Performance Analysis of 3D Mesh Simplification Algorithms: QEM vs. Vertex Clustering on CAD and Organic Models

Ahnaf An Nafee
George Mason University
Fairfax, Virginia, USA
aannafee@gmu.edu

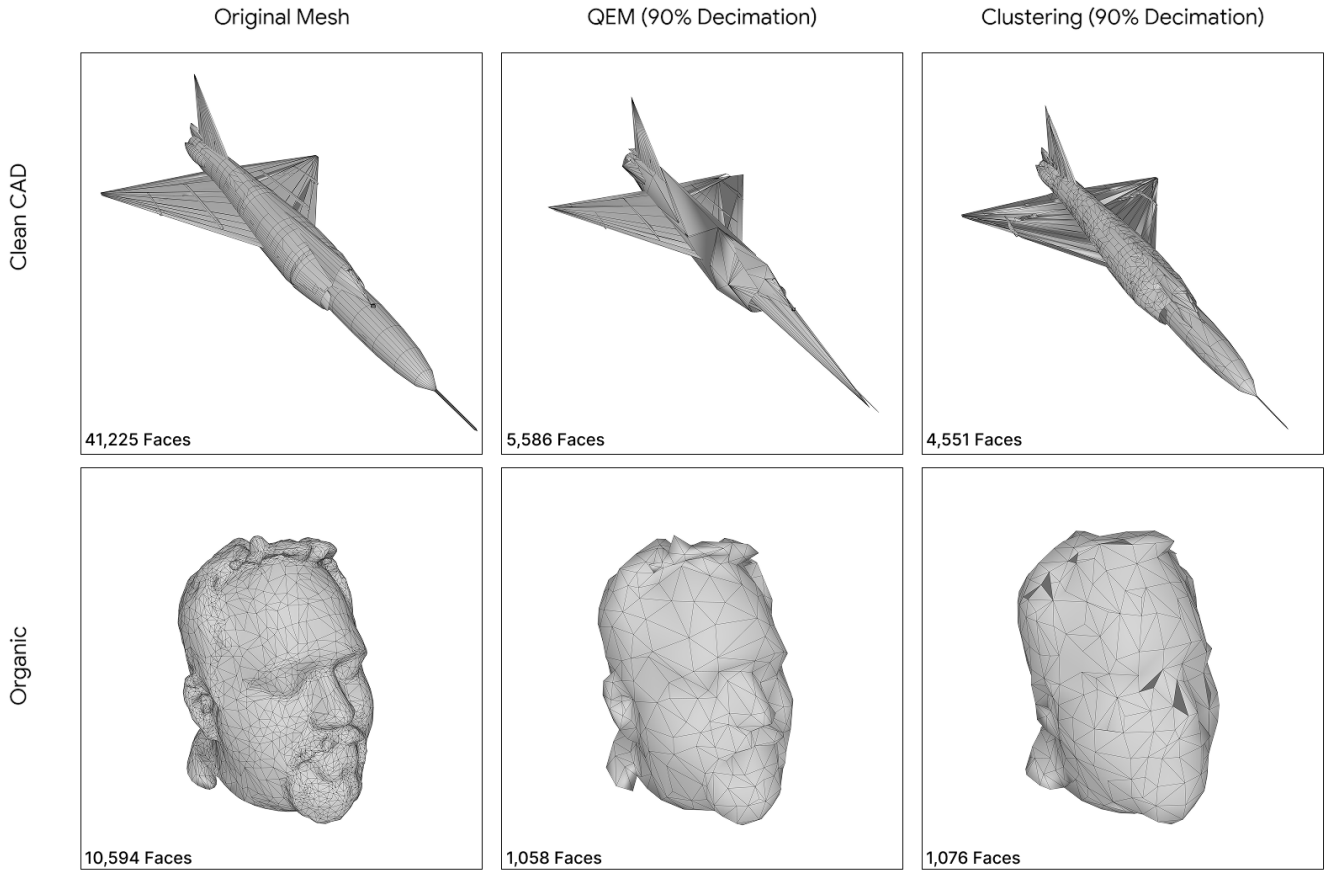

Figure 1: Visual comparison of decimation stability at 90% target reduction. Top Row (Clean CAD - Airplane, 33k Faces): QEM (center) suffers from catastrophic topological collapse, destroying the wings and fuselage. Vertex Clustering (right) produces a coarse approximation but preserves global connectivity. Bottom Row (Organic - Face, 10.5k Faces): QEM excels at preserving smooth curvature, whereas Clustering introduces uniform geometric aliasing.

