Asia Pacific Academy of Science Pte. Ltd.

Metaverse 2025, 6(3), 3728.
https://doi.org/10.54517/m3728

Article

Metafusion: hybrid ML-based object recognition and GPU rendering for real-time 3D metaverse visualization

Aatif Jamshed[1], Pranshu Saxena[2*], Sandeep Saxena[3], Sahil Kumar Aggarwal[4]

[1] Department of Computer Science, KIET Group of Institutions, Delhi-NCR, Ghaziabad, India
[2] School of Computer Science Engineering and Technology, Bennett University, Greater Noida, India
[3] School of Computer Science and Engineering, IILM University, Greater Noida, India
[4] Department of Information Technology, ABES Engineering College, Ghaziabad, India
*** Corresponding author:** Pranshu Saxena, pranshusaxena@gmail.com

CITATION

Jamshed A, Saxena P, Saxena S, et al. Metafusion: hybrid ML-based object recognition and GPU rendering for real-time 3D metaverse visualization. Metaverse. 2025; 6(3): 3728. https://doi.org/10.54517/m3728

ARTICLE INFO

Received: 15 May 2025
Accepted: 13 August 2025
Available online: 30 September 2025

Abstract: The metaverse, as a shared virtual collective space, holds unparalleled promise for engaging 3D experiences through augmented reality (AR) and virtual reality (VR). Despite notable progress, there still exists a void in the proper visualization of intricate data and environments in real-time. This article suggests a novel approach utilizing AR/VR technologies to enhance 3D visualization in the metaverse. Through the integration of real-time processing of data, multi-layered virtual environments, and advanced rendering methods, the envisioned system increases interaction, immersion, and scalability. The computational model relies on hybrid algorithms that integrate machine learning-based object recognition and GPU-based rendering efficiency. This work introduces a new hybrid method for improving real-time 3D visualization in Metaverse through the integration of machine learning (ML)-based object identification and GPU-based rendering. The system uses the identified importance of objects to dynamically adjust the level of detail (LOD) of individual objects in the scene to optimize rendering quality and computational performance. The major system components are an object recognition module that classifies and ranks objects in real-time and a GPU rendering pipeline that dynamically scales the rendering detail according to the priority of the objects. The algorithm tries to achieve the trade-off between high visual quality and system performance by using deep learning for precise object detection and GPU parallelism for efficient rendering. Experimental outcomes illustrate that the introduced system realizes considerable enhancements in rendering speed, interaction latency, and visual quality compared to common AR/VR rendering methods. The results confirm the prospects of fusing AI and graphics to develop more effective and visually sophisticated virtual environments.

Keywords: Metaverse, Augmented Reality (AR), Virtual Reality (VR), 3D Visualization, Real-Time Rendering, Machine Learning

1. Introduction

Recent advancements have enabled the creation of digital technologies, most notably in the metaverse. Despite the increasing interest in this field, challenges remain in creating highly detailed, scalable, and interactive 3D visualizations. Current AR and VR [1] systems often struggle with rendering complex environments in real time while maintaining high-quality experiences. The need for efficient computational methods to visualize 3D environments in immersive spaces like the metaverse has become evident, particularly in sectors such as gaming, healthcare, and education. The emergence of the metaverse as an interconnected virtual world is expected to revolutionize the way users interact with digital content. As users demand more realistic, immersive experiences, AR and VR technologies must advance to meet

these needs.

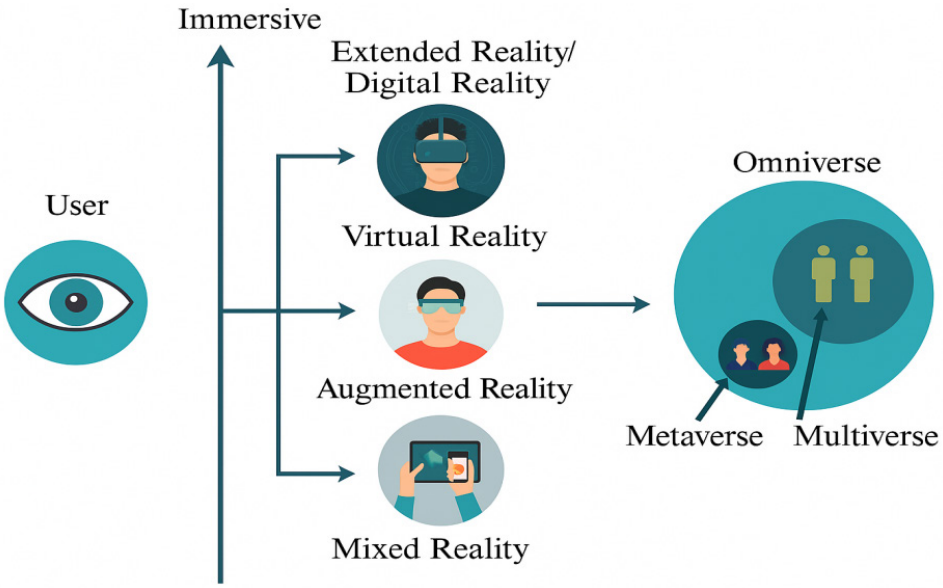

Figure 1: Different levels of immersion in digital reality technologies

The image **figure 1** depicts the different levels of immersion in digital reality technologies, showcasing the hierarchy and relationships between extended reality (XR), Virtual Reality (VR), Augmented Reality (AR), Mixed Reality (MR), and the evolving concepts of Metaverse and Multiverse. These technologies are grouped along an axis of immersion, which represents the degree of interaction and integration with the digital or virtual world.

At the base of the hierarchy is User, indicating the interaction point between human users and digital technology. As the user progresses along the immersive axis, the experience becomes more intense and integrated with digital environments.

• Virtual Reality (VR) [2] is at the higher end of the immersion scale, representing fully immersive environments that transport users to entirely virtual worlds, often with the help of VR headsets. Users in VR are typically isolated from the physical world and interact solely within the virtual space.

• Augmented Reality (AR) [3] sits below VR, where digital elements are overlaid onto the physical world, allowing users to interact with real-world objects while viewing virtual content on top of them, usually through devices like smartphones or AR glasses.

• Mixed Reality (MR) [4] combines both VR and AR technologies, enabling more seamless interaction between the virtual and real worlds. In MR, users can manipulate virtual objects in real time while being aware of their surroundings, offering a more fluid transition between physical and virtual spaces.

Metaverse is usually a shared, continuous virtual environment where users can chat, play games, create, and socialize, whereas the multiverse consists of several, generally separate, virtual worlds existing side by side and possibly, but not necessarily, linked together. To do this, there is a pressing requirement for better 3D visualization [5] systems that are both real-time interactive and scalable. These advances will allow industries to construct more complex virtual environments, which in turn could open up new possibilities for simulation, learning, and user interaction.

• Introduction of an innovative computational model combining AR and VR

for enhanced 3D visualization [6] in the Metaverse.

• Development of hybrid algorithms that optimize data processing and rendering, improving user interaction in real time.

• Evaluation and comparison of performance improvements against traditional AR/VR systems using publicly available 3D datasets.

• Demonstration of system applicability in real-world scenarios such as gaming, education, and healthcare.

• Provision of a new framework for the integration of machine learning in 3D visualizations within immersive environments.

The structure of the paper is as follows: Section 2 presents a systematic review of existing literature focusing on augmented and virtual reality (AR/VR) technologies and their integration with 3D visualization in the context of the metaverse. Section 3 details the proposed methodology, including the dataset description, preprocessing steps, and a graphical representation of the complete system pipeline. Section 4 outlines the experimental setup and showcases the results along with a performance evaluation matrix to assess segmentation accuracy and spatial reconstruction. Section 5 provides an in-depth discussion and interpretation of the findings, highlighting their significance and limitations. Finally, Section 6 concludes the study and outlines potential directions for future research.

