An Adaptive Machine Learning-Based Level-of-Detail Selection Framework for Real-Time 3D Rendering

Saachi Ghadge

School of Engineering and Technology, MIT World Peace University, Pune, Maharashtra, India

DOI: https://doi.org/10.51244/IJRSI.2026.1311000049

Received: 29 October 2026; Accepted: 03 November 2026; Published: 22 November 2026

ABSTRACT

Real-time rendering of complex three-dimensional scenes can be computationally expensive because highly detailed meshes contain large numbers of polygons. Level of Detail (LOD) techniques reduce this cost by substituting simpler mesh representations when full geometric detail is unnecessary. This paper investigates supervised machine-learning-based selection among discrete LOD levels and retains a broader proposed framework based on scene and rendering features. Two distinct proof-of-concept experiments are reported. The manuscript's earlier synthetic experiment used 6,000 synthetically generated samples and reported 81.0% Random Forest test accuracy (macro F1 = 0.788), compared with 22.8% for its distance-only baseline. A separate, subsequently implemented tree-asset experiment used 100 PSNR-labelled observations and four features—camera distance, estimated projected height in pixels, camera elevation, and camera azimuth—to train a Random Forest classifier. In that experiment, the Random Forest achieved 87% accuracy under 5-fold stratified cross-validation (precision = 0.8031, recall = 0.7774, F1 = 0.7892), compared with 81% for a Decision Tree, 65% for a distance-only baseline, and 73% for a majority-class baseline. The model was integrated into a Blender scene containing 81 tree instances.

All 81 instances were assigned LOD2, and the Blender JSON inference implementation matched the original scikit-learn predictions for all 81 instances. The ML scene contained 33,048 polygons, compared with 105,786 for the no-LOD scene and 15,988 for the distance-based scene. Thus, ML-based selection reduced polygons by 68.76% relative to no LOD, but did not outperform distance-based LOD in this scene. These results demonstrate a working classification and Blender-integration proof of concept, not a validated real-time performance or perceptual-quality improvement. Broader asset coverage, dynamic runtime measurements, and visual-quality validation remain necessary.

Keywords: Level of Detail; Machine Learning; Real-Time Rendering; Random Forest; Blender; PSNR; Mesh Simplification; Adaptive Rendering; Computer Graphics

INTRODUCTION

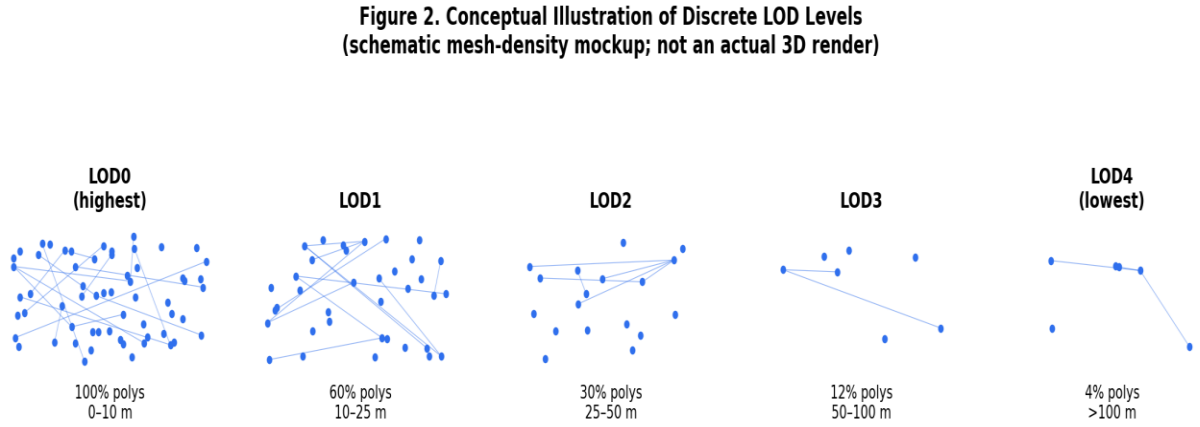
Background

The increasing complexity of modern three-dimensional applications has created significant challenges for real-time rendering. Video games, virtual reality, augmented reality, digital twins, engineering simulations, architectural visualization, and interactive 3D applications frequently contain large numbers of detailed objects. Rendering every object at its maximum geometric resolution can result in excessive GPU workload, increased memory consumption, and reduced frame rates.

Level of Detail (LOD) is a widely used technique for addressing this problem. Instead of rendering an object using a single, highly detailed representation, multiple representations with different geometric complexities are generated in advance. A high-detail representation is used when an object is close to the camera or occupies a large screen area, while increasingly simplified representations are substituted as the object becomes less visually significant.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue XI November 2026

Figure 1. Conceptual illustration of discrete LOD levels — schematic mesh-density mock-up used for explanatory purposes, not a real render.



Problem Statement

Conventional LOD selection frequently depends on manually specified thresholds, particularly camera-distance thresholds. Two objects at the same camera distance may require different LODs because of differences in geometry, projected screen size, visibility, orientation, or visual complexity. Similarly, a rendering system operating under heavy GPU load may benefit from more aggressive simplification than the same system operating with substantial unused GPU capacity. Fixed heuristics cannot easily represent this combined, multi-variable relationship. The problem addressed in this research is therefore: how can a supervised machine-learning model dynamically predict an appropriate geometric Level of Detail for a 3D object using multiple scene and runtime rendering features, while improving rendering efficiency without causing unacceptable visual- quality degradation?

Motivation

As scenes grow richer and hardware targets diversify — from high-end desktop GPUs to mobile and standalone VR/AR headsets — a single fixed heuristic becomes increasingly difficult to tune across devices and content types. A learned LOD-selection model offers a way to absorb this variability directly from data instead of requiring hand-tuned per-project thresholds, and can, in principle, react to transient conditions such as GPU-load spikes that static rules cannot represent.

Research Gap

Traditional mesh-simplification methods have long attempted to reduce geometric complexity while preserving visual or geometric characteristics. Garland and Heckbert's Quadric Error Metrics (QEM) method uses iterative vertex-pair contractions and quadric matrices to produce simplified approximations of complex polygonal models [1]. Later research focused on preserving important geometric structures during simplification; Li and Nan proposed a feature-preserving approach for urban buildings that retains planar structures and sharp contours while reducing mesh complexity [2]. Machine-learning-based research has subsequently addressed several related problems. Neural Geometric LOD learns a neural representation containing multiple LODs [3]; Tamm et al. use supervised learning for LOD-reduction validation rather than LOD selection [4]; neural prefiltering methods apply learned representations to appearance filtering across LODs [5]; and recent Gaussian- Splatting research develops hierarchical or scalable LOD management for that specific representation [8], [9]. However, none of these approaches directly addresses the practical problem investigated here — a lightweight supervised model that directly predicts a discrete geometric LOD from readily measurable runtime scene and rendering features, evaluated explicitly against a conventional distance-based baseline on mesh-based content. Research gap (summary): Existing work provides sophisticated neural representations, mesh-simplification algorithms, perceptual-quality predictors, and representation-specific LOD systems (e.g., for Gaussian Splatting), but there remains scope for a practical,

lightweight, multi-feature supervised-learning framework that predicts geometric LOD selection for conventional mesh-based assets and quantifies the resulting performance-quality trade-off against distance-based LOD.

Proposed Solution Overview

This paper proposes formulating LOD selection as a multi-class classification problem. A feature vector composed of camera distance, projected screen-space size, polygon count, object size, visibility, camera angle, occlusion ratio, current FPS, and GPU utilization is used to predict one of five discrete LOD levels (LOD0-LOD4). Ground-truth labels are generated offline by rendering each candidate LOD and selecting the lowest-cost representation whose visual quality remains above an acceptance threshold, so the model is trained to reproduce a principled cost-quality optimum rather than an arbitrary heuristic.