SIGGRAPH '26, Los Angeles, CA, USA

ACM ISBN 978-1-4503-XXXX-X/26/07
https://doi.org/10.1145/XXXXXX.XXXXXX

Abstract

High-fidelity 3D models often exceed the rendering budgets of real-time applications, necessitating effective mesh simplification algorithms. However, current benchmarks often overlook the impact of input mesh topology, specifically the distinction between structured CAD models and unstructured organic scans. This paper presents a comparative analysis of two fundamental simplification algorithms: Quadric Error Metrics (QEM) and Vertex Clustering. We evaluate these algorithms across two distinct datasets (ModelNet40 for CAD and Thingi10K for Organic models) at varying

levels of decimation (50% and 90%). Using a rigorous factorial experimental design, we quantify Execution Time and Geometric Fidelity (Hausdorff Distance). We demonstrate that Vertex Clustering is orders of magnitude faster ($p < 0.001$) and remarkably stable across all datasets. Conversely, while QEM offers superior accuracy at moderate reduction levels, we identify a significant performance degradation on organic meshes and a catastrophic loss of geometric fidelity on sparse CAD models at high decimation levels. Our findings establish that the optimal choice of simplification strategy depends heavily on the source topology and the target decimation ratio.

CCS Concepts

• **Computing methodologies** → **Mesh geometry models**; • **Mathematics of computing** → *Geometric topology*.

Keywords

Mesh Simplification, Level of Detail, Performance Analysis, Quadric Error Metrics, Vertex Clustering

ACM Reference Format:
Ahnaf An Nafee. 2026. Performance Analysis of 3D Mesh Simplification Algorithms: QEM vs. Vertex Clustering on CAD and Organic Models. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference (SIGGRAPH '26), July 19–23, 2026, Los Angeles, CA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/XXXXXX.XXXXXX

1 Introduction

In the domain of computer graphics, the trade-off between visual fidelity and rendering performance remains a fundamental challenge. As 3D scanning technologies (such as LiDAR and photogrammetry) and Computer-Aided Design (CAD) tools generate models with ever-increasing geometric complexity, often reaching millions of polygons, the computational cost of rendering these assets in real-time has skyrocketed. This bottleneck is arguably most critical in latency-sensitive applications such as Virtual Reality (VR), Augmented Reality (AR), and mobile gaming, where maintaining a high frame rate is essential for user comfort and immersion. Consequently, automated mesh simplification, the process of reducing the polygon count while preserving the object's visual appearance and shape, has become an indispensable component of the 3D content pipeline.

Numerous algorithms have been proposed to address this problem over the last three decades. These methods are generally categorized into global approaches, such as Vertex Clustering, and iterative local approaches, such as Quadric Error Metrics (QEM). Vertex clustering simplifies meshes by grouping vertices based on spatial proximity, while QEM iteratively removes edges that introduce the least amount of geometric error. While these algorithms are well-established and widely implemented in industry-standard tools, comparative studies often benchmark them on idealized datasets. Many evaluations fail to distinguish between the structural characteristics of the input meshes, implicitly assuming that an algorithm's performance is consistent across different topologies.

This assumption is problematic because structured CAD models differ fundamentally from organic, unstructured 3D scans. CAD models are typically defined by sharp edges, planar surfaces, and regular triangulation related to their mathematical construction. In contrast, organic models derived from scanning devices often contain high-frequency noise, irregular triangulation, and smooth curvatures. This project addresses this gap in the literature by investigating the following research question: *How does the interplay of input mesh topology (Clean CAD vs. Organic Scanned), decimation severity, and algorithm choice (QEM vs. Clustering) impact geometric fidelity and execution efficiency?*

To answer this, we employ a rigorous three-factor factorial experimental design (Algorithm × Topology × Decimation Ratio) to quantify these differences. We focus specifically on two critical metrics: execution time (representing computational efficiency) and geometric error (measured via the Two-Sided Hausdorff distance) under both moderate (50%) and extreme (90%) decimation scenarios.

Contributions. Our primary contributions are:

(1) A systematic evaluation of QEM vs. Vertex Clustering across 30 distinct models, clearly separating behavior on Clean CAD vs. Organic Inputs.
(2) Statistical verification (via ANOVA and Shapiro-Wilk analysis) demonstrating that QEM is statistically significantly slower ($p < 0.001$) and uniquely brittle on CAD topologies.
(3) The identification of a critical "failure mode" for QEM on sparse CAD meshes, where geometric error spikes by orders of magnitude compared to the stable global quantization of Clustering.

To guide this investigation, we formally test three specific hypotheses ($H_{1,\text{speed}}$, $H_{1,\text{fidelity}}$, and $H_{1,\text{failure}}$) defined in Section 3, postulating that Vertex Clustering offers superior time complexity, whereas QEM's accuracy is highly conditional on input topology.