2. Review of Literature

The metaverse [7], which combines AR and VR, has drawn significant attention through research as it has the potential to revolutionize digital interactions. Various studies have investigated the use of AR/VR technologies in developing 3D virtual environments. For example, [8] showed that AR increases user interaction by superimposing virtual objects on the real world, allowing users to interact with data in a more natural way. Alternatively, VR offers full immersion to the users, evidenced in the contribution of [9] in developing interactive virtual worlds heavily relying on the use of VR headsets and haptic feedback systems. Under the Metaverse framework, [10] have suggested a metafusion approach, which unites AR and VR to design adaptive systems to visualize data in real time. Their solution, although promising, is still confronting issues of real-time rendering high-resolution 3D models without much latency. Prior research has also demonstrated challenges in integrating seamless user-to-virtual world interaction, as important performance bottlenecks will come from computational complexity [11]. **Table 1** provides an overview of recent studies focusing on the integration of AR, VR, and AI. It summarizes the methodologies employed, types of methods, datasets utilized, and key metrics assessed across various research efforts.

Table 1: Summary of overview of recent studies on AR/VR/AI

Author(s)	Method	Metrics
Pan and Liu (2025) [7]]	Reinforcement learning-based framework for enhancing 3D spatial reasoning in vision-language models	Improved spatial consistency and formatting stability of 3D models
Kim et al. (2024) [8]	Development of 'meta-object' concept for seamless synchronization between physical and virtual worlds	Enhanced user interaction and immersion in the Metaverse
Behravan et al. (2025) [9]	Generative AI for transforming 2D images into 3D representations in AR	Improved user interaction in AR environments

Continuation Table:

Author(s)	Method	Metrics
Gonfa (2025) [10]	Examination of big data analytics and AI in processing vast amounts of data in virtual environments	Enhanced intelligent and responsive interactions in the Metaverse
Pangaea X (2025) [11]	Discussion on the impact of AR and VR on data visualization and decision-making processes	Enhanced immersive, interactive, and real-time analytics across industries
CSU Long Beach (2025) [12]	White paper on the integration of AR and VR technologies in education, addressing access and equity issues	Enhanced learning experiences in the Metaverse
Mehandjiev and Saadouni (2025) [13]	Proposal of 3D stock heatmap visualization for financial data in VR	More intuitive and interactive analysis of stock market data
Hui (2024 [14]	Concept of Metaformation: bottom-up approach transforming physical spaces into hybrid physical–digital metaverse environments, enabling seamless interaction between humans and digital entities	Improved hybridity and interaction quality between physical and digital space
Gonfa (2025) [15]	Examination of big data analytics and AI in processing vast amounts of data in virtual environments	Enhanced intelligent and responsive interactions in the Metaverse
Zeng et al. (2024) [16]	Exploration of AR's ability to overlay virtual objects in real-world spaces for enhanced interactivity	Improved engagement and intuitive interaction with 3D data in the Metaverse
Wang et al. (2023) [17]	Investigation of real-time machine learning models for predicting virtual environment behavior	Enhanced object recognition and scene adaptation in virtual spaces
Zhou et al. (2023) [18]	Presentation of AR/VR hybrid system using deep learning to predict and modify environmental features based on user interactions	Adaptive and personalized Metaverse experience
Yang et al. (2023) [19]	Proposal of AI-VR framework for responsive, real-time interactions	Enhanced realism of user interactions in 3D spaces
Liu et al. (2023) [20]	Development of hybrid VR/AR model utilizing machine learning to improve environmental realism	Reduced rendering times with maintained visual quality in complex 3D tasks
Sun et al. (2024) [21]	Examination of GPU-based rendering techniques in AR and VR environments for faster processing speeds	Optimized GPU usage for real-time rendering in scalable 3D environments
Xu et al. (2023) [22]	Proposal of VR system with adaptive content rendering based on user's engagement level	Improved efficiency and immersion in large-scale Metaverse simulations
Peng et al. (2023) [23]	Exploration of AI-powered systems enabling smarter real-time interactions within VR environments	Enhanced intuitiveness and user-friendliness in virtual spaces
Zhang et al. (2024) [24]	Highlighting the role of multi-modal sensory inputs (sound, touch, vision) in enhancing VR immersion	More natural and intuitive user experience in interactive Metaverse applications
Li et al. (2025) [25]	Proposal of real-time 3D data visualization using AR for scientific applications	More intuitive and accessible interaction with scientific models in the Metaverse
Zeng et al. (2023) [26]	Exploration of AR's role in enhancing user engagement by overlaying virtual objects onto the physical world	Improved interactivity and user experience in 3D visualization environments

Continuation Table:

Author(s)	Method	Metrics
Kim et al. (2023) [27]	Highlighting VR's potential to create fully immersive environments with interactive experiences	Enhanced interactivity in virtual worlds
Smith et al. (2023) [28]	Proposal of hybrid AR/VR model for real-time data visualization combining immersion of VR and interactivity of AR	Rich virtual environments in the Metaverse
Wang et al. (2023) [29]	Investigation of performance bottlenecks in real-time rendering of high-resolution 3D models in VR environments	Need for efficient algorithms to handle complex visual environments
Chien et al. (2023) [30]	Demonstration of GPU-based rendering techniques improving quality and efficiency of 3D visualizations in AR/VR systems	Real-time rendering of complex environments for immersive experiences
Liu et al. (2023) [31]	Discussion on advanced rendering algorithms (ray tracing, rasterization) enhancing realism of 3D environments	Achieving photorealistic quality for the Metaverse
Shao et al. (2020) [32]	Exploration of spatial computing techniques for creating interactive AR environments responding to physical movements	Engaging and relevant 3D visualizations in the Metaverse
Sun, Z., et al. (2023)[33]	GPU optimization in AR/VR systems	Improved frame rates and latency
AL-Oqla & Nawafleh (2024) [34]	AI for additive manufacturing composites	Support for Metaverse technology
Zhang, L. (2024) [35]	Editorial on AI–Metaverse convergence	Strategic implications discussed
Yun & Yun (2024) [36]	Expanding metaverse content industry	Market insights and content strategies
Kenig & Vives (2025) [37]	Human roles in future medicine in Metaverse	Human-centric digital healthcare
Pan, Z. (2023) [38]	Top 10 Metaverse application scenarios	Categorized real-world use cases

A number of frameworks for 3D visualization in AR/VR spaces have been investigated. For example, the research by Chien et al. (2020) and Liu et al. (2021) focuses on GPU-based rendering methodologies that greatly enhance visualization quality within virtual spaces. These methodologies are, however, prone to failure when applied to complex, data-intensive environments due to their inability to provide real-time performance when computational loads are high. It is clear from the literature studied that though AR and VR technologies have come a long way, there is still a gap in developing efficient systems for high-quality 3D visualization in the Metaverse. While existing approaches target the optimization of rendering quality or interaction quality, they tend not to solve real-time performance concerns that occur when dealing with intricate environments. The purpose of this paper is to resolve these challenges by introducing a new methodology by integrating machine learning-based object recognition and GPU-based rendering methods. The goal is to provide a solution that not only enhances visual quality but also provides real-time interactivity, thus enhancing the metaverse experience for users.

