NORTHWESTERN
UNIVERSITY

Computer Science Department

Technical Report
NWU-CS-02-09
March 29, 2002

Completely Adaptive Simplification of Massive Meshes

Prasun Choudhury **Benjamin Watson**

Abstract

The growing availability of massive models and the inability of most existing visualization tools to work with them requires efficient new methods for massive mesh simplification. In this paper, we present a completely adaptive, virtual memory based simplification algorithm for large polygonal datasets. The algorithm is an enhancement of `RSimp` [2], enabling out of core simplification without reducing the output quality of the original algorithm.

The primary improvement in our new algorithm, `VMRSimp`, builds and preserves memory locality throughout the simplification process. This is crucial for successful simplification of massive meshes. An additional enhancement based on a depthfirst simplification approach improves running time further by increasing reuse of the resident working set, at the cost of a minor reduction in output quality. `VMRSimp` performs completely adaptive simplification of massive meshes at reasonable rates, reducing for example 18 million to 100 thousand vertices in 51 minutes. This permits simplification to output sizes in the millions without thrashing, improves accuracy for all smaller output sizes, and enables sensitivity in simplification to mesh boundaries and topological changes.

CR Categories and Subject Descriptors: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling – Geometric algorithms.

Additional Keywords: model simplification, massive models, quadric error, memory locality, out-of-core simplification

Completely Adaptive Simplification of Massive Meshes

Prasun Choudhury*
Northwestern University

Benjamin Watson†
Northwestern University

ABSTRACT

The growing availability of massive models and the inability of most existing visualization tools to work with them requires efficient new methods for massive mesh simplification. In this paper, we present a completely adaptive, virtual memory based simplification algorithm for large polygonal datasets. The algorithm is an enhancement of `RSimp` [2], enabling out of core simplification without reducing the output quality of the original algorithm.

The primary improvement in our new algorithm, `VMRSimp`, builds and preserves memory locality throughout the simplification process. This is crucial for successful simplification of massive meshes. An additional enhancement based on a depth-first simplification approach improves running time further by increasing reuse of the resident working set, at the cost of a minor reduction in output quality. `VMRSimp` performs completely adaptive simplification of massive meshes at reasonable rates, reducing for example 18 million to 100 thousand vertices in 51 minutes. This permits simplification to output sizes in the millions without thrashing, improves accuracy for all smaller output sizes, and enables sensitivity in simplification to mesh boundaries and topological changes.

CR Categories and Subject Descriptors: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling – Geometric algorithms.

Additional Keywords: model simplification, massive models, quadric error, memory locality, out-of-core simplification

1 INTRODUCTION

Recent years have seen a rapid increase in the size of polygonal datasets [1,8]. Several new technologies are contributing to this effect, such as the development of high-resolution 3D scanners. The Digital Michelangelo Project [8] at Stanford University has produced finely detailed meshes consisting of over 300 million triangles.

The enormity of these models poses a real challenge. Certainly such massive models will not be rendered in real time in the near future. Model simplification would seem to be an obvious solution, but ironically enough, most simplification algorithms are not designed to handle datasets at these sizes. Typically these algorithms require random access to mesh data, an approach completely impractical in out of core settings. There are a few algorithms designed to work out of core [9,10,15]. While fast, all these algorithms make use of a non-adaptive simplification phase that harms output quality.

In order to work efficiently with massive datasets, algorithms must minimize disk access. One way of achieving this goal is to ensure that all simplification occurs in core memory. The algorithms described by Lindstrom [9], El-Sana and Chiang [4], and Shaffer and Garland [15] take this approach, which unfortunately limits output accuracy by requiring that the output model fit in core memory. In [10], Lindstrom and Silva describe a memory-independent algorithm that substitutes disk for core memory. However, their simplifications must be non-adaptive to avoid thrashing.

*Dept. Mechanical Engineering, 2145 Sheridan Ave., Evanston, IL 60208 USA. p-choudhury@northwestern.edu
†Dept. Computer Science, 1890 Maple Ave., Evanston, IL 60201 USA. watson@northwestern.edu

2 CONTRIBUTIONS

Our approach, an enhancement of `RSimp` [2], relies on virtual memory, building and preserving memory locality to minimize page faults. By using this approach, `VMRSimp` is able to perform completely adaptive simplification of massive meshes, from the first operation on input until the final operation producing output. Despite this, simplifications are performed in reasonable time. The benefits of our approach include:

Improved accuracy: simplified models produced by `VMRSimp` are more accurate than models output by other algorithms for massive meshes.

Topological and boundary sensitivity: because it uses virtual memory, `VMRSimp` does not require a dereferenced "polygon soup" format, and can retain input topological information. With this information, `VMRSimp` is able to avoid joining topologically disjoint model components, and to preserve the shape of mesh boundaries.

Large, adaptively simplified output models: models output by the algorithms described in [4,9,15] must fit in core memory, limiting output size. `VMRSimp` does not have this limitation. Lindstrom and Silva's algorithm [10] can output arbitrary sizes, but these are non-adaptive and less accurate than `VMRSimp`'s.

In the remainder of this paper, we review related work (section 3), describe the simplification algorithm (section 4), discuss the results with examples (section 5) and finally conclude (section 6).

3 RELATED RESEARCH

Polygonal simplification has been an area of active research for close to a decade. A complete review of the field is beyond the scope of this paper – we will restrict our review to only those algorithms most relevant in our out of core context. (Good reviews of the entire field of model simplification are available at [6] and [11]).