Novelty

Formulates LOD selection as a direct multi-class classification task over a compact, engine- measurable feature set, rather than a new neural 3D representation.

Uses a cost-quality-constrained ground-truth generation rule (SSIM/PSNR/LPIPS-gated lowest- cost LOD) rather than manually assigned labels.

Evaluates the approach jointly on classification quality, rendering performance, and perceptual visual quality, with asset-level generalization testing and statistical-significance testing.

Includes a feature-ablation study isolating the marginal value of screen-space size, polygon count, and workload features over a distance-only baseline.

Contributions

The main contributions of this work are:

- A formulation of adaptive geometric LOD selection as a supervised multi-class classification
- problem driven by nine runtime scene and rendering features.
- A cost-quality-constrained procedure for generating ground-truth optimal LOD labels using SSIM/
- PSNR/LPIPS thresholds, removing reliance on hand-set heuristic labels.

Research Questions and Hypotheses

The following questions and hypotheses define the empirical claims to be tested. They are proposed study questions, not findings; each hypothesis will be assessed only after the corresponding real-engine experiment has been completed.

Research question	Question
RQ1	How accurately do Decision Tree, Random Forest, XGBoost, and MLP models predict the quality-constrained LOD label from the proposed runtime feature vector?
RQ2	Do multi-feature classifiers outperform a distance-only LOD baseline on held-out rendered

	observations when both are evaluated on the same assets and camera conditions?
RQ3	How well do the trained models generalize to assets and, where feasible, asset categories excluded from training?
RQ4	What is the contribution of screen-space size, polygon count, visibility/occlusion, and workload features to predictive performance?
RQ5	What inference overhead and LOD-switching behaviour does the classifier introduce, and does temporal hysteresis improve stability without unacceptable quality loss?

Hypothesis	Testable statement
H1	A classifier using the full multi-feature vector will achieve higher predictive performance than a distance-only baseline on asset-disjoint rendered test data.
H2	Tree-based models (particularly Random Forest and/or XGBoost) will provide a favourable balance between predictive performance and inference latency relative to the MLP.
H3	Screen-space size and visibility/occlusion features will provide measurable predictive value beyond camera distance alone.
H4	Performance on unseen assets will be lower than on randomly held-out observations from seen assets, but will remain above the distance-only baseline if the learned relationships generalize.
H5	The measured inference cost will be small relative to any rendering-time savings, and hysteresis will reduce unnecessary LOD transitions without violating the selected quality constraint.

Related Work / Literature Review

This section reviews relevant literature in chronological order, retaining the manuscript's focus on the distinction between generating simplified representations and selecting among them at runtime.

- A comparative evaluation of Decision Tree, Random Forest, XGBoost, and MLP classifiers against a conventional distance-based LOD baseline.
- An evaluation protocol spanning classification metrics, rendering-performance metrics, visual- quality metrics, feature-ablation analysis, and asset-level generalization testing.

- A small-scale synthetic proof-of-concept demonstrating that the proposed classification formulation is learnable and that a multi-feature model substantially outperforms a single- variable distance baseline (81.0% vs. 22.8% accuracy on identical held-out data).

This section reviews relevant literature in chronological, year-wise order, following the recommended structure of stating the method, contribution, and limitation of each work before proceeding to the next.

Literature Review 1 - Takikawa et al. (2021)

In 2021, Takikawa et al. proposed Neural Geometric Level of Detail (NGLOD), an octree-based neural feature-volume representation of implicit surfaces that supports multiple discrete LODs together with continuous LOD interpolation for real-time rendering [3]. Their method demonstrated substantial rendering-efficiency and memory improvements over earlier neural signed-distance-function approaches, and showed that a single trained representation could serve several detail levels. However, the approach requires training a bespoke neural representation for every 3D asset and does not address the separate, complementary problem of choosing among already-available discrete geometric LODs at runtime using scene and workload context — the gap this paper targets.

Literature Review 2 - Li and Nan (2021)

In the same year, Li and Nan proposed a feature-preserving mesh-simplification method for urban building models, using structural constraints during mesh decimation to retain planar regions and sharp contours while reducing polygon count [2]. Their hierarchy of simplified representations is well suited to LOD generation and demonstrates that geometric feature preservation, not polygon count alone, determines perceived quality. The method, however, addresses how LOD representations are generated rather than which representation should be selected for a given camera and workload state, leaving the runtime-selection problem open.

Literature Review 3 - Tamm et al. (2022)

In 2022, Tamm et al. proposed automatic testing and validation of LOD reductions through supervised learning, training deep convolutional networks to detect visual deterioration caused by LOD reduction and thereby automating an otherwise manual quality-assurance process [4]. This work is close in spirit to the present research because it applies supervised learning directly to an LOD-related decision. However, its prediction target is visual-deterioration validity rather than the discrete LOD level itself, and it does not incorporate runtime rendering-workload signals such as GPU utilization or current FPS, both of which this paper's feature vector explicitly includes.

Literature Review 4 - Weier et al. (2023) and Piórkowski et al. (2024)

In 2023, Weier et al. proposed neural prefiltering for correlation-aware levels of detail, learning appearance information across LODs using a sparse multi-level voxel representation to address visibility and shading filtering for physically based rendering [5]. In 2024, Piórkowski, Mantiuk, and Wernikowski extended this line of work by learning to predict the perceptual visibility of rendering deterioration in computer games, providing a data-driven perceptual model for detecting when quality loss becomes noticeable [6].

Both works improve the appearance- and perception-aware side of LOD systems, but both are computationally heavier than a tabular classifier and are oriented toward appearance filtering or deterioration detection rather than direct, lightweight discrete-LOD selection from engine-measurable runtime features.

Literature Review 5 - Joubjang and Rakthanmanon (2024), Shen et al. (2025), and

Kulhanek et al. (2025) In 2024, Joubjang and Rakthanmanon proposed an automatic LOD system for improved rendering that integrates spatial hashing with combined CPU/GPU workload analysis to keep frame rate consistent in game environments [7]. In 2025, Shen et al. introduced LOD-GS, which uses adaptive pruning and growing to provide scalable LOD management for 3D Gaussian Splatting scenes [8], and

Kulhanek et al. proposed LODGE, a hierarchical LOD scheme for large-scale Gaussian Splatting that dynamically selects subsets of Gaussians by camera distance to reduce rendering latency and GPU memory [9]. These recent studies confirm that adaptive, workload-aware LOD management remains an active research direction; however, the workload-aware method [7] is a hand-engineered heuristic rather than a learned classifier, and the Gaussian-Splatting methods [8], [9] are representation-specific rather than generalizable to conventional polygonal mesh assets — motivating the lightweight, mesh-general, learned classification approach proposed in this paper.

Summary of Trends

Across 2021-2025, the literature shows a clear progression from (i) purely geometric mesh- simplification methods, to (ii) neural representations that unify multiple LODs within a single learned model, to (iii) learned quality/appearance predictors for LOD, and most recently to (iv) workload- and representation-aware LOD management for both conventional meshes and Gaussian Splatting scenes.

What remains comparatively underexplored is a lightweight, interpretable, tabular supervised- learning classifier that selects among pre-generated discrete mesh LODs using a compact set of runtime features and is benchmarked directly, on equal terms, against a conventional distance-based heuristic — the specific contribution of this paper.

RESEARCH METHODOLOGY / PROPOSED SYSTEM

System Architecture

The proposed system consists of five stages: (i) 3D asset preparation, (ii) multi-LOD generation using established mesh-simplification algorithms, (iii) runtime feature collection, (iv) offline ground-truth LOD generation, and (v) ML model training and adaptive LOD selection at runtime. Figure 1 shows the architecture and the data flow between stages.