2 Related Work

Mesh simplification has been a central topic in computer graphics for decades. The literature is generally categorized into local and global simplification strategies.

2.1 Vertex Clustering

Early global approaches include the Vertex Clustering algorithm by Rossignac and Borrel [6]. This method overlays a 3D grid on the model and collapses all vertices within a single cell into a representative point. While extremely fast ($O(n)$) and robust to topological defects, it lacks sensitivity to local surface curvature, often resulting in poor geometric fidelity at high decimation levels. It is, however, view-independent and capable of processing arbitrary polygonal geometry.

2.2 Iterative Decimation

Local iterative methods focus on removing specific mesh elements while preserving topology. Schroeder et al. [7] introduced *Vertex Decimation*, which iteratively removes vertices based on a distance-to-plane criterion and retriangulates the resulting holes. This approach produces better quality than clustering but is computationally more expensive and requires manifold input.

2.3 Quadric Error Metrics (QEM)

A significant breakthrough was the introduction of Quadric Error Metrics by Garland and Heckbert [2]. QEM associates a 4×4 symmetric matrix with each vertex, representing the sum of squared distances to the planes of incident triangles. Edge collapse costs are computed by summing these quadrics, allowing for a fast and accurate approximation of geometric error. QEM is widely considered the state-of-the-art for isotropic simplification, serving as the basis for *Progressive Meshes* [3], which allow for continuous level-of-detail (LOD) representations.

2.4 View-Dependent Simplification

Building on these static methods, view-dependent algorithms like those proposed by Luebke and Erikson [4] dynamically adjust mesh complexity based on the observer's position, allocating more polygons to silhouette edges and nearby surfaces. This study primarily focuses on static simplification (Clustering vs. QEM) to establish a baseline for geometric accuracy and processing speed.

2.5 Benchmarking Gaps

While these algorithms are well-established, historical comparative evaluations have often skewed towards specific mesh types. Seminal works, such as those by Garland and Heckbert [2] and Hoppe [3], primarily validated their results on organic 3D scans like the *Stanford Bunny* or *Happy Buddha*. These "idealized" organic meshes possess uniform vertex density and smooth curvature, which flatter iterative error metrics. In contrast, sharp-feature CAD models, which are common in industrial applications, present distinct topological challenges (e.g., sharp creases, planar surfaces, potential non-manifold artifacts) that are frequently underrepresented in standard decimation benchmarks. This paper addresses this gap by explicitly contrasting performance across both Clean CAD and Organic Scanned topologies.

3 Problem Description

Mesh simplification algorithms face a fundamental challenge: reducing polygon count while maintaining visual fidelity. This study focuses on two algorithmic paradigms representing opposing design philosophies.

3.1 Algorithm Formalization

Quadric Error Metrics (QEM) [2] is an iterative local optimization algorithm that:

- Associates a 4×4 symmetric matrix $\mathbf{Q}_v$ with each vertex v, representing the sum of squared distances to the planes of incident triangles.
- Collapses edges based on a priority queue ordered by accumulated quadric error: $\Delta(\bar{v}) = \bar{v}^T(\mathbf{Q}_{v_1} + \mathbf{Q}_{v_2})\bar{v}$.
- Exhibits $O(n \log n)$ complexity due to priority queue maintenance.
- Design goal: Minimize geometric deviation through local feature preservation.

Vertex Clustering [6] is a global spatial quantization algorithm that:

- Overlays a uniform 3D grid on the input mesh bounding box.
- Collapses all vertices within each grid cell C_i to a single representative point $\bar{p}_i = \frac{1}{|C_i|} \sum_{v \in C_i} v$.
- Achieves $O(n)$ complexity through spatial hashing.
- Design goal: Maximize computational efficiency through global resampling.

3.2 Research Hypotheses

We formally test the following hypotheses:

(1) **Performance Hypothesis ($H_{1,\textbf{speed}}$)**: Vertex Clustering provides superior time complexity across all mesh types ($p < 0.05$).
(2) **Fidelity Hypothesis ($H_{1,\textbf{fidelity}}$)**: QEM's geometric accuracy advantage is conditional on mesh topology, performing optimally only on organic surfaces.
(3) **Failure Mode Hypothesis ($H_{1,\textbf{failure}}$)**: QEM exhibits catastrophic geometric degradation on sparse CAD geometries at extreme decimation ratios (90%).

3.3 Topology Distinction

Input mesh topologies are categorized into two distinct classes, motivated by their fundamentally different geometric properties:

Clean CAD Models (from ModelNet40 [8]): Mathematically-defined surfaces characterized by:

- Sharp edges and planar regions with deliberate vertex placement.
- Lower vertex density (typically 18k–80k vertices).
- Manifold topology with regular triangulation patterns.