Simplification of massive models requires extreme efficiency. For this reason, two algorithms designed for models that fit in core memory have been very influential. The non-adaptive algorithm introduced by Rossignac and Borrel [13] runs in linear time on input and is extremely fast. When performing adaptive simplification, the quadric metric introduced by Ronfard and Rossignac [12] and refined by Garland and Heckbert [5] has proven to be an extremely compact and efficient method for summarizing surface characteristics.

When models do not fit in core memory, the random memory access most simplification algorithms require results in severe thrashing, making them nearly impossible to use. To address this problem, Hoppe [7] proposed segmenting massive meshes, and

simplifying each segment independently in core, with boundary edges preserved so that the meshes can be rejoined. However, segmented simplification is unlikely to approach globally optimal accuracy. Lindstrom [9] developed an out of core algorithm based on Rossignac and Borrel's linear non-adaptive technique [13]. If the input model uses an indexed vertex format, the algorithm begins by converting input to a dereferenced "polygon soup" format that increases input size and discards topological information in order to gain locality of reference. The resulting algorithm is extremely fast and can simplify input models with more than 300 million triangles. However, non-adaptive simplification also results in poor accuracy, especially at smaller output sizes. The algorithm also assumes that one has sufficient core memory to contain the simplified output model. In [10], Lindstrom and Silva describe an approach that improves [9] to overcome this output size limitation.

Shaffer and Garland [15] use a two-pass algorithm and achieve better output quality at the expense of more computation time. The first, non-adaptive pass applies a version of Lindstrom's algorithm [9]. The output from this pass becomes input to the final adaptive pass, which uses a variation of `RSimp`'s splitting approach, described by Brodsky and Watson in [2]. Vertices are clustered using a BSP tree based partitioning scheme. Although this algorithm improves accuracy at smaller output sizes, quality still suffers from the first non-adaptive pass. Salamon et al. [14] characterized memory use in `RSimp` during simplification of large models, and implemented some improvements.

4 VMRSimp

In order to describe our new virtual memory version of `RSimp` (`VMRSimp`), we will provide a brief overview of the `RSimp` algorithm as well as the data structures used in the algorithm. Further details of the `RSimp` algorithm can be found in [2].

4.1 RSimp data structures and algorithm

The principal data structure in this algorithm is the *cluster*, which represents a surface patch on the input model and a single vertex in the output model. We label the variation of the face normals in the surface patch `nv`. Clusters are organized into a priority queue sorted by `nv` for greedy, best-first processing. The other important data structures are the global face and global vertex lists (`gfl` and `gvl` respectively). The `RSimp` algorithm has three stages:

Initialization: in this stage the global face (`gfl`) and vertex (`gvl`) lists are created and filled. The simplified model is initialized to represent a single cluster, which is immediately split into eight smaller clusters.

Simplification: choose the cluster with the largest `nv` from the priority queue and split the cluster based on the pattern of normal variation into at most eight subclusters. Compute `nv` for the new subclusters and place them into the priority queue. Iterate until the required output size is reached. While splitting ensure that no topologically disjoint components of the model are in the same cluster (creating new clusters if necessary).

Post Processing: for each of the remaining clusters, compute a representative output vertex using quadric error optimization [5]. Retain in the output model only those input faces with vertices in three different output clusters.

4.2 Modifications to RSimp

`VMRSimp`'s primary enhancement of the `RSimp` algorithm builds and preserves memory locality during simplification. In `Rsimp`, the vertices and faces contained in each cluster were represented by two lists of indices into the `gvl` and `gfl`. `VMRSimp` eliminates the per cluster lists of indices and replaces them with four array indices. These indices represent the ranges owned by the cluster in the `gvl` and `gfl`. In order to enable this representation, `VMRSimp` must sort the `gvl` and `gfl` to correspond to clusters. During splitting, the vertices and faces in a cluster's range of the `gvl` and `gfl` are sorted into new subranges corresponding to the new subclusters. Since the number of new subclusters is constant, this sort can be performed in time linearly proportional to the size of the original cluster's ranges in the `gvl` and `gfl`. Any splitting required to avoid topological joining is performed in a similar fashion by sorting disjoint components in the new subclusters into separate subranges.

This enhancement, while simple, proves to be extremely powerful. The primary benefit results from the locality of reference built by this new sorting scheme: all the data necessary for working on a cluster is brought into a continuous memory range. This locality is improved as the algorithm progresses and clusters become smaller. In addition, by replacing the per-cluster lists with just four indices, the memory used per cluster is reduced significantly. In `RSimp`, the memory required to represent clusters in the priority queue was proportional to input size. In `VMRSimp`, cluster representation is proportional to output size.

4.2.1 Depth-first heuristic

`RSimp` simplifies until a user chosen output size is reached, splitting clusters across the model in a best-first fashion (we call this size-based control). To exploit locality further through reuse of the current working set, in `VMRSimp` we have introduced an optional depth-first simplification heuristic. By splitting a cluster more than once while it is in core memory, this heuristic introduces a user-controllable tradeoff between output quality and simplification time.