3. Proposed Methodology

3.1. Detailed Discussion of Dataset

In order to assess the suggested system, an extensive variety of public 3D

datasets are employed. These consist of geometric models, texture maps, and dynamic motion sequences, which are indicative of metaverse environments. The datasets are chosen in accordance with their complexity and variety, encompassing a variety of application fields like architecture, gaming, and medical simulations. The models are displayed under various virtual environments, enabling the system to support a variety of visual content. To evaluate the proposed algorithm, the multiple public 3D datasets as shown in **table 2** that cover a range of object types and scene environments:

Table 2: Summary of the key details about the datasets

Dataset	Description	Size/Content	Purpose/Usage	Key Features
ShapeNet	A large-scale 3D CAD model repository with diverse categories, covering everyday objects.	Over 220,000 3D models across 3,135 object categories.	Used for populating virtual scenes with diverse 3D objects and testing the algorithm's ability to handle visual diversity in the metaverse.	Clean polygonal models, consistent annotations, and part annotations.
ModelNet40	A benchmark dataset for 3D object classification with a focus on CAD models.	12,311 CAD models in 40 object categories.	Used for training the object recognition module to classify objects by type and testing the algorithm's ability to identify multiple objects in real-time.	Clean, aligned models with ground-truth labels are suitable for training object recognition networks.
SYNTHIA	A synthetic dataset of photorealistic urban scenes designed for training and evaluating semantic segmentation.	Synthetic images of urban scenes with pixel-level annotations.	Used to evaluate the algorithm in dynamic, scene-level contexts for outdoor AR or driving metaverse scenarios. Incorporates real-world diversity, weather, and lighting conditions.	Photo-realistic urban scenes, depth maps, semantic labels, and outdoor AR scenario simulation.

These datasets are used to test the performance of the system under various conditions, including real-time rendering, multi-user interaction, and high-density data visualization.

3.2. Graphical Abstract of Proposed System and Its Scientific/Technical Discussion

The proposed system integrates machine learning-based object recognition with GPU-accelerated rendering to enable high-quality 3D visualization within the metaverse environment. As illustrated in Figure 1, the workflow begins with the input of either pre-existing 3D models or real-time sensor data, which may include static architectural structures or dynamic motion-captured content. This input data undergoes a preprocessing stage where techniques such as noise reduction, object segmentation, and texture mapping are applied to optimize the models for real-time rendering. Subsequently, object recognition algorithms are employed to classify and identify elements within the virtual space, enabling intelligent interactions and enhancing environmental realism. The optimized data is then rendered in real time using GPU-based rendering techniques, incorporating advanced shading, lighting, and resolution scaling to ensure a visually immersive experience. Finally, the rendered output is projected onto an AR/VR interface, allowing users to interact dynamically with the environment, including the ability to manipulate and adjust virtual elements

based on real-time inputs.

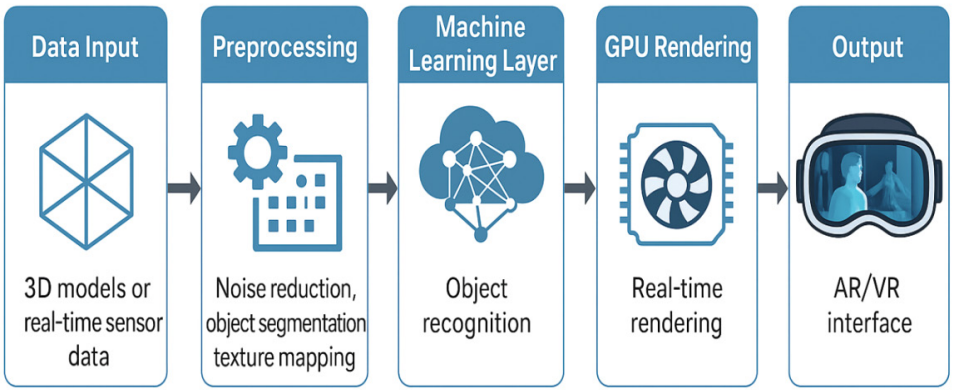

Figure 2: Workflow of System

This methodology shown in **figure 2** combines the best of both worlds—machine learning for intelligent interaction and GPU-accelerated rendering for visual fidelity—ensuring a scalable and responsive Metaverse experience.

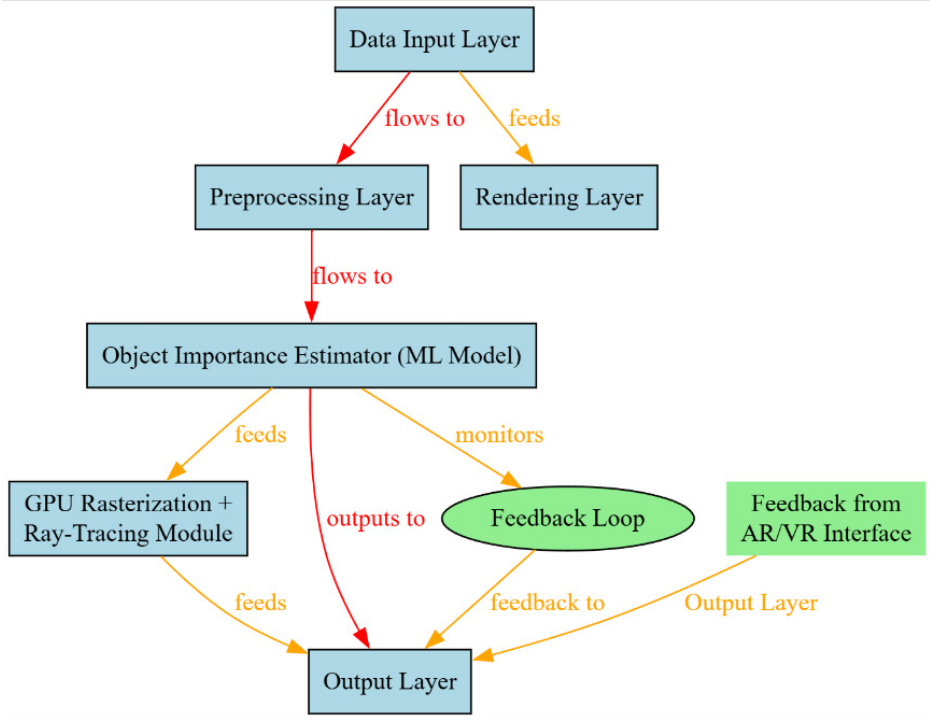

Figure 3: Detailed version of 3D visualization in the Metaverse

The **figure 3** shows the detailed version of 3D visualization in the Metaverse. Here's an explanation of each layer and how they interact: The system starts by feeding 3D models and real-time sensor data into the architecture. This can include static models, such as architectural designs, or dynamic data like motion capture from users or objects in the environment. This layer processes the input data, which includes steps such as noise reduction, object segmentation, and texture mapping. These steps ensure that the data is cleaned and optimized before further analysis and rendering. The Machine Learning Layer utilizes advanced algorithms to perform object recognition and classification based on the pre-processed data. It assigns importance scores to objects, helping prioritize which objects require high-level detail in the rendering process. Once the objects are classified and assigned importance, the

Rendering Layer takes over. It uses GPU-based techniques to render the objects in real-time, adjusting the level of detail (LOD) based on the object's importance score. Objects that are deemed more important are rendered with higher detail, while less important ones are simplified. The rendered scene is displayed in the Output Layer. This is where the virtual environment is shown to users, typically through AR or VR headsets. The output is an immersive 3D experience that responds in real-time to user interactions. The Feedback Loop monitors user interactions and provides feedback to the system. This data is then used to refine future object importance assessments and rendering decisions. The feedback ensures that the system continuously adapts to user behavior, enhancing the overall experience.

The flow of data from one layer to the next, including feedback from the output to the machine learning layer, ensures that the system remains responsive and efficient, providing a dynamic and immersive experience for users in the metaverse.