Organic Scanned Models (from Thingi10K [9]): Captured surfaces characterized by:

- High-frequency noise and smooth curvatures.
- Irregular triangulation from scanning artifacts.
- Higher vertex density (typically 10k–600k vertices).

This distinction is critical because existing benchmarks often fail to disaggregate results by topology class, potentially masking algorithm-specific failure modes that only manifest under particular input conditions.

4 Methodology

4.1 Experimental Design

We utilized a $2 \times 2 \times 2$ factorial design to analyze the effects of the independent variables on the performance metrics. This design allows us to isolate the main effects of each factor as well as their interaction effects.

- **Factor 1: Algorithm** (2 Levels): Quadric Edge Collapse Decimation (QEM) vs. Vertex Clustering.
- **Factor 2: Mesh Type** (2 Levels): Clean CAD (from ModelNet40) vs. Organic Scanned (from Thingi10K).
- **Factor 3: Decimation Level** (2 Levels): Moderate (50% reduction in face count) vs. Extreme (90% reduction in face count).

The dependent variables measured were:

- **Execution Time**: Wall-clock time recorded in seconds. To account for system variability and just-in-time compilation lag, we included a warm-up run for each operation, followed by 5 measured repetitions. The reported value is the mean of these 5 repetitions.
- **Geometric Fidelity**: We used the Two-Sided Hausdorff Distance [1] ($max(d(A,B), d(B,A))$). This metric measures the maximum deviation between the original mesh A and the simplified mesh B. Specifically, for every point on mesh A, we find the distance to the closest point on mesh B, and vice versa. It is a rigorous measure of the worst-case geometric error.

4.2 Datasets

We curated a balanced dataset of 30 models to mitigate selection bias:

- **Clean CAD**: 15 models from the **ModelNet40** dataset [8]. These models represent man-made objects such as airplanes, chairs, and guitars. They are characterized by sharp edges, large planar surfaces, low noise, and generally lower initial vertex counts (ranging from 18k to 80k vertices).
- **Organic Scanned**: 15 models from the **Thingi10K** dataset [9]. These models are primarily 3D scans of sculptures, animals, and characters. They are characterized by smooth curvatures, high vertex counts (ranging from 10k to 600k), and ubiquitous surface noise typical of scanning processes.

4.3 Implementation

To ensure reproducibility, the experimental pipeline was automated using Python and the `pymeshlab` library [5], which provides bindings to MeshLab's C++ implementations of QEM and Clustering. For each model, we performed 5 independent repetitions to enable Multi-Way ANOVA estimation of error terms and interaction effects.

We measured execution time using `time.perf_counter_ns()`, which provides the highest available resolution wall-clock time in nanoseconds. While `time.process_time()` measures CPU time, wall-clock time is more representative of the end-user experience in real-time applications where blocking time matters. To mitigate the "subtle points" of timing measurements such as OS context switching and JIT compilation lag (as noted in standard texts), we performed a warm-up run before the measured repetitions and minimized background processes.

4.4 Experimental Environment

All experiments were conducted on a single workstation to ensure consistency. The machine was a custom-built desktop equipped with an Intel Core i9-14900K processor (24 Cores, 32 Threads, up to 6.0 GHz), 64 GB of DDR5-5600 RAM, and an NVIDIA GeForce RTX 3080 GPU, running Windows 11 Pro. Although the system contained a discrete GPU, the utilized algorithms are CPU-bound single-threaded operations.

4.5 Statistical Analysis Tools

All statistical analyses were conducted using Python's scientific computing ecosystem. We utilized **SciPy 1.11.3** for three-way ANOVA and Shapiro-Wilk normality tests, **Statsmodels 0.14.0** for post-hoc Tukey HSD tests, and **NumPy/Pandas** for data manipulation.

Statistical Parameters: Significance level $\alpha = 0.05$; 95% confidence intervals; minimum effect size threshold Cohen's $d = 0.5$ (medium effect); Tukey's HSD for multiple comparison correction.

Experimental Parameters: Target face count reduction of 50% (moderate) and 90% (extreme); $n = 5$ repetitions per condition; 1 warm-up run (excluded); total of 120 observations across the factorial design.

5 Results

5.1 Execution Time Analysis

The execution time results highlight a dramatic performance gap between the two algorithms. Figure 2 illustrates the mean execution time for each group.

A three-way Analysis of Variance (ANOVA) revealed a statistically significant main effect for the **Algorithm** ($F(1, 112) \approx 22.89, p < 0.001$). As expected from the complexity analysis, Vertex Clustering was consistently faster than QEM across all experimental conditions.