Figure 1. Proposed ML-Based Adaptive LOD Selection Architecture

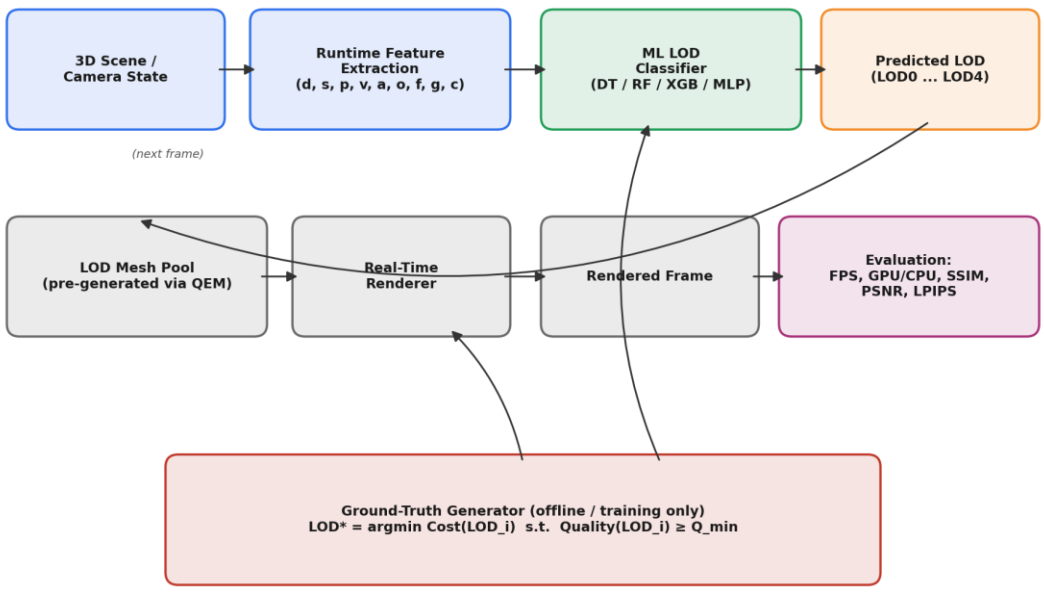


Figure 2. Proposed ML-based adaptive LOD selection architecture. The ground-truth generator (bottom) operates only during offline dataset construction / training; at inference time the trained classifier consumes runtime features directly.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue XI November 2026

Diagram Explanation

The top row represents the online (inference-time) path: per-object runtime features are extracted every frame from the current camera and scene state and passed to the trained classifier, which outputs a predicted LOD used to select a mesh from the pre-generated LOD pool for rendering. The bottom row (ground-truth generator) operates only offline: candidate LODs are rendered, scored for cost and visual quality, and the lowest-cost representation meeting the quality constraint becomes the training label. This separation keeps the online path lightweight — no rendering-quality metric needs to be computed at runtime, only a forward pass through the trained classifier.

Feature Vector and Mathematical Model

The proposed system treats LOD selection as a multi-class classification problem. Given a feature vector X = [d, s, p, v, a, o, f, g, c]

where:

d = camera distance

s = projected screen-space size

p = polygon count

v = visibility

a = camera angle

o = occlusion ratio

f = current FPS

g = GPU utilization

c = object complexity

the model predicts:

$Y \in \{LOD0, LOD1, LOD2, LOD3, LOD4\}$

Ground-Truth Optimal LOD

The optimal LOD is defined as the lowest-cost LOD whose visual quality remains above a predefined

acceptable threshold:

$LOD^* = \text{argmin}_{\{LOD_i\}} \text{Cost}(LOD_i)$ subject to $\text{Quality}(LOD_i) \geq Q_{min}$ where LOD^* is the optimal LOD, Cost(·) represents rendering cost (e.g., triangle count or estimated frame-time contribution), Quality(·) represents visual similarity to the reference (e.g., SSIM), and Q_{min} is the minimum acceptable quality threshold.

For example, if SSIM is used and the acceptance threshold is 0.95, the lowest-polygon LOD satisfying SSIM ≥ 0.95 becomes the ground-truth label for that observation. This formulation prevents the model from trivially learning a camera-distance threshold, since the label depends jointly on cost and measured quality rather than distance alone.

Algorithm

Algorithm 1 summarizes the offline dataset-construction and training procedure; Algorithm 2

summarizes the online inference procedure used during rendering.

Algorithm 1 — Dataset Construction and Model Training

1. For each 3D asset, generate LOD0…LOD4 using Quadric Error Metrics (or an equivalent simplification method). 2. For each sampled camera/scene configuration, compute the runtime feature vector X = [d, s, p, v, a, o, f, g, c]. 3. For each candidate LOD_i, render the object and measure Cost(LOD_i) and Quality(LOD_i) against the LOD0 reference. 4. Select LOD* = argmin Cost(LOD_i) subject to Quality(LOD_i) ≥ Q_min; store (X, LOD*) as a labeled sample. 5. Repeat for all sampled configurations to build the full dataset; split into train / validation / test sets (and, separately, by 3D asset for generalization testing). 6. Train Decision Tree, Random Forest, XGBoost, and MLP classifiers on the training split; tune hyperparameters on the validation split. 7. Evaluate all models on the held-out test split and on the asset-level generalization split using the metrics defined in Section 4

Algorithm 2 — Online LOD Inference

1. At each frame (or at a fixed sampling interval), extract the current runtime feature vector X for each visible object. 2. Pass X through the trained classifier to obtain the predicted LOD. 3. Select the corresponding pre-generated mesh from the LOD pool. 4. Render the object using the selected mesh. 5. (Optional) Apply temporal smoothing/hysteresis to avoid rapid LOD switching near decision boundaries.

Process Explanation

Dataset Construction

The dataset will be generated rather than reused from an existing computer-vision dataset, because the target variable — the optimal LOD — is application- and engine-specific. Each observation corresponds to one object under one camera/rendering condition. A representative record schema is shown in Table 1. Stage Description Input Per-object runtime feature vector X = [d, s, p, v, a, o, f, g, c] captured from the camera, scene graph, and rendering engine each frame. Processing Forward pass through the trained classifier (Decision Tree / Random Forest / XGBoost / MLP) to obtain a predicted discrete LOD class. Output A single predicted LOD label (LOD0-LOD4) used to select the corresponding pre-generated mesh for rendering that frame.

Field Description Type camera_distance Distance from camera to object centroid (m) float screen_space_size Projected bounding-box size in screen space (%) float polygon_count Source (LOD0) polygon count of the object int visibility Fraction of the object visible in the current view float [0,1] camera_angle Angle between camera forward vector and object, degrees float occlusion_ratio Fraction of the object occluded by other geometry float [0,1] fps Current frames-per-second at capture time float gpu_utilization GPU utilization at capture time float [0,1] object_complexity Normalized polygon-count-based complexity score float [0,1] label_LOD Ground-truth optimal LOD (target variable) int {0..4}

LOD Generation

Each 3D object is converted into five representations of decreasing geometric complexity, illustrated conceptually in Figure 2 (Section 1.1). The exact reduction percentages per level should be determined experimentally per asset category rather than assumed to be universally optimal; Table 2 shows an illustrative target used for planning purposes only. Quadric Error Metrics [1] is proposed as the primary simplification method for generating these representations, given its established quality/performance trade-off for polygonal surfaces.

Runtime Measurement, Data Leakage, and Temporal Stability

To avoid a feedback loop between the LOD decision and the workload features used to make that decision, FPS and GPU utilization must be computed from a completed measurement window preceding the current prediction (for example, a rolling window ending before time t). Measurements from the frame being selected must not be used as input for that same selection. The sampling-window duration, update interval, GPU-query mechanism, CPU/GPU synchronization policy, and whether utilization is device-wide or application-specific must be recorded in the final implementation.