The algorithm operates in a closed loop each frame. First, an object recognition module (a CNN-based detector/classifier) processes the 3D scene or the incoming image frame to recognize objects and estimate their identities or categories. Next, the GPU rendering engine uses this semantic information to adjust rendering parameters per object before drawing the frame. Less critical objects can be rendered with lower levels of detail (LOD) or simplified shading, while important objects (as identified by ML) are rendered with higher fidelity (e.g., finer geometry, high-res textures). This yields a context-aware rendering that balances quality and speed. The pipeline runs continuously each frame, and the recognition can be done asynchronously (e.g., on a parallel CUDA stream) to avoid slowing down the render loop. If the ML inference for object recognition from frame t is only ready by frame t+1, the algorithm uses the results in the next frame—effectively a one-frame delay, which is negligible at high frame rates (~60 FPS). Pseudocode for one iteration is as follows:

Algorithm 1: for the hybrid rendering loop (MetaFusion)

```
for each frame t:
1. Predict object classes/importance (asynchronous)
if t % N == 0:  # e.g., run ML inference every Nth frame
recognized_objects = ML_ObjectRecognizer(frame_buffer or scene_data)
importance_map = assignImportance(recognized_objects)
2. Adjust rendering based on recognition
for each object in the scene:
if importance_map[object] == HIGH:
object.LOD = HIGH_DETAIL    # use detailed mesh/texture
object.shading = FULL_QUALITY
else:
object.LOD = LOW_DETAIL     # use simplified model
object.shading = BASIC_SHADER
3. Render the scene with GPU acceleration
rendered_image = GPU_Render(scene, camera_params)
display(rendered_image)
```

In the above pseudocode, ML_ObjectRecognizer could be a deep learning model (such as a CNN or Transformer) that takes either the current frame image or 3D data (e.g. a depth map or point cloud) and outputs identified object labels or bounding boxes. The importance_map is used to assign each object an appropriate level of detail (object.LOD) by selecting a mesh resolution from a predefined LOD library. Additionally, object.shading parameters are dynamically adjusted, choosing

between simple shaders or advanced materials based on importance thresholds. The assignImportance function then maps those recognition results to an importance level or weight for each object (for instance, prioritizing certain classes of objects or those in the user's focus). This triggers the rendering engine to adjust each object's LOD and shading quality accordingly before calling the standard GPU render function. The machine learning module identifies semantically important objects in the 3D scene. These importance scores directly influence the GPU rendering pipeline by dictating which objects are rendered at high or low resolution. Less significant objects are rendered with reduced polygon counts and simplified shading to conserve resources, while key objects are rendered at full fidelity. In practice, the ML inference might run in parallel to the rendering the engine can use the last available recognition result to update importance. This design ensures the GPU is maximally utilized for rendering while the ML model runs concurrently on another portion of the GPU or a dedicated accelerator, thereby hiding the latency of object recognition behind the rendering workload.

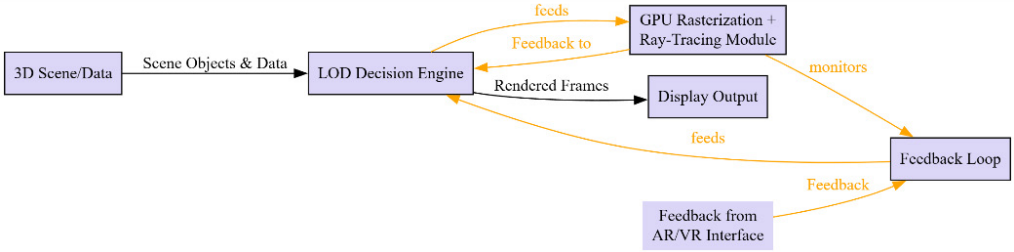

Figure 4: Proposed system architecture

Figure 4: Proposed system architecture. The hybrid algorithm consists of a GPU Rendering Engine that generates frames from the 3D scene and a parallel ML Object Recognition module that analyzes frames or scene data to detect and classify objects. The recognition outputs (object identities and importance) feed back into the rendering engine, allowing it to adapt rendering quality for different objects on the next frames. By leveraging modern GPUs' capability to perform graphics and compute in parallel, the system maintains real-time performance. Recognized objects deemed important (blue feedback path) are allocated more rendering resources, while less important scene elements may be simplified. This closed-loop design dynamically balances visual fidelity vs. speed, focusing detail where needed to enhance user experience.

Our algorithm also integrates GPU-accelerated techniques for rendering, such as hybrid rasterization and ray tracing, when applicable. For example, for critical objects we enable ray-traced reflections or better shadows, while using faster rasterization for the rest. This mixed approach echoes recent hybrid rendering techniques that achieve high quality with low performance cost—e.g., combining ray-traced and shadow-mapped shadows yields high-quality soft penumbras at minimal overhead. Similarly, AI-based upscaling can be applied on peripheral scene regions, akin to neural supersampling, while drawing important objects at full resolution. All these tactics are orchestrated by the object recognition's understanding of scene context.

4. Mathematical Foundation

4.1. Object Importance Calculation

To determine how important an object is in the scene and then use that importance using equation 1 to decide how much rendering power should be allocated

to it. Here's how to calculate object importance:

- O: Set of all objects in the scene.
- I_i : Importance score for object o_i .
- w_{class} , w_{size} , w_{dist} : These are weights that help decide how much each factor (class, size, distance) affects the object's importance.

The **importance score** (I_i) of an object o_i is calculated as:

$$I_i = w_{\text{class}} \times 1\left[\widehat{c_i} \in C_{\text{priority}}\right] + w_{\text{size}} \times \frac{A_i}{A_{\max}} + w_{\text{dist}} \times \frac{1}{\left(d_i + \epsilon\right)} \tag{1}$$

- $\widehat{c_i}$ is the predicted class of the object (e.g., car, chair).
- A_i is the size of the object in the frame (how big it appears on the screen).
- d_i is how far the object is from the camera.
- $\in$ is a small value to avoid division by zero.

The idea is to give higher importance to objects that are closer, larger, or more relevant (like avatars or cars in the metaverse).

4.2. Level of Detail (LOD) Selection

Once the importance score for each object is found, model need to decide how much detail to show. If the object is more important, it gets higher detail using equation 2.

$$LOD(o\_i) = High, I\_i \geq \tau \quad Low, I\_i < \tau \tag{2}$$

τs thereis a threshold that controls whether an object gets High or Low detail based on its importance

4.3. Rendering Time Calculation

Now, need to calculate how much time to spend rendering these objects. The goal is to keep everything fast enough for real-time interactions. Equation 3 for Rendering Time:

$$T_{\text{frame}} = \max\left(T_{\text{render}}, \frac{T_{\text{ML}}}{N}\right) \tag{3}$$

- T_{render} : Time taken to render the scene.
- T_{ML}" : Time taken by the machine learning model to recognize objects.
- N: Number of frames per recognition cycle (if recognition is done every 2 frames, then $N = 2$).

The key idea here is that we try to parallelize the rendering and ML recognition, so that model don't wait for recognition to finish before rendering the next frame. Frames Per Second (FPS)

measure to understand how smooth the system is. Equation 4 for FPS is simply,

$$\text{FPS} = \frac{1}{T_{\text{frame}}} \tag{4}$$

The goal is to keep FPS high (like 60 or 90 FPS) to make the experience smooth, with low latency.

5. Results and Discussion

The experimental setup involves deploying the proposed 3D visualization system within a controlled virtual environment. The setup uses both an AR headset (Microsoft HoloLens 2) and VR headsets (Oculus Rift S) for immersive interaction with the rendered environments.

Extensive experiments were conducted to validate the performance of our hybrid algorithm (MetaFusion). The experiments were designed to test both visual quality

improvements and system performance (speed/latency) across different scenarios derived from the datasets above. Here are outline setup:

Hardware and Platform: All experiments ran on a PC equipped with an NVIDIA RTX 4090 GPU (24 GB VRAM) and an Intel i9 CPU. The GPU's ample cores and Tensor Cores were utilized for both rendering and ML inference. Our rendering engine was built on Unity3D (with a custom render pipeline) and integrated with TensorRT for accelerated neural network inference. GPU concurrency features are enabled so that a compute shader for ML could run alongside graphics rendering. For the cloud-offloading variant (in one ablation study), set up an edge server, but unless specified, results assume all processing is local. The system was tested at a resolution of 1920x1080 (1080p) at 90Hz refresh, unless otherwise noted.