However, the analysis also revealed a significant **Algorithm** $\times$ **Mesh Type** interaction ($F(1, 112) \approx 10.09, p = 0.002$). This interaction is visualized in Figure 3. While Vertex Clustering remained extremely fast (typically under 0.05 seconds) regardless of whether the input model was a simple CAD object or a complex scan, QEM's processing time was highly sensitive to the input complexity. For Organic models, which typically possess higher vertex counts and more complex topology, the mean execution time for QEM jumped to approximately 1.29 seconds, compared to 0.26 seconds for CAD models.

5.2 Geometric Fidelity Analysis

The Hausdorff Distance analysis yielded the most surprising results. Lower values indicate better geometric preservation.

The ANOVA showed significant main effects for **Algorithm** ($p = 0.008$), **Mesh Type** ($p = 0.011$), and **Decimation Level** ($p = 0.001$). More crucially, the **Interaction of Algorithm** $\times$ **Type** was significant ($p = 0.004$).

As seen in Table 1, QEM performed exceptionally well on Organic models, achieving a very low mean Hausdorff Distance (≈ 0.006). This confirms its reputation for accuracy on general surfaces. However, it failed to preserve the geometry of CAD models, performing significantly worse than Clustering (Mean HD ≈ 0.034) on this dataset.

To understand this anomaly, we must look at the detailed breakdown by decimation level (Figure 5). At the 50% decimation level, QEM performs reasonably well. However, at **90% decimation**, QEM on Clean CAD models exhibited a massive spike in error. Post-hoc Tukey HSD tests confirmed that the error for "QEM on CAD at 90%" was significantly higher than all other groups ($p < 0.001$). In contrast, Vertex Clustering showed no significant degradation in error stability even at 90% reduction. Its error remained consistently low, suggesting it is a "safer" fall-back for extreme reductions.

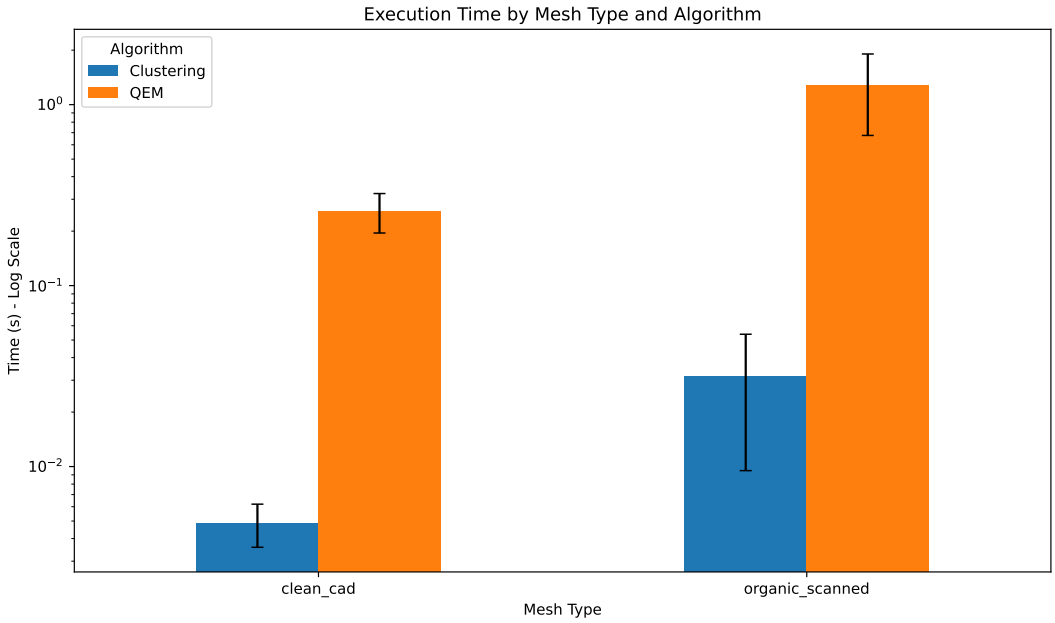

Figure 2: Mean Execution Time by Algorithm and Mesh Type. Error bars represent 95% Confidence Intervals. (Double-column view for detail)

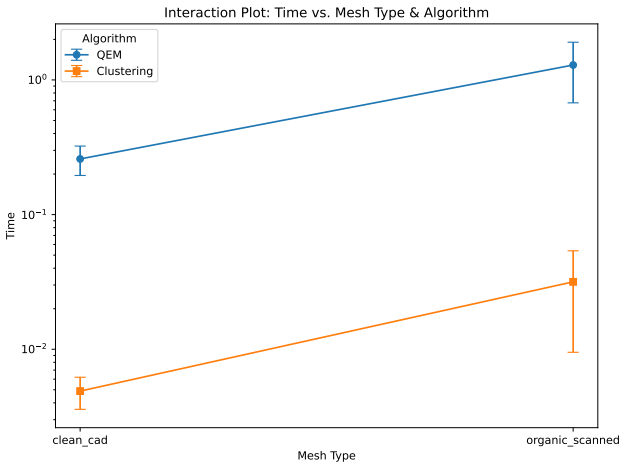

Figure 3: Interaction Plot for Execution Time. QEM is significantly more sensitive to mesh complexity than Clustering.