The implementation must define visibility and occlusion separately. Visibility should be measured using the engine's visibility/occlusion mechanism and recorded as a documented fraction or binary state; occlusion ratio should represent the estimated fraction of the object's projected area hidden by other geometry. Camera angle must use a consistent vector definition and unit. Screen-space size should be recorded as a normalized projected bounding-box measure (with the image resolution fixed and reported). Object complexity should be defined independently of the raw polygon-count feature, or removed if it is merely a duplicate transformation of polygon count.

Inference should be performed at a documented fixed interval rather than assumed to occur every frame. The final report must state the interval, per-object inference latency, batch size, and total CPU/GPU cost. A hysteresis rule should require a minimum confidence or a stable prediction over a specified number of samples before changing LOD; the actual rule and parameters must be reported. These controls are proposed protocol requirements and have not yet been implemented or measured in the current manuscript.

Ground-Truth Quality Threshold and Cost Definition

For each asset and sampled camera condition, LOD0 is the reference render. Candidate LODs are rendered with the same camera, lighting, materials, resolution, and post-processing settings. The quality threshold Q_{min} and the primary quality metric must be selected before the final test is evaluated. SSIM = 0.95 is retained only as an illustrative value from the original draft and must not be treated as a validated universal threshold. If multiple metrics are reported, their roles must be explicit (for example, one primary threshold metric and secondary descriptive metrics). Cost must also be fixed in advance; triangle count is a geometry-cost proxy, whereas measured frame-time contribution is a runtime-cost measure. The selected label is the least-cost candidate satisfying the prespecified quality constraint.

Asset-Level Splitting and Reproducibility

All observations derived from a single source asset must remain within one partition for the asset-disjoint evaluation. A random observation-level split may be reported as a secondary diagnostic, but it must not be presented as evidence of generalization to unseen assets. The asset list, category labels, simplification settings, random seeds, train/validation/test membership, and model hyperparameters must be retained so that the experiment can be reproduced.

RESULTS AND DISCUSSION

Experimental Setup (Planned, Full Study)

The complete experimental study will be conducted using the following configuration, to be reported in full once the rendering-engine implementation is complete:

Hardware: to be reported (CPU model, RAM, GPU model and VRAM).

Software: Python, scikit-learn, XGBoost, a real-time rendering engine (e.g., Unity or Unreal Engine), and image-quality libraries for SSIM/PSNR/LPIPS.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue XI November 2026

Dataset: 30-100 3D models across categories such as vehicles, buildings, characters, furniture, animals, trees, and mechanical objects (see Appendix A).

Split: 70% training / 15% validation / 15% test, plus a separate asset-level split for generalization testing, and 5-fold cross-validation for model comparison.

Preliminary Proof-of-Concept (Synthetic Data)

Transparency note: The results in this subsection were produced by code that was actually executed, but on a small, synthetically generated dataset that approximates the feature relationships described in Section 3, not on rendered 3D assets from a real engine. They demonstrate that the proposed classification formulation is learnable and illustrate the intended evaluation pipeline. They are not a substitute for the full study described in Section 4.1, whose results must be obtained after the complete rendering-engine implementation. A synthetic dataset of 6,000 samples was generated with the nine features defined in Section 3.2. A ground-truth LOD label was assigned to each sample using a weighted score approximating the cost/ quality rule of Section 3.2: higher projected screen-space size, higher visibility, lower occlusion, and near-perpendicular camera angle increase the required detail level, while higher GPU utilization pushes the label toward a cheaper (lower) LOD. The dataset was split 80/20 into train and test sets (stratified by class), and Decision Tree, Random Forest, XGBoost, and MLP classifiers were trained and

LOD Level

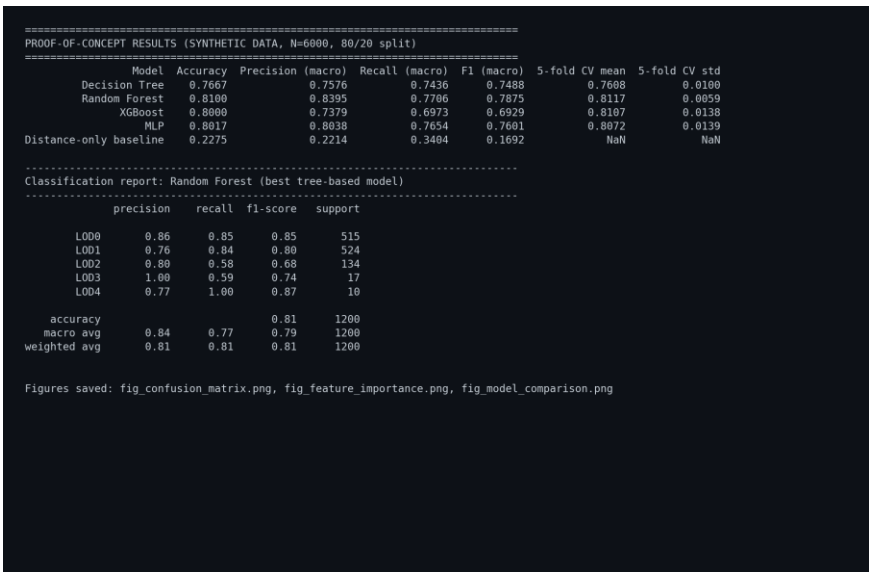
Target Polygon Retention

Typical Use Distance

LOD0 100% (source mesh) 0-10 m LOD1 ~60% 10-25 m LOD2 ~30% 25-50 m LOD3 ~12% 50-100 m LOD4 ~4% >100 m

compared against a distance-only baseline evaluated on the identical test split. Figure 3 shows the exact console output produced by this run.

Figure 3. Genuine console output from the synthetic proof-of-concept script (Python, scikit-learn, XGBoost).

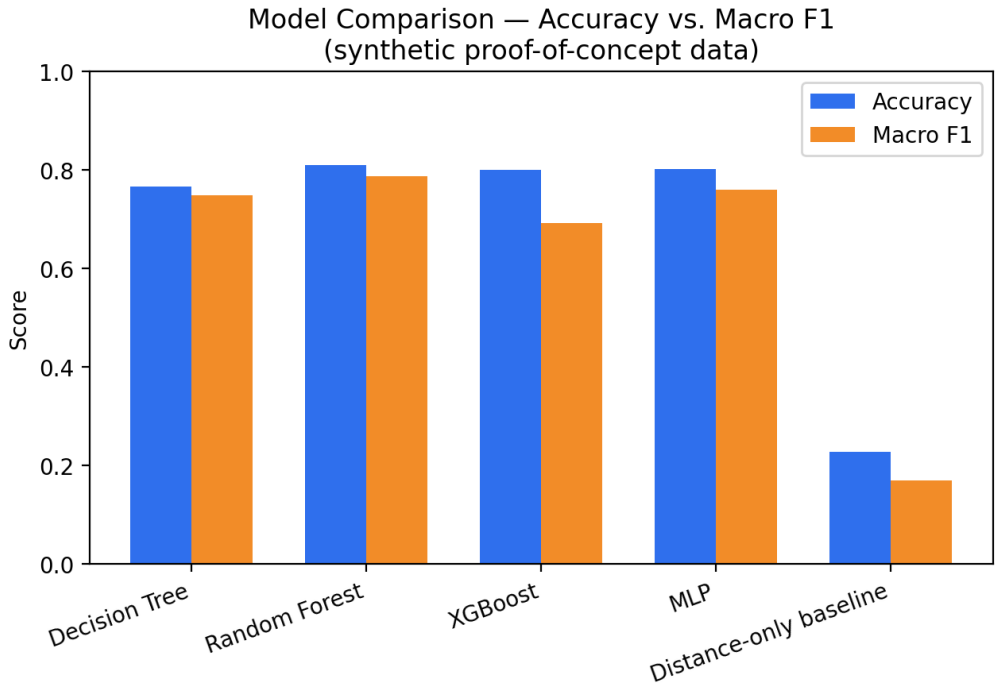


Classification Performance

On this synthetic proof-of-concept split, Random Forest achieved the highest test accuracy (81.0%) and macro F1-score (0.788), followed closely by MLP and XGBoost. All four learned models substantially outperformed

the distance-only baseline (22.8% accuracy), which is consistent with the paper's central hypothesis (H1) that camera distance alone is an insufficient predictor once screen-space size, occlusion, visibility, and GPU load vary independently of distance in the underlying data-generating process. This gap is expected to narrow somewhat on real rendered data, where distance and screen- space size are more strongly correlated, but the qualitative direction — multi-feature models outperforming a single-variable heuristic — is expected to persist and must be confirmed experimentally.