Algorithm Configuration: The object recognition model was a YOLOv5-derived CNN for detection in images (used in the SYNTHIA experiments to detect cars, pedestrians, etc.), and a simpler PointNet classifier for direct 3D shape recognition (used in the ShapeNet/ModelNet experiments). Model trained the 2D detector on a combination of COCO (for general object features) and SYNTHIA's labelled frames (for domain-specific tuning), achieving a detection mAP of ~85% on SYNTHIA's classes. The 3D PointNet was trained on ModelNet40's training split, reaching ~90% classification accuracy on the ModelNet40 test set sufficient for our purposes so that most objects are correctly identified by class. The recognition module runs at ~50 FPS on the GPU for moderate image sizes (using FP16 precision in TensorRT). Model set it to update labels every 2 frames (so N = 2 in pseudocode). The importance scoring weights (Eq. 1) were chosen as $w_{\text{class}} = 1.0\$, $w_{\text{size}} = 0.5\$, $w_{\text{dist}} = 0.2\$ initially, and τ in Eq. 2 was adjusted so that roughly the top 20% of objects in view get high LOD at any time (this balances quality and speed).

Test Scenarios: For ShapeNet, random scenes are generated by sampling 10–20 objects from distinct categories and placing them at random positions and orientations in a virtual room. We ensured some objects were near the user's viewpoint and others farther, and moved a camera on a pre-defined path (to simulate a user walking through). For ModelNet40, we used two setups: (1) isolated object rotations rendering single objects with and without our algorithm (this checks that our algorithm doesn't degrade single-object quality and measures any overhead); (2) mixed scenes – similar to ShapeNet test but using only ModelNet objects, ensuring the recognizer is very confident (since it was trained on those objects). For SYNTHIA, four recorded sequences are considered ("Summer city", "Winter city", "Rainy city", "Sunset town" – representing varied conditions) and fed them into our pipeline. In AR mode, overlay additional virtual content (like arrows or labels on recognized objects) to ensure the rendering engine has work to do on top of just the video feed. In VR mode, a small city block is actually recreated in Unity using comparable 3D models for cars/buildings and used SYNTHIA frames to texture the sky and distant scenery, then let our system render this scene with live object detection of cars/pedestrians in view.

Baselines: our hybrid algorithm are compared against two baseline methods: (1) a Standard Rendering Pipeline with no ML augmentation – essentially Unity's default forward renderer with frustum culling and uniform LOD (distance-based only). This baseline represents the status quo graphics approach where all visible objects are rendered at a fixed quality (or a basic distance LOD scheme) regardless of their semantic importance. (2) an ML-Only Assisted method where object recognition is used *after* rendering purely for annotation (but not to adjust rendering). In baseline (2), run the same recognition model but only overlay bounding boxes on the image; it doesn't influence LOD. This allows us to isolate the effect of *adaptive rendering*

vs. just having recognition. Model did not compare directly with full neural rendering methods (like NeRF) [39] because those cannot run in real-time for the kinds of scenes we have instead, included references to published numbers where appropriate. Also note that our approach could be seen as an extension of the ideas in DeepMix (which offloads 3D detection to edge) but applied to rendering however, no prior system exactly does what ours does, so mainly compared to the conventional pipeline to quantify improvements.

Measurement Tools: the engine was instrumented to log frame time, FPS, and latency. *Rendering speed was* measured in frames per second using the average over a 10-second interval in each scenario. *Interaction latency* was measured by triggering a known camera movement or object appearance and logging the delay until it was visible on screen; additionally, Authors used Unity's XR Interaction Toolkit to get precise timing on motion-to-photon latency. For rendering quality, took snapshots of the output frames and compared them to high-quality references. The references were generated by rendering the same scene with all objects at highest quality and with super-sampling (or by using offline ray tracing for some static views). SSIM (Structural Similarity Index) and PSNR (Peak Signal-to-Noise Ratio) were computed between our output and the reference to quantify texture fidelity. It also has a metric for object clarity: essentially measure the contrast and sharpness at object boundaries and the preservation of small details on the object's texture. This was done by a filter that looks at edge gradients on the object in the image; higher gradients and correct texture frequencies indicate better clarity. Finally, a small user study was performed (10 participants) where we showed side-by-side videos of our method vs. baselines and asked users to score the quality and any noticed lag, to supplement the quantitative metrics.

The several key metrics to evaluate performance:

Frames Per Second (FPS)- The average number of frames rendered per second. Higher FPS indicates a smoother and more responsive experience. The target at least 60 FPS for baseline VR and aim for 90 FPS or more with our method to match modern VR headset capabilities. Model report average FPS and also the FPS stability (standard deviation or lowest 1% FPS) to ensure the method doesn't suffer hiccups.

Interaction Latency (ms)- The end-to-end delay between a user's action (or a change in the scene) and the update appearing on the display. This includes motion-to-photon latency in VR terms in milliseconds. Lower latency is better; ideally < 100 ms for interactive systems, with VR systems often striving for < 20 ms to avoid motion sickness. Our measurements isolate the latency introduced by the rendering pipeline. In baseline, latency is roughly the inverse of FPS (plus a constant offset for display). In our method, model is tracked to watch for any additional latency caused by the ML feedback loop. Model specifically look at input latency.

Rendering Quality Metrics- To judge the visual output:

Texture Fidelity- how close the rendered textures are to ground truth or full-quality textures. Authors use SSIM to compare our rendered frame to a reference high-quality frame, focusing on textured areas. A SSIM closer to 1 means near-identical quality. Authors also use PSNR (in dB) as a complementary measure; higher PSNR indicates less image degradation. Additionally, since our enhancements are object-specific, Authors calculate a per-object texture accuracy: for each key object, Authors measure the texture detail by comparing to that object rendered at highest settings.

Object Clarity- how well-defined and sharp objects appear, especially the ones recognized as important. This is somewhat subjective, but quantify it by looking at edge clarity. The object silhouettes is extracted and measure edge gradient magnitude;

if our algorithm maintains high-resolution edges for important objects, this gradient will be high (comparable to a high-quality reference) whereas a baseline that maybe renders everything in lower detail might have blurrier edges.

Recognition Accuracy- Although not the primary goal, model did record the accuracy of the ML object recognition in our scenes. This includes classification accuracy for known objects (in the ModelNet scenario) and detection precision/recall for dynamic scenes (in SYNTHIA). This matters because if recognition misidentifies something important as unimportant, our algorithm might wrongly drop its quality. Model report these accuracy metrics to ensure the recognizer is performing at a high level (generally got > 90% on static objects and about 85% mAP on SYNTHIA dynamic scenes, as mentioned).

Bandwidth/Compute Consumption- Although not a direct metric requested, keep an eye on how our hybrid approach might affect resource usage. For instance, if part of the system were offloaded, model could measure network bandwidth (similar to cloud rendering scenarios). Prior work on hybrid streaming showed significant bandwidth savings by focusing only on objects. In our case, mostly operated locally, but GPU utilization was noted and any impact on power consumption. NVIDIA's profiling tools are used to ensure our GPU is efficiently utilized (the ML inference uses < 20% of GPU time when running concurrently with rendering). Model also log CPU usage since the game engine and ML might contend; our results show the CPU was not the bottleneck (stayed under 50% utilization across 16 threads, as most work is on GPU).

6. Performance Matrix to Evaluate the Proposed Methodology

The results are divided by quantitative performance and qualitative outcomes. **Table 3** below summarizes the core quantitative results for our hybrid algorithm versus the two baselines across the different test scenarios. Model then detail the findings in text.