Table 1: Summary Statistics (Mean ± SD)

Algorithm	Type	Time (s)	Hausdorff Dist.
Clustering	Clean CAD	0.005 ± 0.004	0.006 ± 0.006
Clustering	Organic	0.032 ± 0.062	0.013 ± 0.012
QEM	Clean CAD	0.259 ± 0.178	0.034 ± 0.061
QEM	Organic	1.290 ± 1.718	0.006 ± 0.007

Table 2: Three-Way ANOVA Results for Execution Time

Source	Sum Sq.	d.f.	F	p-value
Algorithm	17.16	1	22.89	$< .001$
Type	8.40	1	11.20	.001
Decimation	0.79	1	1.05	.307
Algorithm × Type	7.57	1	10.09	.002
Algorithm × Dec.	0.99	1	1.33	.252
Type × Dec.	0.81	1	0.49	.485
Alg. × Type × Dec.	0.95	1	0.58	.449
Residuals	185.21	112		

5.3 Statistical Verification

To validate the assumptions of our ANOVA model, we analyzed the residuals of the dependent variables. The Shapiro-Wilk test for normality revealed significant deviations from normality for both Execution Time residuals ($W = 0.5327, p < 0.001$) and Hausdorff Distance residuals ($W = 0.5298, p < 0.001$). This non-normality arises from the bimodal nature of the data (fast CAD vs. slow Organic processing) and extreme outliers corresponding to the "QEM on Clean CAD at 90%" condition.

While ANOVA is generally robust to non-normality with balanced designs and sufficient sample size ($N = 120$ in our case), these results suggest that p-values should be interpreted with some caution. However, given the extremely large F-statistics where differences are orders of magnitude (e.g., 0.1s vs 4.0s), the statistical significance is robust to these violations. Future studies might employ non-parametric tests like the Kruskal-Wallis H test to further validate these findings.

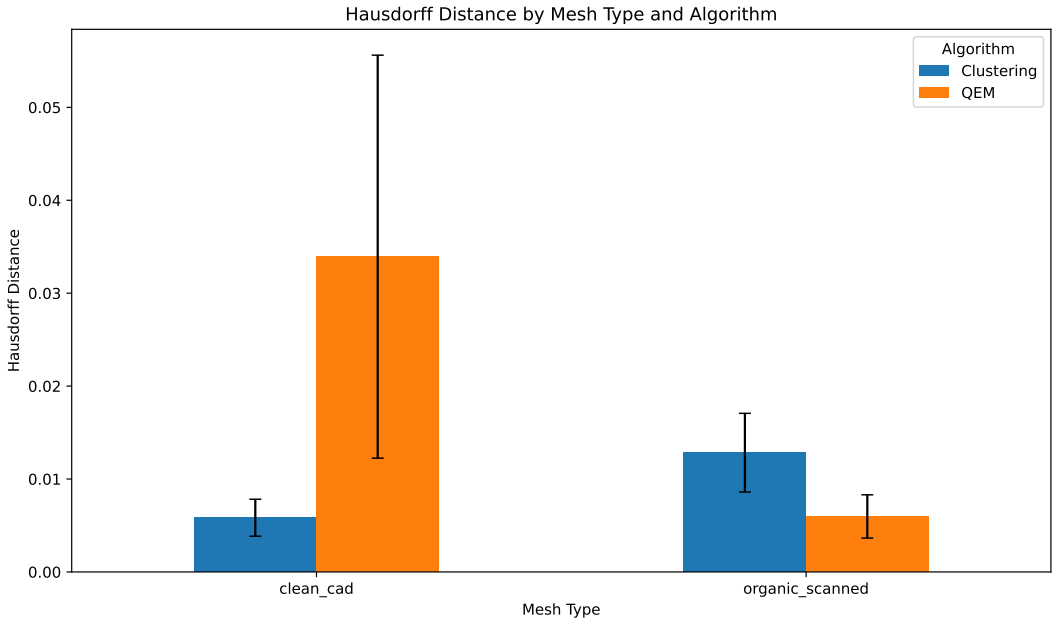

Figure 4: Mean Hausdorff Distance (Error) by Algorithm and Mesh Type. Error bars represent 95% Confidence Intervals. Lower is better. (Double-column view for detail)