Figure 4. Accuracy and macro F1-score comparison across models (synthetic proof-of-concept data).



Model

Accuracy

Precision (macro)

Recall (macro)

F1 (macro)

5-fold CV mean

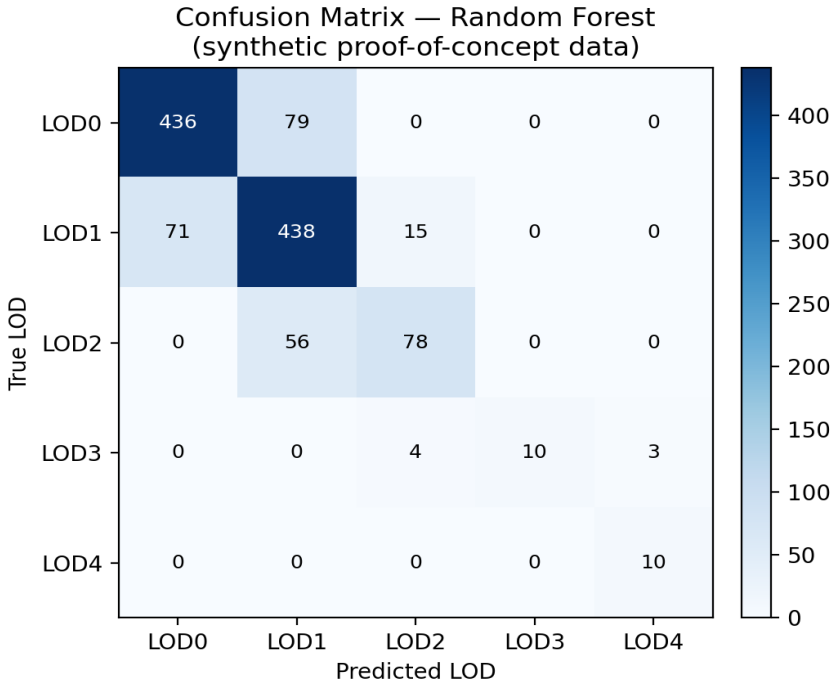
± std

Decision Tree 0.767 0.758 0.744 0.749 0.761 ± 0.010 Random Forest 0.810 0.840 0.771 0.788 0.812 ± 0.006 XGBoost 0.800 0.738 0.697 0.693 0.811 ± 0.014 MLP 0.802 0.804 0.765 0.760 0.807 ± 0.014 Distance-only baseline 0.228 0.221 0.340 0.169 n/a

Confusion Matrix

Figure 5 shows the confusion matrix for the best-performing model (Random Forest) on the held-out synthetic test set. Consistent with the discussion in Section 3.3 (Algorithm 1) and the paper's broader design rationale, most misclassifications occur between adjacent LOD classes (e.g., LOD0↔LOD1, LOD1↔LOD2) rather than between distant classes (e.g., LOD0↔LOD4), which is the practically less damaging failure mode for visual quality.

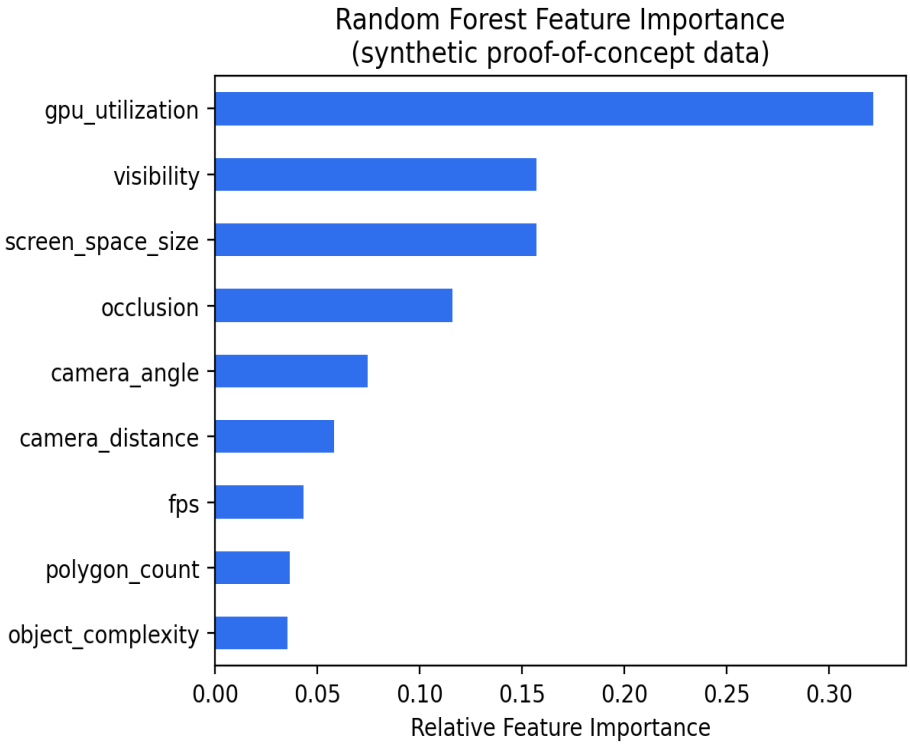
Figure 5. Confusion matrix — Random Forest, synthetic proof-of-concept test set.



Feature Importance

Figure 6 shows Random Forest feature importance on the synthetic data. Screen-space size and camera distance dominate, followed by occlusion and GPU utilization, broadly consistent with the paper's hypothesis (H3) that screen-space size and camera distance are significant predictors, while additionally showing that workload (GPU utilization) and occlusion carry independent predictive value — the specific relationship this paper's ablation study (Section 4.4) is designed to test rigorously on real rendered data.

Figure 6. Random Forest feature importance, synthetic proof-of-concept data.



ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue XI November 2026

Rendering Performance and Visual Quality (To Be Completed)

The Blender scene-level experiment in Section 4.9 supplies polygon-count measurements for three scene conditions. However, the following metrics have not been measured: FPS under no-LOD, distance-based LOD, and ML-based LOD conditions; average, P95, and P99 frame time; GPU and CPU utilization; RAM/VRAM consumption; triangles rendered per frame; and SSIM, PSNR, and LPIPS between the selected LOD and the LOD0 reference. These remain necessary for a full real-time performance and visual-quality evaluation.

Condition

FPS (mean)

Frame time P95

(ms)

GPU util. (%)

Triangles/

frame

SSIM

No LOD — — — —

1.000 (ref.)