The heuristic calculates V, the number of vertices that should be produced from a new cluster, based on currently available information. When $V > 1$, the new cluster is immediately split again, and resulting subclusters receive the same recursive treatment. Clusters in which $V \leq 1$ are placed back into the priority queue. Only when $V \leq 1$ in all locally and recursively produced clusters does regular greedy, best-first processing resume. The heuristic itself takes the form

$$V = \min(C,(1-M/N)B + (M/N)D)$$

where C is the number of input vertices in the cluster, M is the targeted number of vertices in the output model, and N the number of vertices in the input model. B is a quantity representing the number of vertices that should produced from the cluster using only best-first criteria. D is a similar quantity representing the number of vertices that should be produced according to strict depth-first criteria. Thus V represents a compromise between best and depth-first guesses at the number of vertices that should be produced from a cluster. As targeted output size approaches input size, the depth-first guess dominates. The minimum function ensures that V will never be larger than the number of input vertices actually in the cluster.

Because it is not best-first, D ignores the differences in normal variation between clusters, and is equal to $(M/N)C$. B is much more complex:

$$B = K(M-m)D$$

where m is the number of clusters existing at the current stage of simplification, K is a value in range [0,1] indicating confidence in B's best first criteria (maximum confidence is 1), and D is the normal variation in the current cluster normalized by the summed normal variation of all clusters: $\mathtt{nv}/(\Sigma_{1 \text{ to } m}\, \mathtt{nv}_i)$.

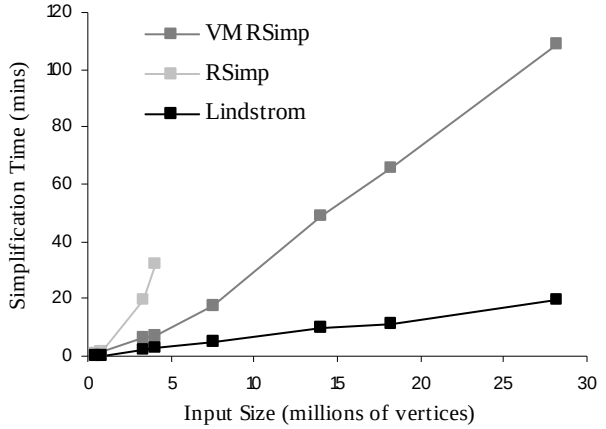

Figure 1: Simplification times for `VMRSimp` (size-based control), Lindstroms's algorithm [9] and `Rsimp` [2]. Differently sized models were simplified to 100,000 vertices.

4.2.2 Accuracy-based control

In addition to size-based control, `VMRSimp` allows accuracy-based output control. In this case, clusters are split until user-specified accuracy is reached. Accuracy is calculated using the diagonal of the bounding box containing the input vertices in the cluster. During post processing, if the output vertex obtained through minimization of the cluster's quadric does not lie in the bounding box, the mean of the input vertices is output instead. Accuracy-based control not only allows explicit control of simplification accuracy, it also makes the decision to split a cluster completely local, allowing optimal memory reuse through depth-first traversal of the simplification tree. The cluster priority queue is replaced by an output cluster list, storing clusters represented with the required accuracy; and an inaccurate cluster stack, storing clusters that require more refinement. While placing new subclusters into the priority queue takes log *m* time, placing them into the stack takes constant time.

4.2.3 Mesh boundary sensitivity

To make simplification sensitive to mesh boundaries, `VMRSimp` uses a variation of the method described in [5]. Every edge on a mesh boundary is paired with a new boundary plane perpendicular to the face containing the edge. These planes are squared and added to the quadric used to locate output vertices. Boundary planes can also influence splitting. In `RSimp` splitting is accomplished by finding the eigenvectors and eigenvalues of the quadric matrix defined by faces in the cluster. Two of the eigenvectors indicate the directions of maximum and minimum normal variation. These directions are used to orient the splitting planes. `VMRSimp` defines an additional boundary quadric matrix using only the boundary planes. The boundary quadric only improves splitting when change in boundary plane orientation is unrelated to change in face plane orientation. This occurs when the cluster's surface patch is roughly planar near the boundary, yet despite this the boundary is still curved. In this case change in boundary plane orientation is primarily one dimensional, and so during splitting, `VMRSimp` adds the boundary matrix to the face matrix only when the eigenvalue corresponding to maximum boundary plane orientation change is significantly greater than the eigenvalue corresponding to minimum boundary plane orientation change.

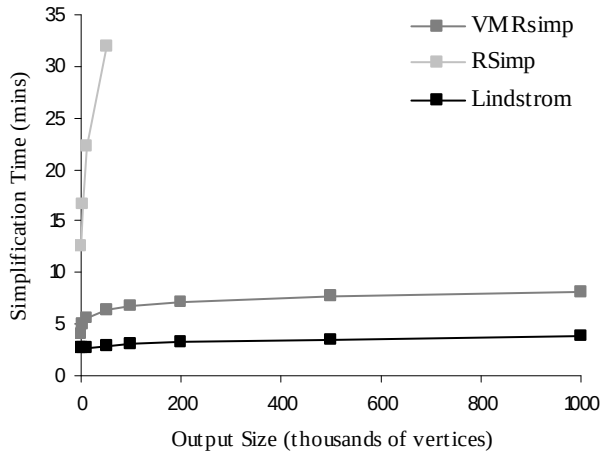

Figure 2: Performance time of `VMRSimp` (size-based control) for different output sizes of the David (4.1 million vertices) model as compared to Lindstrom's algorithm [9] and `RSimp`

5 RESULTS

Simplification algorithms are usually judged by two criteria: simplification quality and simplification time. For out of core simplification we add memory efficiency to this list: a good out of core algorithm should have as small a memory footprint as possible, and exploit locality in its use of that memory.