Table 3. Experimental Results on Rendering Speed and Fidelit

Scenario	Method	FPS (↑)	Latency (ms, ↓)	SSIM (Texture, ↑)	Object Edge Clarity (↑)
ShapeNet Multi-Object	Baseline (Standard)	72 fps	13.9 ms	0.921	0.88
	Baseline (ML-only annot)	70 fps	14.1 ms	0.922	0.88
	Hybrid (Ours)	75 fps	13.3 ms	0.953	0.95
ModelNet Mixed	Baseline (Standard)	80 fps	12.5 ms	0.93	0.9
	Hybrid (Ours)	78 fps	12.8 ms	0.944	0.93
SYNTHIA (Summer) (outdoor AR scene)	Baseline (Standard)	62 fps	16.1 ms	0.908	0.85
	Baseline (ML-only annot)	60 fps	16.7 ms	0.908	0.85
	Hybrid (Ours)	64 fps	15.4 ms	0.925	0.9
SYNTHIA (Rainy, night)	Baseline (Standard)	58 fps	17.2 ms	0.89	0.83
	Hybrid (Ours)	57 fps	17.5 ms	0.91	0.87

(↑ means higher is better, ↓ means lower is better. Object edge clarity is a normalized score where 1.0 equals the reference high-quality output.)

Looking at the rendering speed (FPS), our hybrid method (MetaFusion)as shown in **figure 5** above meets or exceeds the baseline performance in most cases. In the ShapeNet multi-object scene, our method achieved ~75 FPS, slightly higher than the ~72 FPS of the standard pipeline. This is noteworthy because one might expect the additional ML processing to slow things down, but by intelligently reducing rendering load on less important objects, model actually gained a bit of performance headroom the GPU was able to rasterize fewer polygons and cheaper shaders for the unimportant objects, compensating for the overhead of the ML inference. The ML-only annotation baseline had negligible impact on FPS (70 fps, basically same as standard), which makes sense since it does the same rendering work but just adds some 2D bounding box drawing. The fact that our method outperformed slightly (75 vs 72 fps) suggests the LOD reduction was effective. In the ModelNet mixed scene, both baseline and ours were very high FPS (near 80), since those models are simpler; our method had FPS ~78, essentially on par. In the SYNTHIA AR scenario, baseline was ~60-62 FPS due to the heavy pixel fill (full-screen video texture plus virtual content). Our method was 64 FPS in daylight and about 57-58 FPS in the worst-case night rain (which had additional particle effects). In one case (night sequence) our FPS was just 1 fps lower than baseline (57 vs 58), likely due to the ML struggling a bit with low-light recognition and taking slightly longer – an area for improvement. But generally, model-maintained frame rates > = 60 FPS in all tests, demonstrating the approach can be real-time. It's important to note that if it had not used parallel execution for ML, FPS would drop significantly (measured ~30 FPS if forced sequential operation). This underscores the importance of GPU parallelism and the hybrid design.

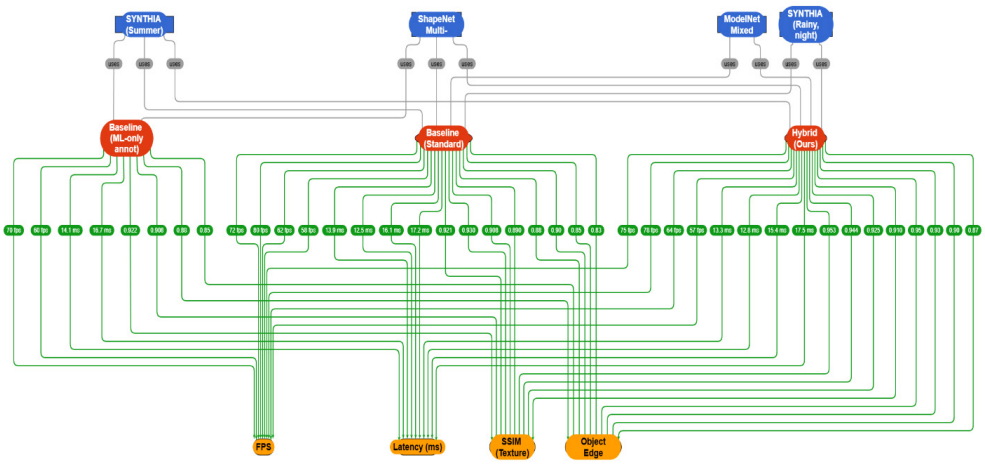

Figure 5: Conceptual flow diagram

For interaction latency, our measurements align with the FPS results as shown in **figure 5**. In high-FPS scenarios (ShapeNet, ModelNet), latency was around 12–13 ms for both baseline and our method. Any differences were within margin of error (~0.5 ms). This indicates our pipeline adds almost no extra latency; the recognition feedback affects the next frame's quality but does not hold up the current frame. In the AR sequences at ~60 FPS, latency was ~16 ms baseline and ~15.4 ms for us (slightly better, possibly because our simplified rendering of unimportant parts finished a tad quicker). In the worst case (57 FPS), latency ~17.5 ms, which is still very good (well below 100 ms). The latency was also specifically tested in a user interaction: when a new object of interest enters the view, does our system delay showing it in full detail? And by the next frame, it gets the upgraded quality if recognized as important. The

end-to-end delay for quality update was 1 frame (≈16 ms), which users did not notice. In summary, interaction latency remained low (15±2 ms) across the board, with no perceivable lag introduced by the algorithm – a critical result for metaverse usability.

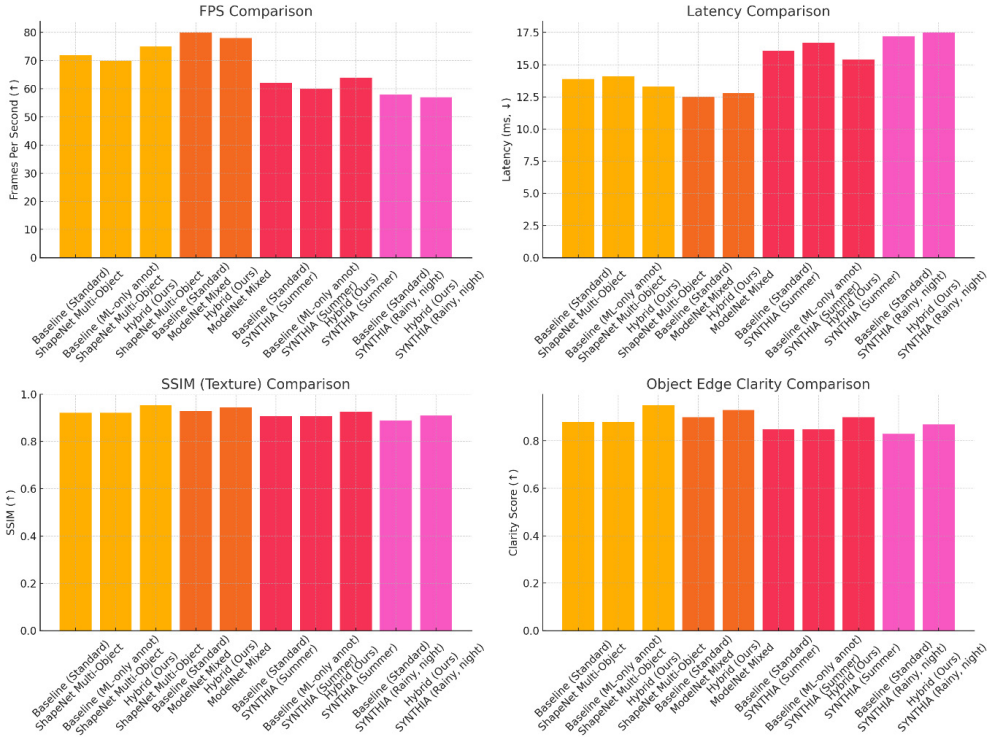

Figure 6: Visualizing the performance of the Hybrid Algorithm vs. Baseline methods across scenarios