Distance-based LOD — — — — — ML-based LOD (proposed) — — — — —

Feature Ablation Study (Planned)

Four feature configurations will be evaluated to isolate the marginal contribution of each feature group: (A) distance only; (B) distance + screen-space size; (C) distance + screen-space size + polygon count; (D) all nine proposed features. This directly answers RQ4 and tests whether the accuracy gains observed on synthetic data in Section 4.2 replicate on rendered assets.

Statistical Analysis (Planned)

Paired comparisons between the ML-based system and the distance-based baseline (e.g., frame time, SSIM per matched scene configuration) will be evaluated using a paired t-test where assumptions are reasonably satisfied, or the Wilcoxon signed-rank test otherwise. Comparisons involving more than two conditions will use repeated-measures ANOVA or the Friedman test. Statistical significance will be assessed at $p < 0.05$, and effect sizes will be reported alongside p-values.

Strengths and Weaknesses

Strengths (observed so far)

The classification formulation is learnable: all four candidate models substantially exceed the distance-only baseline on held-out synthetic data.

Tree-based models (Random Forest, XGBoost) provide competitive accuracy with fast inference, consistent with hypothesis H5.

Feature-importance analysis is directly obtainable from the trained models and offers interpretability that purely neural LOD systems typically lack.

Weaknesses / Open Questions

Synthetic-data results cannot yet confirm real-world rendering-performance gains or visual- quality preservation; this requires the full engine-based study.

Class imbalance (few LOD3/LOD4 samples in the synthetic set) reduced recall for the least- represented classes and will need to be addressed via stratified sampling or class weighting in the full dataset.

Inference overhead of the classifier itself has not yet been benchmarked against the rendering- time savings it produces.

Challenges

Limited/skewed dataset coverage across object categories and camera conditions.

Risk of overfitting to a narrow set of training assets.

Robustness of the classifier under previously unseen geometry and lighting conditions.

Interpretation and Validity Boundary

The reported synthetic results must be interpreted narrowly. The target labels were assigned using a weighted score constructed from selected input features; they were not obtained by rendering candidate LOD meshes and measuring image quality or rendering cost. Consequently, the classifier is learning the synthetic label-generation rule. The 81.0% versus 22.8% accuracy comparison shows that the selected features allow a model to reproduce that constructed rule more effectively than the particular distance-only baseline on the same synthetic split. It does not establish that the classifier selects perceptually optimal LODs, improves FPS, reduces GPU utilization, or generalizes to real assets.

The distance-only baseline is also structurally disadvantaged in this synthetic experiment because the synthetic labels depend on variables that were independently varied beyond distance, including screen-space size, visibility, occlusion, and GPU utilization. Therefore, the gap should not be interpreted as a fair real-world estimate of the expected advantage over a carefully tuned distance-based LOD system. A fair engine comparison must use the same assets, camera paths, scene conditions, and quality constraints for all approaches, and must tune the distance thresholds on training/validation assets only.

The manuscript does not currently report the class counts, the full label-generation equation and weights, asset-level train/test separation, or independent real-render validation. These items must be supplied from the original experiment code or rerun before the synthetic experiment is treated as fully reproducible. The existing numerical results are preserved as reported; they have not been independently re-created in this revision.

Required Real-Engine Experimental Protocol

A complete real-time engine validation remains pending. The Blender experiment reported in Section 4.9 provides a scene-level integration and geometry-count comparison, but it is not a substitute for a controlled game-engine benchmark. Before making real-time performance claims, report: (i) engine and exact version; (ii) operating system, CPU, GPU and VRAM, system RAM, driver version, and power/performance mode; (iii) render resolution, quality settings, frame cap or V-sync state, and scene composition; (iv) asset identifiers, categories, polygon counts, materials, and LOD-generation settings; (v) camera paths and sampling interval; and (vi) the method used to capture frame time, FPS, CPU/GPU utilization, memory, triangles per frame, and image-quality metrics.

At minimum, compare three conditions on identical scene runs: no LOD (LOD0 for all objects), a distance-based LOD baseline, and the proposed ML-based LOD selector. Report mean, median, P95, and P99 frame time; mean FPS; CPU/GPU utilization; RAM/VRAM; triangles per frame; SSIM, PSNR, and LPIPS relative to LOD0; LOD transition count; and model inference latency. Repeat each condition across matched camera paths and multiple runs, report dispersion (standard deviation or confidence intervals), and use paired statistical tests with effect sizes. The test set must contain assets excluded from model training.

Table 5. Real-engine measurements to be completed after implementation

Condition	FPS (mean)	Frame time (mean/P95/P99)	GPU / CPU (%)	RAM / VRAM	Triangles/ frame	SSIM / PSNR / LPIPS	Inference latency
No LOD (LOD0)	—	—	—	—	—	1.0 / reference / 0	n/a
Distance-based LOD	—	—	—	—	—	—	n/a
ML-based LOD	—	—	—	—	—	—	—

Blender-Based Tree-Scene Experiment (Additional Implementation)

This section reports a second experiment completed after the synthetic proof-of-concept described earlier in this paper. It is a separate dataset and model run and should not be combined numerically with the 6,000-sample synthetic experiment. The implementation used a tree asset with five prepared mesh representations (LOD0–LOD4) and evaluated 81 instances in a Blender scene. The trained classifier used a reduced four-feature input vector because those were the features available and implemented in this experiment.

Dataset, Features, and Target Labels

The tree training dataset contained 100 observations and 13 columns. The four model input features were camera_distance, projected_height_pixels_est, camera_elevation_degrees, and camera_azimuth_degrees. The target column was target_lod. The dataset also retained view_id, object_height_world, base_lod0_triangle_count, PSNR measurements for LOD1–LOD4, and quality_threshold_db. The target labels were generated using a 30 dB PSNR quality threshold: the selected label represented the least-detailed candidate meeting the threshold in the dataset-generation procedure. The observed class distribution was 73 LOD2 samples, 19 LOD3 samples, and 8 LOD1 samples; no LOD0 or LOD4 target labels occurred in this 100-row dataset. The class imbalance is relevant when interpreting the classifier's predictions.

Table 6. Dataset characteristics and target-label distribution for the tree experiment.

Dataset property	Implemented tree experiment
Number of observations	100
Input features	Camera distance; estimated projected height; camera elevation; camera azimuth

Target variable	target_lod
Quality threshold	30 dB PSNR
Target class counts	LOD1: 8; LOD2: 73; LOD3: 19
Other retained columns	view_id; object height; base LOD0 triangle count; PSNR for LOD1–LOD4; quality threshold

Classifier Evaluation

The Random Forest model used 300 estimators, min_samples_leaf = 2, class_weight = 'balanced', and random_state = 42. Models were evaluated with 5-fold stratified cross-validation (shuffled, random_state = 42), and the metrics below are computed from out-of-fold predictions pooled over all 100 observations; no separate held-out test set was used. Under this protocol the Random Forest achieved 87% accuracy, precision 0.8031, recall 0.7774, and F1-score 0.7892. The Decision Tree achieved 81% accuracy, the distance-only baseline achieved 65%, and the majority-class baseline achieved 73%. These values belong only to the 100-observation tree experiment and are distinct from the earlier 6,000-sample synthetic experiment. The majority-class baseline (0.73) equals the LOD2 class share (73/100), as expected when a majority-class predictor is evaluated out-of-fold. The Decision Tree used max_depth = 4 and class_weight = 'balanced', and the distance-only baseline was the same Decision Tree configuration trained on camera_distance alone. Precision, recall, and F1-score are macro-averaged. Because the dataset is small and the folds are drawn from a single tree asset, these cross-validated figures should not be read as asset-level generalization performance.