In this section, we present simplification times for differently sized input models simplified to various output sizes. We then examine the quality of these output models, as well as memory efficiency during the entire simplification process. Next we study the tradeoff between simplification time and quality enabled by our depth-first heuristic. Finally we study the performance benefits of error-based control of `VMRSimp`.

5.1 Simplification time

We used eight different models in our comparisons. All models were simplified on a 1GHz Dell PowerEdge 2400 PC with 1GB RAM and gigabytes of swap space, running RH Linux 7.1.

Figure 1 shows the simplification times of `VMRSimp` using size-based control compared to `RSimp` [2] and Lindstrom's algorithm [9] when simplifying various input models to 100,000 output vertices. Lindstrom's non-adaptive algorithm [9] is linear in input and six to seven times faster than `VMRSimp`. (Note that our implementation of Lindstrom's algorithm is likely not the most superbly optimized of codes). Although they theoretically have the same complexity (O(N log M), where N is input size and M is output size), `VMRSimp`'s efficient memory use makes it much faster than `RSimp`. In fact `RSimp` is unable to simplify models larger than 4 million vertices without severe thrashing. Shaffer and Garland [15] reported running times 3 to 6 times faster than `VMRSimp` on a slightly lower performance platform. Recall that the first pass of Shaffer and Garland's algorithm is also linear and non-adaptive.

Figure 2 shows how size-based simplification is affected by the size of the output model. Here the algorithms were simplifying the David model, which contains roughly 4.1 million vertices. `Rsimp`'s simplification time increases steeply as output model size increases. For both `VMRSimp` and Lindstrom's algorithm, simplification time increases only gradually with larger output sizes. The similarity between the `VMRSimp` and Lindstrom curves is striking, considering the differing complexities of these algorithms. Apparently the constantly improving locality of

Model	Input Vertices	Output Vertices	Page Faults (VMRSimp)	Page Faults (Rsimp)
Blade	0.9M	100K	36,476	95,186
St. Matthew Face	3.4M	10K	112,034	249,142
St. Matthew Face	3.4M	100K	120,947	368,378
David (small)	4.1M	10K	138,061	1,345,855
David (small)	4.1M	100K	147,563	3,310,098
Lucy	14.0M	10K	4,840,122	N/A
Lucy	14.0M	100K	5,120,327	N/A
David (big)	28.2M	10K	10,831,349	N/A

Table 1: Number of page faults for VMRSimp (size-based control) and Rsimp for different input and output sizes.

Model	Input Vertices	Output Vertices	Memory allocated at initialization	Swap space used
David (small)	4.1M	1M	368.6 MB	114 MB
Lucy	14.0M	10K	1267 MB	613 MB
Lucy	14.0M	500K	1267 MB	920 MB
David (big)	28.2M	10K	2416 MB	1960 MB

Table 2: Virtual memory used by VMRSimp during size-based simplification of different models.

Output Size (tris)	Lindstrom [9]		Adaptive simplification [15]		VMRSimp	
	mean	*max*	*mean*	*max*	*mean*	*max*
1K	0.4549	26.10	0.4821	25.91	0.345	14.89
10K	0.0986	24.35	0.0946	24.43	0.0598	12.8
100K	0.0266	24.47	0.0164	24.17	0.0119	10.45

Table 3: Quality of output models of different sizes for the Lucy model using VMRSimp with size-based control. Lindstrom's algorithm and adaptive simplification. Mean and maximum error are expressed as percentages of the model bounding box.

VMRSimp compensates for its disadvantage in complexity, at least at these output sizes. Supporting evidence for this explanation comes from RSimp's performance, which differs from VMRSimp primarily in its inefficient use of memory.

5.2 Paging and memory usage

We used the unix function getrusage to compare page faulting during the entire simplification process in RSimp and VMRSimp using size-based control. As foreshadowed by the simplification time results, page faulting is dramatically lower in VMRSimp. Table 1 summarizes these results. At larger input sizes, faulting is reduced by over an order of magnitude in VMRSimp. RSimp is unable to simplify models over 4.1 million vertices without severe thrashing.

We measured virtual memory footprints of VMRSimp using the unix command vmstat. Table 2 presents these results. Virtual memory usage, like simplification time, scales linearly with input model size. In our current implementation, memory is used primarily to store the input model's face list (15 bytes per record) and vertex list (56 bytes per record). Since simplification time is dominated by use of virtual memory, we reduce the memory footprint with on the fly computation of the area and normal vector of faces. As output sizes grow beyond those presented here, representing the output model requires more virtual memory. However, since each output vertex (or cluster) requires only 25 bytes of memory, representation of the input model will always the dominant memory cost.

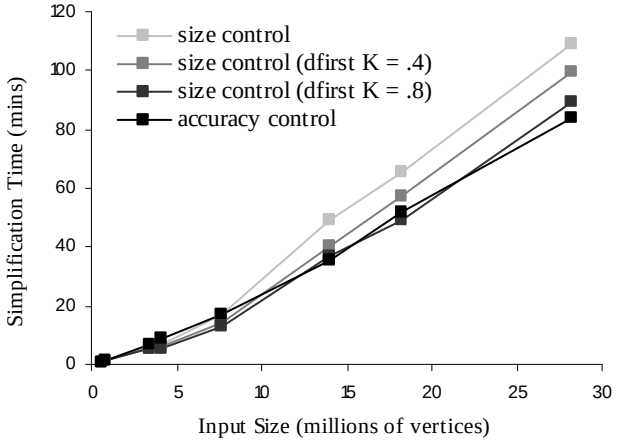

Figure 3: Simplification times for VMRSimp using accuracy-based control, size-based control with no depth first heuristic, and size-based control with the heuristic at two confidence levels. Different input models were simplified to 100,000 vertices.