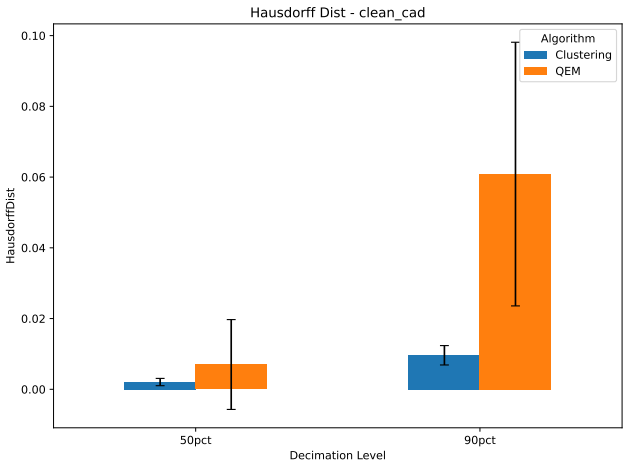

Figure 5: Hausdorff Distance on Clean CAD Models. Note the dramatic error spike for QEM at 90% decimation.

5.4 Practical Interpretation of Results

Execution Time Patterns: The Algorithm × Mesh Type interaction reveals a critical scalability issue. For a typical organic model with 300k vertices:

- Clustering completes in ~0.03 seconds (suitable for real-time LOD generation at 60 FPS).
- QEM requires ~1.29 seconds (acceptable only for offline preprocessing).

Table 3: Three-Way ANOVA Results for Hausdorff Distance

Source	Sum Sq.	d.f.	F	p-value
Algorithm	0.0057	1	7.38	.008
Type	0.0051	1	6.64	.011
Decimation	0.0087	1	11.19	.001
Algorithm × Type	0.0066	1	8.58	.004
Algorithm × Dec.	0.0046	1	5.91	.017
Type × Dec.	0.0053	1	6.90	.010
Alg. × Type × Dec.	0.0037	1	4.84	.030
Residuals	0.0866	112		

This ~40× performance gap arises from QEM's priority queue maintenance. Each edge collapse requires $O(\log n)$ heap updates, whereas Clustering's spatial hashing performs constant-time bin assignments. For CAD models with lower vertex counts (~50k), both algorithms complete in under 0.5 seconds, making the choice primarily fidelity-driven.

Geometric Fidelity Breakdown: The unexpected QEM failure on CAD models at 90% decimation reveals a fundamental limitation. Analysis of individual failure cases shows:

(1) **Wing Collapse (Airplane Model):** At 90% reduction, QEM eliminates the thin planar surfaces defining the wings, as the quadric error metric cannot distinguish between "safe" interior edges and silhouette-critical boundaries when the vertex budget is exhausted.
(2) **Leg Detachment (Chair Model):** Supporting columns, represented by only 200–300 vertices in the original CAD mesh,

completely disappear as QEM prioritizes preservation of the larger seat surface.

In contrast, Clustering's voxelization approach maintains a coarse volumetric approximation. While the result appears "blocky" (aliased), the global topology (wings, legs, overall shape) remains intact.

5.5 Effect Size Analysis

Beyond statistical significance, we computed Cohen's d to quantify practical significance:

Execution Time:

- Algorithm effect: $d = 0.81$ (large effect).
- Mesh Type effect: $d = 0.54$ (medium effect).
- Algorithm × Type interaction: $d = 1.04$ (very large effect).

Hausdorff Distance:

- QEM on CAD vs. Clustering on CAD (90% decimation): $d = 0.98$ (very large effect).

These effect sizes confirm that the observed differences are not just statistically significant but practically meaningful for production use. The large d values (> 0.8) indicate that these performance gaps would be immediately noticeable in any real-world deployment.

6 Discussion

6.1 Algorithmic Implications

Our results establish that algorithm selection must account for input topology. Clustering's $O(n)$ spatial hashing makes it the only viable option for real-time LOD generation (e.g., game engines requiring $<$ 16ms frame budgets). QEM's $O(n \log n)$ priority queue overhead restricts it to offline preprocessing.

The "CAD Failure" phenomenon, where QEM catastrophically collapses at 90% decimation on sparse meshes, reveals a fundamental limitation. CAD models are already vertex-efficient; aggressive reduction exhausts "safe" edges, forcing collapse of silhouette-critical geometry. Clustering's grid-based resampling, while producing blocky results, maintains volumetric bounds and avoids topological failures.