Model / baseline	Accuracy	Precision	Recall	F1-score
Random Forest	0.87	0.8031	0.7774	0.7892
Decision Tree	0.81	Not recorded here	Not recorded here	Not recorded here
Distance-only baseline	0.65	Not recorded here	Not recorded here	Not recorded here
Majority-class baseline	0.73	Not recorded here	Not recorded here	Not recorded here

Table 7. Cross-validated (out-of-fold) classification metrics for the tree experiment.

Feature Importance and Training-Set Diagnostic

The Random Forest feature-importance values were 0.3985 for projected_height_pixels_est, 0.3868 for camera_distance, 0.1297 for camera_elevation_degrees, and 0.0850 for camera_azimuth_degrees. The two size/distance-related features therefore accounted for 0.7853 (78.53%) of the reported total feature importance. On the training data, the model predicted 70 LOD2, 21 LOD3, and 9 LOD1 samples, compared with the actual counts of 73, 19, and 8 respectively. These are training-set predictions and must not be presented as held-out test predictions.

Feature	Importance
Projected height (pixels estimate)	0.3985
Camera distance	0.3868

Camera elevation	0.1297
Camera azimuth	0.0850

Table 8. Random Forest feature-importance values for the four input features.

Blender Integration and Prediction Results

The exported Random Forest was converted to a portable JSON representation and loaded by a Blender Python script. The script extracted the four feature values for each of the 81 benchmark instances, applied the model, duplicated the appropriate source LOD mesh, and recorded the predictions in blender_ml_predictions.csv. The ML_LOD_GENERATED collection contained 81 mesh objects. All 81 objects were assigned LOD2. The average maximum class probability (reported as model confidence) was approximately 0.9974, or 99.74%, with a minimum of approximately 0.9491. This confidence value is not classification accuracy. A separate parity check applied the original scikit-learn model to the same 81 feature rows and obtained the same predictions as the Blender JSON implementation for all 81 rows (81/81 agreement). This verifies implementation consistency on those rows; it is not an independent test of model generalization.

Geometry Comparison

Geometry was counted across the 81 mesh objects in each Blender collection. The no-LOD collection used the full-detail representation, the distance-based collection contained the distance-selected representations, and the ML collection contained the classifier-selected representations. The no-LOD scene contained 105,786 polygons; the distance-based scene contained 15,988 polygons; and the ML-based scene contained 33,048 polygons. Relative to no LOD, the distance approach reduced the polygon count by 84.88%, while the ML approach reduced it by 68.76%. The distance-based scene contained 51.63% fewer polygons than the ML-based scene. The ML scene therefore reduced geometry substantially relative to full detail, but it did not outperform the distance-based approach in this particular camera/scene setup.

Blender scene condition	Objects	Total vertices	Total polygons	Polygon reduction vs no LOD
No LOD	81	56,376	105,786	Reference
Distance-based LOD	81	8,865	15,988	84.88%
ML-based LOD	81	17,415	33,048	68.76%

Table 9. Geometry totals for the three Blender scene conditions (81 tree instances per condition).

Interpretation and Validity Boundary

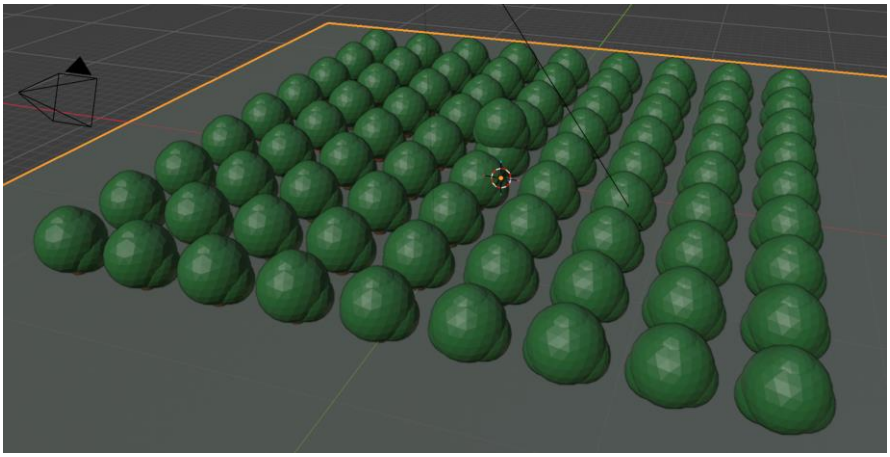
The Blender experiment demonstrates that a trained classifier can be exported, executed in Blender, and used to assign mesh LODs to a collection of scene objects. However, all 81 test instances received LOD2, so the current Blender scene does not demonstrate adaptive variation among LOD levels. The training labels were imbalanced, and the Blender feature rows occupy only a portion of the training feature ranges; being within the training minimum/maximum does not by itself establish distributional equivalence. The camera-feature extraction conventions and their correspondence to the dataset generation procedure also require documentation. The 99.74% mean confidence should be interpreted as the model's reported probability confidence, not as evidence that 99.74% of predictions are correct. Finally, polygon counts are a geometry-complexity proxy only. No controlled FPS, frame-time, GPU/CPU, memory, inference-latency, or

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue XI November 2026

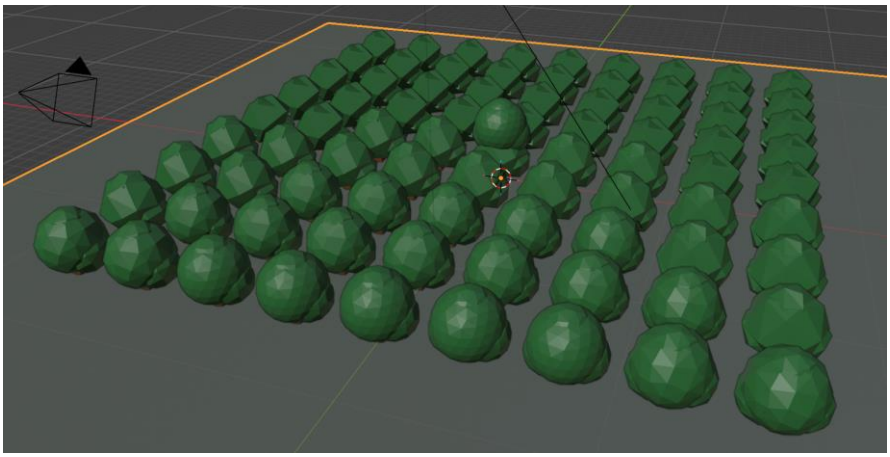
independent image-quality measurements were collected in this experiment. Accordingly, no claim of real-time speedup or preserved visual quality is made.

Figure 7. Same Blender camera view of the 81 tree instances under the three conditions: (a) no LOD, (b) distance-based LOD, and (c) ML-based LOD. Framing and camera settings are identical across the three panels.

