for high resolution meshes. On an Oculus Quest the frame rate drops to 30 FPS for the avatar model wearing a garment combination. After reducing the mesh to 20 % the frame rate reaches an acceptable level of 50 to 72 FPS, depending on how close the viewer approaches the model, and the visual quality impact of the reduced mesh is hardly noticeably. Hence, we believe that future mobile VR applications will benefit strongly from our intersection-free mesh decimation algorithm.

The textures were not explicitly considered as criterion in the reduction, but the resulting reductions had minimal texture deviations. The cloth models have straight seams and patches as textures, but these were barely distorted. For high reduction rates initially tiny gaps between the mesh parts will become noticeable. The boundary constraint alone is not sufficient to prevent diverging boundaries completely.

The clustered BVH has proven to be a fast accelerator for neighbor simplex queries and has outperformed a uniform grid, which has been experimentally implemented. However, the BVH is currently designed for top-down traversals, which is sub optimal for bounding volume adaptations, which take place at the leaves and maybe a few iterations upwards the tree. A BVH supporting bottom-up traversal could decrease the update time.

6 Conclusion and Future work

The step wise decimation method could successfully be carried out for high resolution meshes with the addition of intersection tests and special handling of boundaries. A powerful and fast spatial data structure is crucial for the reduction of high resolution meshes. The visual

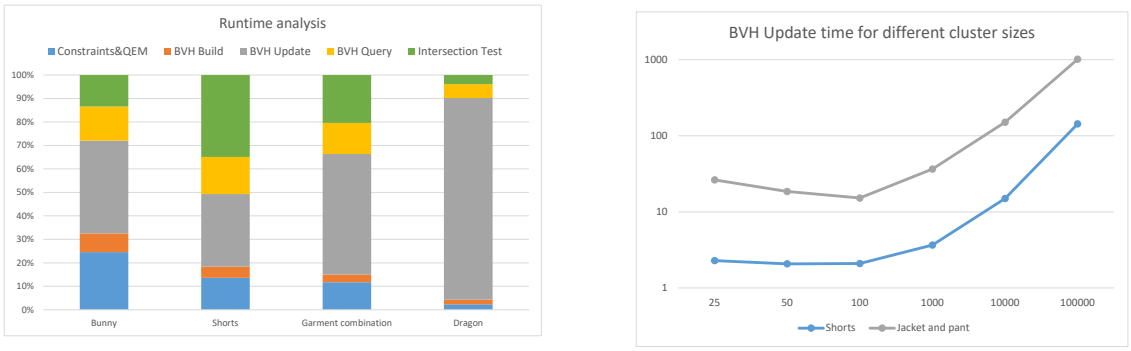

Figure 9: Load distribution with BVH details and cluster size variations

appearance of the reduced meshes resemble the original quite well, and can be loaded into standalone HMDs with acceptable frame rates.

At the moment only the QEM decides which edge is picked, other criteria are only binary (collapse is allowed/not allowed). The basic decimation algorithm could weight different criteria (QEM, aspect ratio, boundary effects) and visual or perceptual metrics, also considering texture effects, could be added.

The maximum reduction for cloth models is currently limited to 20-30 % of the original mesh, depending on the model's complexity. The reduction rate could be slightly better, if intersections would be reevaluated as suggested in chapter 4.3, or even avoided by choosing a vertex placement that does not lead to an intersection, as suggested by [GBK03]. An optimized vertex placement could even preserve the original shape more accurately.

Higher reduction can also be reached, if inner mesh parts are more aggressively reduced than outer visible mesh parts, provided that the outer parts are not transparent. An intelligent mesh inspection could heuristically analyze which layers are visible and which layers are hidden, based on normal evaluation.

To safely prevent the diverging boundary effect, vertices within an ϵ distance could be merged (welded), but still be marked as boundary vertices before the decimation. Currently, all calculations are done on the CPU. A GPU implementation of the algorithm would be a possible optimization candidate.