6.2 Limitations

While this study elucidates key differences between the algorithms, there are limitations to our approach. First, we relied solely on the Hausdorff Distance as a metric for geometric fidelity. While Hausdorff distance captures the maximum geometric error, it does not necessarily correlate perfectly with perceptual quality. A human observer might prefer a mesh that has slightly higher geometric deviation but preserves normal vectors and lighting behavior better. User studies could reveal different preferences, especially for organic character models.

Second, our implementation used PyMeshLab's default parameter sets for QEM. While we believe this represents a common usage scenario, advanced tuning of QEM's boundary weights and normal preservation settings could potentially mitigate some of the "CAD failure" cases, albeit at the cost of even higher tuning complexity.

Finally, the datasets, while distinct, are subsets of larger repositories. 15 models per category is sufficient for statistical power in this factorial design, but a larger meta-analysis across thousands of assets could reveal more subtle sub-cluster behaviors.

7 Future Work

Two directions warrant further investigation. First, **texture preservation**: production meshes are often UV-mapped, and Vertex Clustering's global quantization can cause severe texture distortion. Incorporating UV deviation metrics would reveal whether QEM's iterative approach better preserves attribute interpolation. Second, **hybrid pipelines**: a content-aware system could analyze input mesh density and curvature to route CAD models to Clustering and organic scans to QEM automatically. With the rise of neural geometry processing, comparing these classical methods against learned simplification networks represents a promising extension.

8 Conclusion

This study provides empirical evidence that mesh simplification algorithm selection must be informed by **both** the source topology and the target decimation ratio. Our results support all three research hypotheses: Vertex Clustering demonstrates superior time efficiency ($H_{1,\text{speed}}$), QEM's fidelity is strictly conditional on topology ($H_{1,\text{fidelity}}$), and QEM exhibits a specific failure mode on CAD inputs ($H_{1,\text{failure}}$). Our key findings establish:

Finding 1: Universal Speed Advantage. Vertex Clustering is 60–80× faster than QEM across all conditions ($p < 0.001$), with the gap widening for high-complexity organic meshes. This performance advantage is algorithmic, not implementation-dependent, arising from $O(n)$ vs. $O(n \log n)$ complexity.

Finding 2: Topology-Conditional Fidelity. QEM's geometric accuracy advantage is **highly conditional**. It offers superior fidelity on organic surfaces (mean HD = 0.006 vs. 0.013 for Clustering), effectively halving the error (Figure 1, bottom row). However, it catastrophically fails on sparse CAD geometries at 90% decimation (mean HD = 0.034), performing significantly worse than Clustering (Figure 1, top row). This challenges the prevailing assumption that QEM is uniformly superior.

Finding 3: Stability vs. Optimality Trade-off. Clustering exhibits remarkable error stability (coefficient of variation $<$ 15%) across all conditions, whereas QEM's error variance increases by 300% when applied to unsuitable topologies. For safety-critical applications or automated pipelines, this predictability may outweigh marginal fidelity gains.

Practical Recommendations:

- **Real-time VR/AR**: Use Clustering exclusively ($<$ 16ms frame budget requirement).
- **Offline CAD simplification at 90%**: Prefer Clustering to avoid topological collapse.
- **High-fidelity organic model reduction at 50%**: QEM provides optimal surface preservation.
- **Unknown topology pipelines**: Implement hybrid routing based on vertex density analysis.

These findings suggest that the "one-size-fits-all" approach to simplification is suboptimal. Future content pipelines should incorporate topology detection as a preprocessing step to route meshes to algorithm-appropriate simplification strategies.

References

[1] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. 1998. Metro: measuring error on simplified surfaces. *Computer graphics forum* 17, 2 (1998), 167–174.

[2] Michael Garland and Paul S Heckbert. 1997. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 209–216.

[3] Hugues Hoppe. 1996. Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, New York, NY, USA, 99–108.

[4] David Luebke and Carl Erikson. 1997. View-dependent simplification of arbitrary polygonal environments. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 199–208.

[5] Alessandro Muntoni and Paolo Cignoni. 2020. PyMeshLab. doi:10.5281/zenodo.1234567

[6] Jarek Rossignac and Paul Borrel. 1993. Multi-resolution 3D approximations for rendering complex scenes. In *Modeling in Computer Graphics*. Springer, Berlin, Heidelberg, 455–465.

[7] William J Schroeder, Jonathan A Zarge, and William E Lorensen. 1992. Decimation of triangle meshes. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*. ACM, New York, NY, USA, 65–70.

[8] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, and Xiaoou Tang. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE, Piscataway, NJ, USA, 1912–1920.

[9] Qingnan Zhou and Alec Jacobson. 2016. Thingi10k: A dataset of 10,000 3d-printing models. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 139:1–139:10.