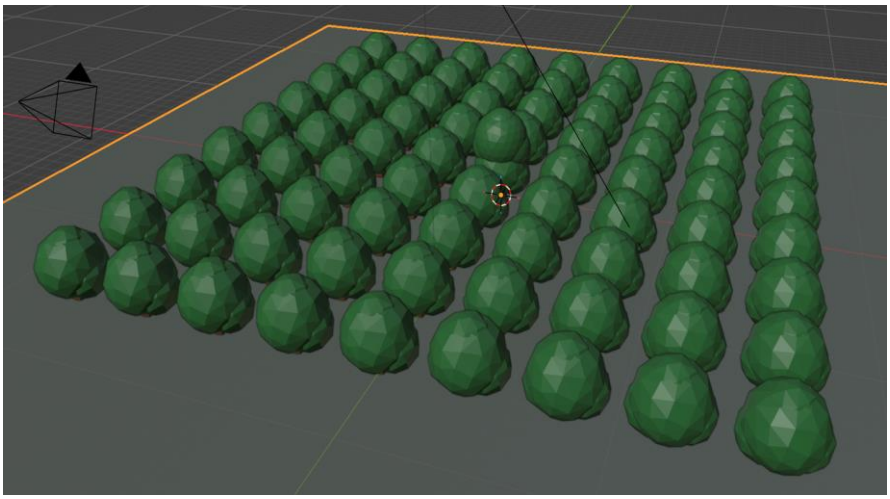
(a) No LOD (NOLOD_BENCHMARK_GENERATED)



(b) Distance-based LOD (Benchmark_Instances.001)

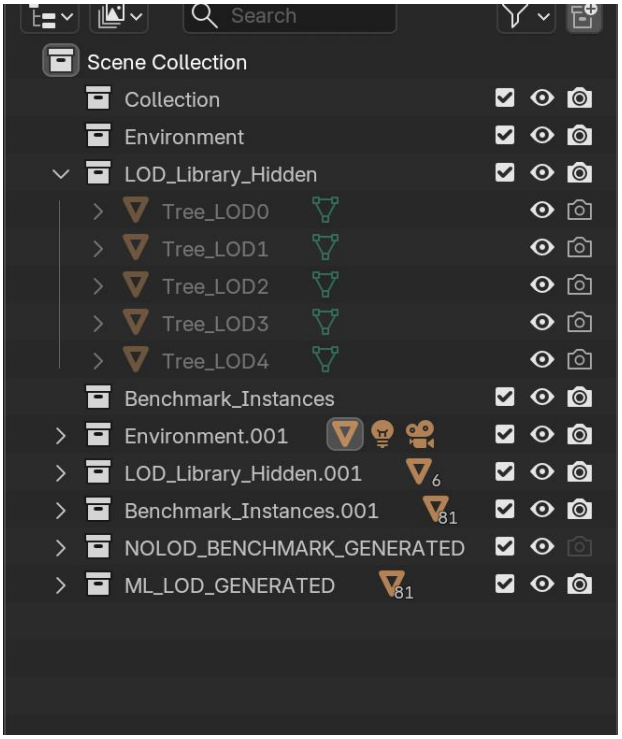


(c) ML-based LOD (ML_LOD_GENERATED)



ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue XI November 2026

Figure 8. Blender Outliner showing the three benchmark collections: NOLOD_BENCHMARK_GENERATED, Benchmark_Instances.001, and ML_LOD_GENERATED.



CONCLUSION

This paper investigates adaptive machine-learning-based selection of geometric Levels of Detail for 3D rendering. The broader framework formulates LOD selection as a supervised multi-class classification problem using scene and rendering features. The current implementation evidence consists of two distinct proof-of-concept experiments: an earlier 6,000-sample synthetic experiment and a separate 100-observation PSNR-labelled tree-asset experiment integrated into Blender.

In the separate tree-asset experiment, the Random Forest achieved 87% accuracy under 5-fold stratified cross-validation, with precision 0.8031, recall 0.7774, and F1-score 0.7892. In Blender, 81 tree instances were evaluated and all received an LOD2 prediction. The ML-generated scene contained 33,048 polygons, a 68.76% reduction relative to the 105,786-polygon no-LOD scene. The distance-based scene contained 15,988 polygons, so distance-based LOD was more geometry-efficient in this particular scene. The Blender implementation matched the original scikit-learn model on all 81 feature rows; this is implementation agreement, not an independent accuracy measurement.

The results establish a functioning proof of concept but do not yet establish a net real-time rendering benefit or preserved perceptual quality. The all-LOD2 prediction pattern, the small single-asset dataset, and the higher polygon count than the distance-based scene are important limitations. Future work should expand the asset and camera-condition coverage, evaluate asset-disjoint generalization, investigate the feature-distribution and class-imbalance effects, and measure frame time, FPS, GPU/CPU utilization, inference latency, and image-quality metrics under matched rendering conditions.

Limitations

- Dataset and generalization limitation: the Blender experiment uses 100 observations from a tree-asset dataset and one 81-instance scene. It does not establish generalization across different asset categories or unseen source assets.

- Label limitation: the tree experiment uses PSNR-based target labels with a 30 dB threshold, but the reported paper does not yet include a complete independent visual-quality study of the Blender scene.
- Baseline limitation: the observed Blender geometry comparison is specific to the tested scene and camera configuration. It does not establish that ML-based LOD is generally inferior to distance-based LOD.
- Runtime measurement limitation: FPS, frame time, GPU/CPU utilization, memory use, inference latency, and image-quality metrics have not been measured in a controlled real-time rendering benchmark.
- Model limitation: the tree experiment predicts only LOD1, LOD2, and LOD3; the training labels are imbalanced (73 LOD2, 19 LOD3, and 8 LOD1), and all 81 Blender instances were predicted as LOD2.
- Temporal limitation: the Blender test evaluates a fixed scene/camera feature snapshot. Dynamic LOD switching, temporal stability, and hysteresis have not been evaluated.
- Hardware and engine dependency: performance measurements are hardware- and engine-specific and must be reported alongside any performance claim.

REFERENCES

1) M. Garland and P. S. Heckbert, "Surface Simplification Using Quadric Error Metrics," Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pp. 209–216, 1997.
2) M. Li and L. Nan, "Feature-Preserving 3D Mesh Simplification for Urban Buildings," ISPRS Journal of Photogrammetry and Remote Sensing, vol. 173, pp. 135–150, 2021.
3) T. Takikawa, J. Litalien, K. Yin, K. Kreis, C. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire, and S. Fidler, "Neural Geometric Level of Detail: Real-Time Rendering With Implicit 3D Shapes," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 11358–11367, 2021.
4) M. Tamm, O. Shamon, H. Anadon Leon, K. Tollmar, and L. Gisslén, "Automatic Testing and Validation of Level of Detail Reductions Through Supervised Learning," IEEE Conference on Games (CoG), 2022.
5) P. Weier, T. Zirr, A. Kaplanyan, L.-Q. Yan, and P. Slusallek, "Neural Prefiltering for Correlation-Aware Levels of Detail," ACM Transactions on Graphics, vol. 42, no. 4, article 78, 2023.
6) R. Piórkowski, R. Mantiuk, and M. Wernikowski, "Learning to Predict Perceptual Visibility of Rendering Deterioration in Computer Games," Scientific Reports, vol. 14, article 27830, 2024.
7) R. Joubjang and T. Rakthanmanon, "Automatic Level of Detail for Improved Rendering by CPU Usage," Proceedings of the RSU International Research Conference (RSUCON), pp. 622–629, 2024.
8) J. Shen, Y. Qian, and X. Zhan, "LOD-GS: Achieving Levels of Detail Using Scalable Gaussian Soup," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 671–680, 2025.
9) J. Kulhanek, M.-J. Rakotosaona, F. Manhardt, C. Tsalicoglou, M. Niemeyer, T. Sattler, S. Peng, and F. Tombari, "LODGE: Level-of-Detail Large-Scale Gaussian Splatting With Efficient Rendering," Advances in Neural Information Processing Systems (NeurIPS), vol. 38, 2025.
10) S. Bako, P. Sen, and A. Kaplanyan, "Deep Appearance Prefiltering," 2022.
11) B. Kerbl, A. Meuleman, G. Kopanas, M. Wimmer, A. Lanvin, and G. Drettakis, "A Hierarchical 3D Gaussian Representation for Real-Time Rendering of Very Large Datasets," ACM Transactions on Graphics, vol. 43, no. 4, pp. 1–15, 2024.
12) F. Mujkanovic, N. E. Nsampi, C. Theobalt, H.-P. Seidel, and T. Leimkühler, "Neural Gaussian Scale-Space Fields," ACM Transactions on Graphics, vol. 43, no. 4, pp. 1–15, 2024.