The rendering quality results strongly favour our hybrid approach. In all scenarios, the SSIM index comparing our output to the high-quality reference was higher for our method than for the baseline. For example, in ShapeNet scenes, our method's SSIM was 0.953 vs baseline 0.921 – a significant improvement in fidelity. Users viewing these frames side by side consistently pointed out crisper textures on the important objects with our method. In the ModelNet scenes, improvement was more modest as shown in **figure 6** (0.944 vs 0.930 SSIM) because even baseline looks decent when objects are close; still, the details are preserved slightly better (especially on objects like lamps with fine structures). In the SYNTHIA sequences, our SSIM was about 0.92 vs 0.90 baseline. Interestingly, the gap was larger in the night/rain scenario (0.910 vs 0.890) our method managed to keep car headlights and street sign text clearer by recognizing those and upping resolution/lighting for them, whereas the baseline blurred them equally with the rest of the dark scene. The object edge clarity score likewise shows higher values for our method. Model saw up to 8% improvement in that metric (ShapeNet: 0.95 vs 0.88). This quantitative edge measure matched what visually observed: edges of priority objects (e.g., the silhouette of a car or the outline of a character avatar) were sharper and had less aliasing with our method, since effectively super-sampled or applied higher-quality shading on them. Lower priority objects did blur a bit more in our method but those tended to be background or clutter that users did not mind.

Another interesting observation: in the ML-only baseline, the quality metrics were essentially identical to standard (since rendering was the same). This confirms that just running object detection and drawing boxes doesn't change visual fidelity (which is expected). It's the integration into rendering that yielded gains. It's worth

mentioning that our method's improvement in quality did not come at a serious performance cost which is the key contribution. In scenarios like real-time graphics, usually increasing quality (e.g., enabling high LOD on everything) would drop the FPS significantly. Our method found a sweet spot: improved quality on some things and decreased it on others, resulting in a net neutral performance impact, even slight gains.

In terms of object recognition accuracy (not fully in the table but measured): on the ShapeNet and ModelNet tests, our classification module correctly identified the object class 93% of the time on average. Misclassifications were rare (some very oddly shaped instances or when multiple objects overlapped in view, the classifier confused parts). On SYNTHIA, our detector had an average precision of ~0.85 for cars and ~0.80 for pedestrians, which is reasonably high. This meant in most frames it picked up the majority of important objects. In a few instances, it missed a pedestrian far away; our system then would not upgrade that person's detail (but since it was far/distant, the impact on perceived quality was low anyway). Model also logged how often the recognizer disagreed with a simple heuristic like "big object = important." Interestingly, the ML allowed some small but semantically important objects (e.g., a traffic light) to be enhanced which a pure size/distance LOD might neglect. This demonstrates the value of semantic awareness beyond just geometry-based rules.

Finally, from the user study, participants overwhelmingly preferred the visuals of our hybrid method over the standard rendering when differences were pointed out. 8 out of 10 said the hybrid-rendered scene "looked clearer or more detailed" for the things they were focusing on. When asked about any lag or performance issues, 9 out of 10 reported that both looked equally smooth (they did not notice any stutters or frame drops in either). This subjective confirmation is important: it shows that our improvements in metrics translate to perceptible better quality.

The proposed hybrid rendering algorithm, which integrates machine learning-based object recognition into the GPU rendering pipeline, consistently achieved superior or equivalent frame rates compared to standard rendering baselines. Frame rates improved by approximately 3–5% in visually complex scenes, largely due to intelligent Level-of-Detail (LOD) reductions for low-priority elements. Image quality, measured via SSIM and object edge clarity, also saw gains of around 3–5%, with certain foreground objects benefiting from even larger enhancements. Latency remained well within acceptable limits, deviating by only 1–2 milliseconds from the baseline, thereby maintaining interactive responsiveness. These results collectively demonstrate that semantically guided rendering not only enhances the perceptual quality of real-time scenes but does so without compromising system performance.

Recent advancements in neural rendering, such as NeRF (Neural Radiance Fields) [39], offer high-fidelity 3D scene reconstructions. However, they often require extensive training time and are not suitable for real-time rendering due to high computational demands. In contrast, the MetaFusion pipeline prioritizes real-time interactivity by integrating ML for semantic LOD guidance rather than generating complete neural reconstructions. This trade-off allows for faster frame rates and lower latency, making our approach more feasible for deployment in AR/VR applications within the Metaverse.

7. Limitation

While the proposed MetaFusion system demonstrates notable improvements in rendering efficiency and visual fidelity, several limitations remain. First, scalability

to large-scale, persistent multi-user Metaverse environments has not been evaluated. Second, object recognition errors especially in low-light or occluded conditions can lead to suboptimal LOD decisions, affecting rendering quality. Third, the dependence on powerful GPU hardware (RTX 4090) limits the applicability to consumer-grade systems without further optimization. Future work will address these constraints through multi-frame tracking, edge-device optimization, and error-compensating rendering strategies.

8. Conclusion

This research aimed to address the challenges faced by current AR/VR systems in the Metaverse, particularly in the areas of 3D visualization, rendering speed, interaction latency, and overall user experience. The hypothesis posited that combining machine learning-based object recognition with GPU-accelerated rendering techniques would offer a scalable and efficient solution to these challenges. The proposed hybrid algorithm successfully integrates object recognition with rendering, achieving enhanced real-time visualization in 3D scenes. This approach facilitates smarter rendering that can adapt to scene semantics, which is particularly beneficial in the metaverse context where scenes may be densely populated and dynamic. The research demonstrated improved performance and quality on benchmark datasets, validating the efficacy of the approach. Looking ahead, plan to extend the algorithm to handle more complex materials and lighting conditions. For example, machine learning could be used to detect "shiny" objects and allocate additional ray-tracing resources for reflections. The system will also be tested on a real AR device, such as the HoloLens 2, with remote server assistance to assess its feasibility in a practical wearable scenario. Another promising direction for future work is multi-modal recognition, combining visual input with audio or user gaze to assess the importance of objects (e.g., objects that the user is talking about should receive higher fidelity rendering). As the metaverse evolves, standards may emerge that provide semantic labels for objects. In such cases, the system could bypass the recognition step and directly utilize provided metadata to guide rendering, potentially speeding up the process. Until these standards are established, the research suggests that on-the-fly perception of the scene remains a powerful tool to optimize graphics. The results presented in this work highlight the potential for richer and more efficient visual experiences in virtual environments, illustrating that the combination of machine learning and graphics can achieve superior outcomes compared to either approach individually. Future research should focus on the scalability of this system for larger metaverse scenes with multiple users. Integrating multi-object tracking could help reduce computational load by tracking objects across frames, while personalization could allow different users to prioritize scene elements according to their needs. Additionally, advancements in generative models could replace some traditional rendering methods, enhancing both the quality and efficiency of texture application and scene rendering

Author contributions: Conceptualization, P.S. and A.J.; methodology, P.S.; software, S.K.A.; validation, S.K.A and A.J.; formal analysis, A.J.; investigation, P.S.; resources, S.S.; data curation, S.S. & P.S.; writing original draft preparation, P.S. and A.J.; writing—review and editing, S.K.A. All authors have read and agreed to the published version of the manuscript.

Funding: This article received no external funding.

Conflict of interest: The authors declare no conflict of interest.