Output Size (tris)	Best-First (no heuristic)		Depth-First K=0.4		Depth-First K=0.8	
	mean	*max*	*mean*	*max*	*mean*	*max*
10K	0.0598	12.8	0.0786	12.95	0.0897	14.41
100K	0.0119	10.45	0.0146	10.47	0.0162	14.31

Table 4: Quality of the Lucy model as simplified by size-based VMRSimp without the depth-first heuristic, and with two levels of confidence in the heuristic. Mean and maximum error are expressed as percentages of the model bounding box.

5.3 Simplification quality

Simplification quality can be measured both visually and quantitatively. For quantitative measurements, Cignoni et al. have developed the Metro [3] tool. Unfortunately Metro does not function when model sizes exceed roughly 1 million faces. We sidestep this problem by comparing smaller output models to models simplified to roughly 1 million faces using VMRSimp. Table 3 compares the output accuracy of VMRSimp to the Lindstrom [9] and the Shaffer and Garland [15] algorithms. Here Metro's standard is the Lucy model, simplified to roughly 1.2 million faces using VMRSimp. In general, mean error as a percentage of the bounding box diagonal is reduced by 25-40% when using VMRSimp rather than Shaffer and Garland's adaptive simplification. The improvement in maximum error is even more pronounced, with VMRSimp's output containing roughly 2 times less error than adaptive simplification. With larger models, maximum error in VMRSimp can be up to six times less than error with adaptive simplification.

Admittedly, these quality measures could be biased, because the standard model was produced using VMRSimp. However, visual quality checks seem to correspond to these quantitative measures. Figure 5 shows the Lucy model simplified to three output sizes by adaptive simplification, Lindstrom's algorithm and VMRSimp. The models simplified by VMRSimp look much smoother, particularly around the wings and the folding dress. This is probably due to VMRSimp's ability to vary face shape according to the local pattern of curvature. Adaptive simplification also introduces some extraneous joins between the wing and the body of the Lucy model. Similar results can be seen in the simplified version of the David model in Figure 6. VMRSimp is able to eliminate these joins with its check for topologically disjoint components. In Figure 4, the two simplified St. Matthew

Face models show the value of VMRSimp's sensitivity to mesh boundaries.

5.4 Depth-First Heuristic Performance

The depth-first heuristic improves simplification time by 10-20%, with higher confidence coefficients (*K*) improving times more (Figure 3). This is particularly true at large input sizes, when memory required per cluster range in the gvl and gfl grows, increasing the benefit from memory reuse. Table 4 indicates the reduction in output quality that comes with these improvements in simplification time. (Once again, we compare smaller simplified results to a simplified version of Lucy at 1.2 million faces). Maximum error remains well below Lindstrom's algorithm and adaptive simplification; while mean error rises as the heuristic is used more aggressively, eventually matching the mean error levels produced by adaptive simplification.

5.5 Accuracy-Based Simplification

Accuracy-based output control allows simplification times nearly 25% lower than purely best-first size-based control, and equal to or better than the most aggressive depth-first heuristic control (see again Figure 3). Once more, this is particularly true at large input sizes. Unlike heuristic control, these improvements in simplification time do not come at the price of output quality. However, accuracy-based control cannot offer users precise selection of output size.

6 CONCLUSION AND FUTURE WORK

We have presented VMRSimp, an algorithm for fully adaptive simplification of massive meshes. The algorithm makes extensive and efficient use of virtual memory, allowing simplification of these massive meshes at reasonable speed. Because the simplification is fully adaptive, the quality of VMRSimp's output is better than quality in output from other simplification algorithms for massive meshes. Unlike these other algorithms, VMRSimp retains the topological information present in its input, allowing it to avoid merging topologically disjoint model components and to preserve mesh boundaries. Models output by VMRSimp can be quite large, and need not fit in core memory.

We have examined the performance of VMRSimp with models that near the limits of virtual address space on 32 bit machines. In particular, the larger David model approached this limit, and took nearly two hours to simplify using size-based control, one and a half hours with accuracy-based control. We plan to examine larger models soon on a 64 bit machine with significant swap space, and believe based on current trends that we will be able to simplify models containing hundreds of millions of faces.

As Figure 2 makes clear, most of the computation required to simplify massive models in VMRSimp is performed at the early stages of simplification, which put the model in coarse spatial order. If input models already had some rough spatial organization, simplification times might drop dramatically, as reported in [14]. We have experimented with creating this rough order in linear time using a hashing scheme similar to Lindstrom's algorithm [9], but have found that the resulting improvements in simplification time were approximately the same as the expended hash times. However, it may be that the spatial organization achieved in our hashing attempts was too coarse. We plan to investigate the possible benefits of using finer resolutions in our hashing scheme.