7 Acknowledgements

This work was funded by the German Federal Ministry of Education and Research (BMBF) under grant number 02K16C232 as part of the project Retail 4.0. We would like to thank our project partners Assyst GmbH and Leineweber GmbH & Co. KG for providing the 3D cloth models. The bunny and dragon models are from the Stanford 3D Scanning Repository. We would also like to thank Gabriel Schmitz for the development of the clustered BVH that was used and further optimised in this paper.

References

[BKP+10] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. *Polygon Mesh Processing*, chapter 7. CRC press, 2010.

[BLMP18] Kanchan Bahirat, Chengyuan Lai, Ryan P Mcmahan, and Balakrishnan Prabhakaran. Designing and evaluating a mesh simplification algorithm for virtual reality. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 14(3s):1–26, 2018.

[BRB+19] Thomas Buffet, Damien Rohmer, Loïc Barthe, Laurence Boissieux, and Marie-Paule Cani. Implicit untangling: A robust solution for modeling layered clothing. *ACM Trans. Graph.*, 38(4), July 2019.

[CCC+08] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In Vittorio Scarano, Rosario De Chiara, and Ugo Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008.

[Gar09] Kirill Garanzha. The use of precomputed triangle clusters for accelerated ray tracing in dynamic scenes. *Comput. Graph. Forum*, 28:1199–1206, 06 2009.

[GBK03] Stefan Gumhold, Pavel Borodin, and Richard Klein. Intersection free simplification. *International Journal of Shape Modeling - IJSM*, 9, 12 2003.

[GH97] Michael Garland and Paul Heckbert. Surface simplification using quadric error metrics. In *Surface Simplification Using Quadric Error Metrics*, volume 1997, 07 1997.

[HF07] Marco Hutter and Arnulph Fuhrmann. Optimized continuous collision detection for deformable triangle meshes. *Journal of WSCG*, 15(1-3):25–32, 2007.

[HOEM15] Liang He, Ricardo Ortiz, Andinet Enquobahrie, and Dinesh Manocha. Interactive continuous collision detection for topology changing models using dynamic clustering. *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 47–54, 02 2015.

[Hop96] Hughes Hoppe. Progressive meshes. *Computer Graphics (Proc. SIGGRAPH '96)*, 30:99–108, 01 1996.

[LN17] Guillaume Loubet and Fabrice Neyret. Hybrid mesh-volume LoDs for all-scale pre-filtering of complex 3D assets. *Computer Graphics Forum*, 36(2):431–442, 2017.

[LVJ05] Chang Ha Lee, Amitabh Varshney, and David W Jacobs. Mesh saliency. In *ACM SIGGRAPH 2005 Papers*, pages 659–666. 2005.

[SSIF09] A. Selle, J. Su, G. Irving, and R. Fedkiw. Robust high-resolution cloth using parallelism, history-based collisions, and accurate friction. *IEEE Transactions on Visualization and Computer Graphics*, 15(2):339–350, 2009.

[SZL92] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '92, page 65–70, New York, NY, USA, 1992. ACM.

[TTWM14] Min Tang, Ruofeng Tong, Zhendong Wang, and Dinesh Manocha. Fast and exact continuous collision detection with bernstein sign classification. *ACM Trans. Graph.*, 33(6), 2014.

[TWL+18] Min Tang, Tongtong Wang, Zhongyuan Liu, Ruofeng Tong, and Dinesh Manocha. I-Cloth: Incremental collision handling for GPU-based interactive cloth simulation. *ACM Transaction on Graphics (Proceedings of SIGGRAPH Asia)*, 37(6):204:1–10, 2018.

[TZCL16] Pingping Tao, Lina Zhang, Junjie Cao, and Xiuping Liu. Mesh saliency detection based on entropy. In *6th International Conference on Digital Home (ICDH)*, pages 288–295. IEEE, 2016.