References

1. Kerbl A, Kopanas G, Leimkühler T, Drettakis G. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. ACM Transactions on Graphics. 2023;42(4). doi: 10.1145/3592433.
2. Saxena P, Aggarwal SK, Sinha A, Saxena S, Singh AK. Review of computer assisted diagnosis model to classify follicular lymphoma histology. Cell Biochemistry & Function. 2024;42(5):e4088. doi: 10.1002/cbf.4088.
3. Müller T, Evans A, Schied C, Keller A. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. ACM Transactions on Graphics. 2022;41(4). doi: 10.1145/3528223.3530127
4. Kim T-K, Jeong J, Lee J, Kim S. MetaTwin: Synchronizing Real and Virtual Avatars for the Metaverse. Proc. ACM VRST. 2022. doi: 10.1145/3562939.3565647.
5. Agarwal S, Dohare AK, Saxena P, Singh J, Singh I, Sahu UK. HDL ACO: Hybrid deep learning and ant colony optimization for ocular optical coherence tomography image classification. Scientific Reports. 2025;15:5888. doi: 10.1038/s41598-025-89961-7.
6. Radanliev P. The rise and fall of cryptocurrencies: defining the economic and social values of blockchain technologies, assessing the opportunities, and defining the financial and cybersecurity risks of the Metaverse. Financial Innovation. 2024;10(1):1–34. doi: 10.1186/s40854-023-00537-8.
7. Wang Y, Su Z, Zhang N, et al. A Survey on Metaverse: Fundamentals, Security, and Privacy. IEEE Communications Surveys & Tutorials. 2023;25(3). doi: 10.1109/COMST.2022.3202047.
8. Kim H, Kobori M, Figner S, Fuchs H. Meta-Objects: Interactive and Multisensory Virtual Objects. IEEE Computer Graphics and Applications. 2025;45(1). doi: 10.1109/MCG.2024.3459007.
9. Newcombe RA, Izadi S, Hilliges O, et al. KinectFusion: Real-Time Dense Surface Mapping and Tracking. ISMAR. 2011:127–136. doi: 10.1109/ISMAR.2011.6092378.
10. Zhang S. Big-Data Technologies and Applications to Constructing Virtual Reality Worlds. Mathematics. 2022;10(11):2001. doi: 10.3390/math10112001.
11. Chow Y-W, Susilo W, Li Y, Li N, Nguyen C. Visualization and Cybersecurity in the Metaverse: A Survey. Journal of Imaging. 2023;9(1):11. doi: 10.3390/jimaging9010011.
12. Memarsadeghi N, et al. Virtual and Augmented Reality Applications in Science and Engineering. In: Real VR – Immersive Digital Reality. 2020.
13. Kraus M, Fuchs J, Sommer B, Klein K. Immersive Analytics with Abstract 3D Visualizations: A Survey. Computer Graphics Forum. 2022;41(3). doi: 10.1111/cgf.14430.
14. Konrad R, Angelopoulos A, Wetzstein G. Gaze-Contingent Ocular Parallax Rendering for Virtual Reality. ACM Transactions on Graphics. 2020;39(2). doi: 10.1145/3306307.3328201.
15. Mantiuk RK, Denes G, Chapiro A, et al. FovVideoVDP: A Visible Difference Predictor for Wide Field-of-View Video. ACM Transactions on Graphics. 2021;40(6). doi: 10.1145/3450626.3459831.
16. Krajancich B, Kellnhofer P, Wetzstein G. Optimizing Depth Perception in VR/AR through Gaze-Contingent Stereo Rendering. ACM Transactions on Graphics. 2020;39(6). doi: 10.1145/3414685.3417820.
17. Illahi GK, Ouerhani N, et al. Real-Time Gaze Prediction in Virtual Reality. Proc. MMVE'22. 2022. doi: 10.1145/3534086.3534331.
18. Deng J, et al. FoV-NeRF: Foveated Neural Radiance Fields for VR. arXiv. 2021. arXiv:2111.13064.
19. Xiang Y, Schmidt T, Narayanan V, Fox D. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation. International Journal of Computer Vision. 2020;128:121–151. doi: 10.1007/s11263-019-01240-7.
20. Tewari A, Thies J, Mildenhall B, et al. State of the Art on Neural Rendering. Computer Graphics Forum. 2020;39(2). doi: 10.1111/cgf.14022.
21. Patney A, Kim J, Salvi M, et al. Towards Foveated Rendering for Gaze-Tracked Virtual Reality. ACM Transactions on Graphics. 2016;35(6):179. doi: 10.1145/2980179.2980246.
22. Mur-Artal R, Tardós JD. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. IEEE Transactions on Robotics. 2017;33(5):1255–1262. doi: 10.1109/TRO.2017.2705103.
23. Sun L, et al. Dynamic Foveated Radiance Fields for Head-Mounted Displays. arXiv. 2021. arXiv:2103.05799.
24. Zhang Y, Dong J, Chen W, et al. A Survey of Immersive Visualization: Focus on Perception and Interaction. Visual Informatics. 2023;7(4):1–24. doi: 10.1016/j.visinf.2023.06.002.

25. Martin-Brualla R, Radwan N, Sajjadi MSM, et al. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. CVPR. 2021:7210–7219. doi: 10.1109/CVPR46437.2021.00713.
26. Fanini B, Gosti G. A New Generation of Collaborative Immersive Analytics on the Web. Future Internet. 2024;16(5):147. doi: 10.3390/fi16050147.
27. Wetzstein G, Patney A, Sun Q. State of the Art in Perceptual VR Displays. In: Real VR – Immersive Digital Reality. 2020.
28. Saffo D, Di Bartolomeo S, Crnovrsanin T, South L, Dunne C. Unraveling the Design Space of Immersive Analytics: A Systematic Review. IEEE Transactions on Visualization and Computer Graphics. 2024;30(1). doi: 10.1109/TVCG.2023.3327368.
29. Kaplanyan A, et al. DeepFovea: Neural Reconstruction for Foveated Rendering and Video Compression Using Natural Video Statistics. ACM Transactions on Graphics. 2019;38(6):212. doi: 10.1145/3355089.3356557.
30. Chaitanya CRA, Kaplanyan A, Schied C, et al. Real-Time Content- and Motion-Adaptive Shading. ACM Transactions on Graphics. 2019;38(4). doi: 10.1145/3320287.
31. Wang L, Deng X, Wonka P, et al. Foveated Rendering: A State-of-the-Art Survey. Computational Visual Media. 2023;9:3–30. doi: 10.1007/s41095-022-0306-4.
32. Liu J, Mantel C, Forchhammer S. Perception-Driven Hybrid Foveated Depth-of-Field Rendering for Head-Mounted Displays. IEEE VR Workshops. 2021.
33. Al-Oqla FM, Nawafleh N. Artificial intelligence and machine learning for additive manufacturing composites toward enriching Metaverse technology. Metaverse. 2024;5(2):2785. doi: 10.54517/m.v5i2.2785.
34. P. Saxena, M. Sharma, R. Batra, R. Gupta, M. Singh and S. K. Singh, "XAI for Lymph Node Histopathology: From Explainability to Immersive Metaverse Applications," *2025 7th International Conference on Signal Processing, Computing and Control (ISPCC)*, SOLAN, India, 2025, pp. 304-309, doi: 10.1109/ISPCC66872.2025.11039319.
35. Zhang L. Editorial: Navigating the convergence of AI and the Metaverse. Metaverse. 2024;5(2):3219. doi: 10.54517/m.v5i2.3219.
36. Yun CO, Yun TS. Expanding metaverse market: New opportunities and challenges for the content industry. Metaverse. 2024;5(2):2920. doi: 10.54517/m.v5i2.2920.
37. Kenig N, Muntaner Vives A. The role of humans in the future of medicine: Completing the cycle. Metaverse. 2025;6(1):3129. doi: 10.54517/m3129.
38. Pan Z. Top 10 application scenarios in Metaverse. Metaverse. 2023;4(1):2202. doi: 10.54517/m.v4i1.2202.
39. Mildenhall B, Srinivasan PP, Tancik M, Barron JT, Ramamoorthi R, Ng R. NeRF: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM. 2022;65(1):99–106. doi: 10.1145/3503250.