7 ACKNOWLEDGMENTS

Our thanks to Eric Shaffer and Michael Garland for providing simplification results with their algorithm and Andrew Raij for his work on accuracy-based simplification. The Lucy, Dragon and Happy-Buddha models were obtained courtesy of the Stanford Scanning Repository, while the David and the St-Matthew models were acquired with the permission of the Stanford Digital Michelangelo Project. The Blade model is distributed with Kitware's VTK package.

8 REFERENCES

[1] F. Bernardini, I. Martin, J. Mittleman, H. Rushmeier and G. Taubin. Building a digital model of Michelangelo's Florentine Pieta. *IEEE Computer Graphics and Applications*, vol. 22(1): 59-67, Jan. - Feb. 2002.

[2] D. Brodsky and B. Watson. Model simplification through refinement. In *Proceedings of Graphics Interface* 2000, pages 221-228, May 2000.

[3] P. Cignoni, C. Rocchini and R. Scopigno. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum*, vol. 17(2): 167-174, 1997. http://vcg.iei.pi.cnr.it/metro.html.

[4] J. El-Sana and Y-J. Chiang. External memory view-dependent simplification. *Computer Graphics Forum*, vol. 19(3): 139-150, August 2000.

[5] M. Garland and P. Heckbert. Surface simplification using quadric error metrics. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 209-216. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-201-32230-7.

[6] M. Garland. Multi-resolution modeling: Survey and future opportunities. In State of the Art Report, *Eurographics*, pages 111-131, 1999.

[7] H. Hoppe. Smooth view-dependent level-of-detail control and its application to terrain rendering. *Proceedings IEEE Visualization*, pages 35-42, IEEE October 1998.

[8] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade and D. Fulk. The Digital Michelangelo Project: 3D scanning of large statues. In Kurt Akeley, editor, *SIGGRAPH 2000 Conference Proceedings*, Annual Conference Series, pages 131-144. ACM SIGGRAPH, Addison Wesley, July 2000. ISBN 0-201-48564-8.

[9] P. Lindstrom. Out-of-Core simplification of large polygonal models. In Kurt Akeley, editor, *SIGGRAPH 2000 Conference Proceedings*, Annual Conference Series, pages 259-262, ACM SIGGRAPH, Addison Wesley, July 2000. ISBN 0-201-48564-8.

[10] P. Lindstrom and C. Silva. A memory insensitive technique for large model simplification. In T. Ertl, K. Joy and A. Varshney, editors, *Proceedings IEEE Visualization*, pages 121-126, IEEE, October 2001.

[11] D. Luebke. A developer's survey of polygonal simplification algorithms. *IEEE Computer Graphics and Applications*, vol. 21(3): 24-35, May – June 2001.

[12] R. Ronfard, and J Rossignac. Full-range Approximation of Triangulated Polyhedra. *Computer Graphics Forum*. vol. 15(3): 67-76 and 462, 1996.

[13] J. Rossaignac and P. Borrel. Multi-resolution 3D approximation for rendering complex scenes. *Modeling in Computer Graphics*, pages 455-465, 1993.

[14] V. Salamon, P. Lu, B. Watson, D. Brodsky and D. Gomboc. A case study in improving memory locality in model simplification: metrics and performance. In B. Monien, V.K. Prasanna and S. Vajpeyam editors, *High Performance Computing Conference Proceedings*, pages 137-148, Springer Verlag, December 2001. ISBN 3-540-43009-1.

[15] E. Shaffer and M. Garland. Efficient adaptive simplification of massive meshes. In T. Ertl, K. Joy and A. Varshney, editors, *Proceedings IEEE Visualization*, IEEE October 2001.

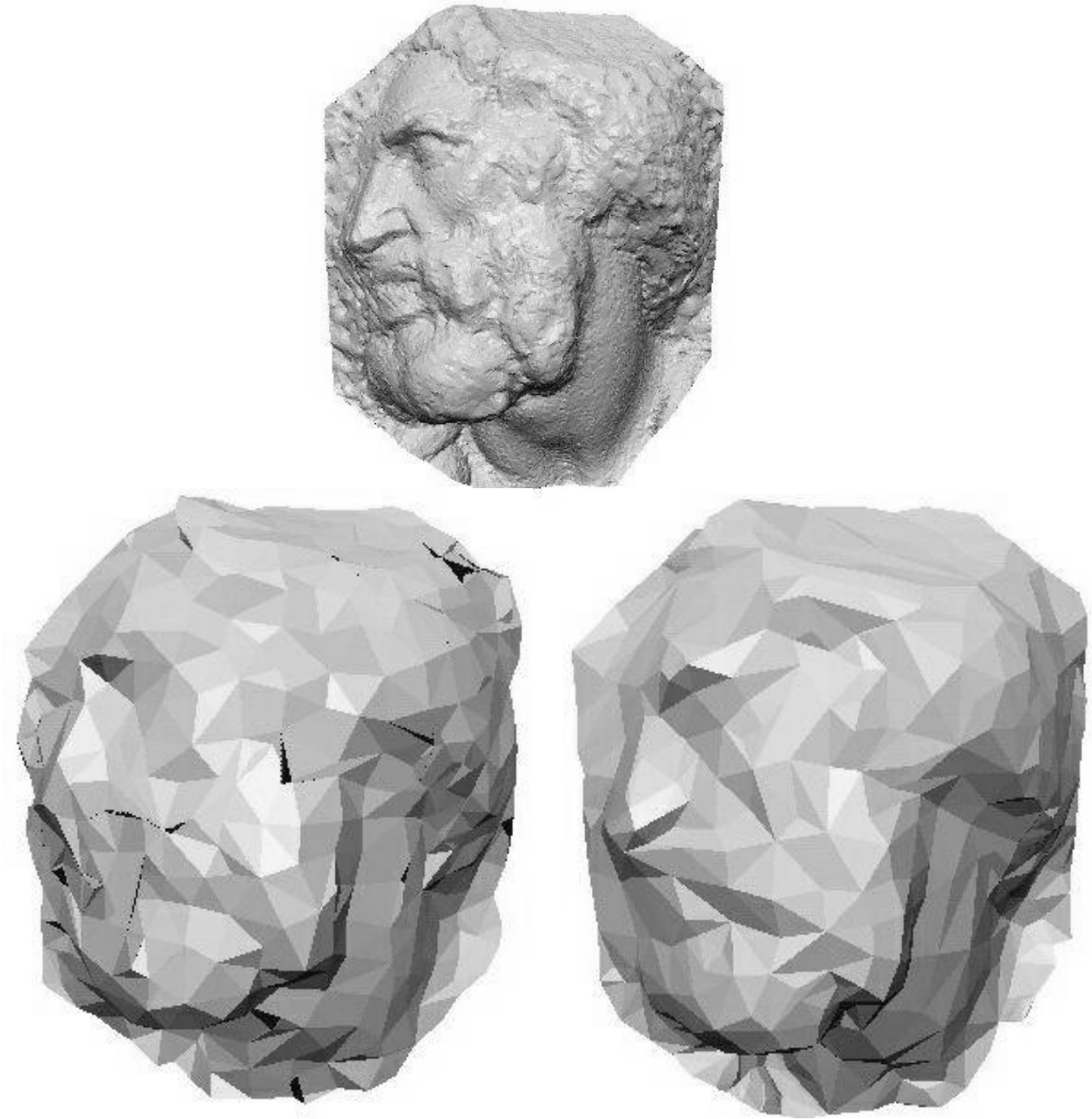

Figure 4: Sensitivity to mesh boundaries. On the bottom row, the St. Matthew Face is simplified to roughly 500 vertices by adaptive simplification [15] (left) and `VMRSimp` (right). Some normals were flipped by adaptive simplification – this could easily be remedied in the algorithm itself. On the top, the face is simplified to 50,000 vertices, showing a better approximation of the original boundaries.

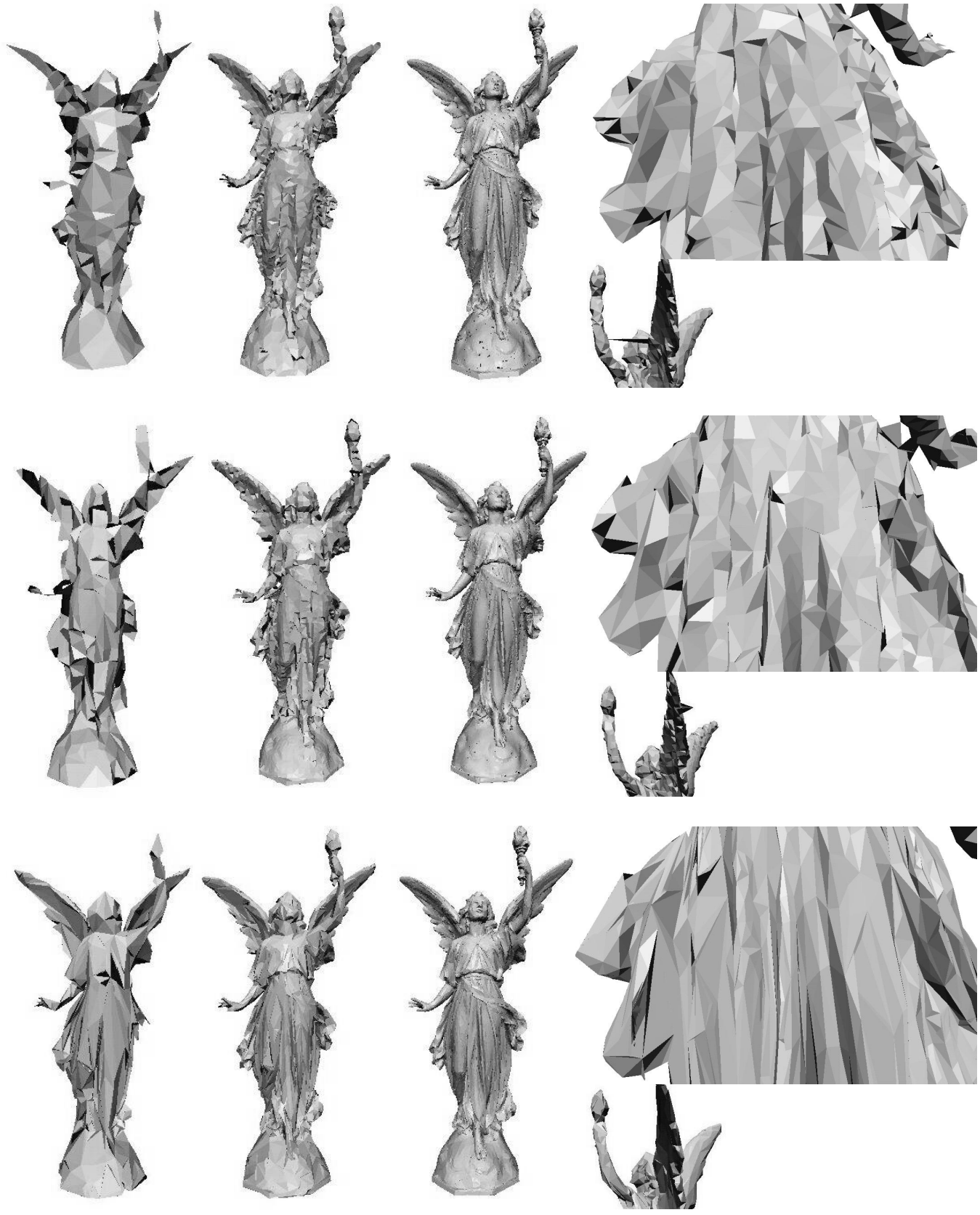

Figure 5: Lucy model simplified to roughly 500 (first column), 5000 (second column) and 50,000 (third column) vertices. The models in the top row were simplified with Shaffer and Garland's [15] algorithm, the middle row with Lindstrom's [9] algorithm, and the bottom row with `VMRSimp`. Each row includes a close-up of Lucy's dress and a different view of her wing.

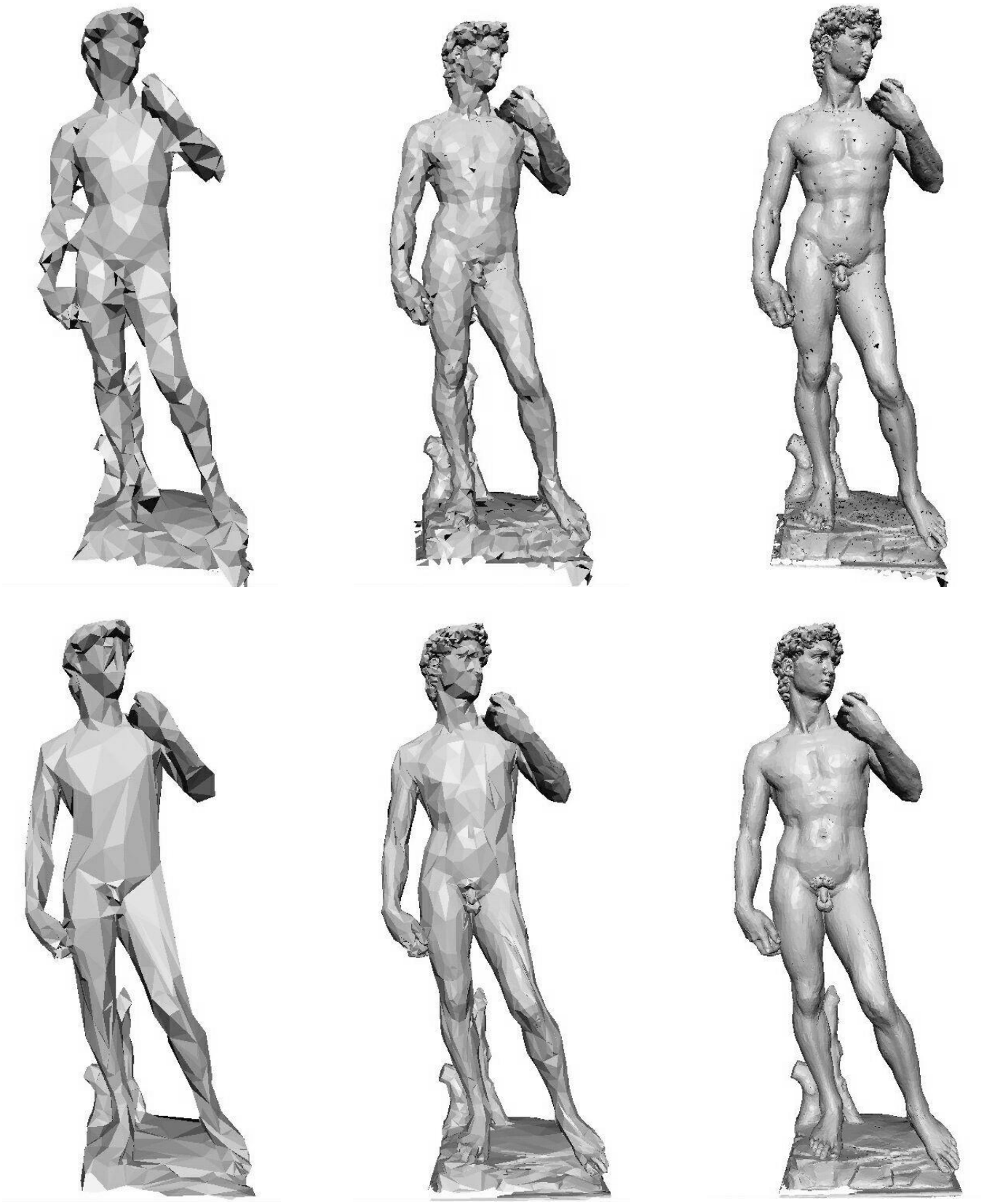

Figure 6: David model (28 million vertices) simplified to roughly 1000 vertices (left), 5000 vertices (middle) and 50,000 vertices (right), using `VMRSimp` (bottom) and adaptive simplification (top). Note the extraneous joins at the groin and armpits in adaptive simplification. Again, the flipped normals in adaptive simplification could easily be fixed.