Adaptive Multi-Resolution Rendering for Virtual Reality Scenes: A Dynamic Resolution and Task Scheduling Approach

Shanshan Yang *, Yiming li
College of Information Engineering, Jiaozuo University, Jiaozuo 454000, Henan, China
E-mail: yss15803915433@126.com
*Corresponding author

Keywords: Virtual reality, rendering optimization, adaptive multi-resolution model, rendering frame, image quality

Received: June 5, 2025

This paper presents an adaptive multi-resolution rendering optimization algorithm for virtual reality (VR) environments based on real-time eye-tracking and dynamic model complexity analysis. The proposed method integrates three key modules: a visual focus detection component using a 2D Gaussian sensitivity function centered on the gaze point, a model importance evaluation mechanism based on texture density and triangle count, and a resolution adjustment scheduler that assigns level-of-detail (LOD) dynamically. To validate performance, comparative experiments were conducted against two baseline methods: fixed-resolution rendering and heuristic rule-based adaptation. Results demonstrate a consistent frame rate improvement from 81 fps to 121 fps (p<0.01), while maintaining subjective visual quality above 7.8 on a 10-point scale. Standard deviation remained within ±2.5 fps across multiple scenes, confirming runtime stability. The algorithm was tested on both indoor and outdoor VR scenes, with additional robustness tests under dynamic object and gaze shifts. The mathematical modeling of the sensitivity score and adaptive resolution mapping is lightweight, enabling real-time execution on mid-tier GPUs. This framework supports future deployment in both high-fidelity immersive experiences and resource-constrained VR systems.

Povzetek: Članek predstavi prilagodljiv algoritem za VR, ki z očesnim sledenjem, oceno pomembnosti modelov in dinamičnim LOD razporejevalnikom v realnem času zviša FPS ob ohranjeni kakovosti tudi na povprečnih GPU-jih.

1 Introduction

In today's digital era, computer technology is advancing rapidly, and virtual reality (VR) has emerged as one of the most prominent fields. According to incomplete statistics, more than 5,000 new companies entered the VR industry in the past year alone, and the market size is projected to exceed USD 200 billion within the next three years [1]. VR technology is being applied across a wide range of domains, including gaming, entertainment, education, training, medical surgery simulation, and architectural visualization. However, despite this growth, a critical and urgent challenge persists: balancing rendering efficiency with visual quality in VR scenes.

For instance, many mainstream large-scale VR games feature scenes containing over 100,000 models. These models are highly complex and detailed, requiring substantial processing of texture and lighting data during rendering. On standard computing setups, the resulting frame rates are often too low—sometimes dropping to as low as 10 frames per second—leading to user discomfort, disorientation, and a diminished immersive experience [2]. In professional VR applications such as architectural design previews, high-precision models often include dense polygonal structures and complex materials, resulting in rendering times that can span several hours. This significantly reduces workflow efficiency and discourages widespread adoption of VR for design purposes.

Currently, numerous research teams and enterprises worldwide are investing substantial efforts in improving VR scene rendering. From an algorithmic standpoint, advancements have been made on traditional graphics techniques. For example, some groups have optimized ray tracing algorithms to enhance visual realism; however, these improvements often come at the cost of higher computational complexity, which can reduce rendering speed. In certain complex scenes, rendering times have increased by more than 30% compared to unoptimized baselines. Deep learning-based rendering methods have also been proposed, where neural networks trained on large image datasets predict scene rendering outcomes. These approaches show promising results in controlled scenarios such as simple indoor environments, achieving rendering time reductions of approximately 20%. Nevertheless, their limitations become evident in complex, dynamic outdoor scenes, where prediction accuracy drops and rendering quality is often unsatisfactory [3].

In terms of hardware acceleration, major graphics card manufacturers continue to release GPUs with increasingly powerful performance and have implemented specific optimizations for VR rendering. However, hardware advancements have consistently lagged behind the growing computational demands of VR scene rendering. Furthermore, rendering algorithms exhibit significant variability in compatibility across different hardware platforms, limiting the effectiveness of many high-performance algorithms in diverse environments. Current research in this area primarily focuses on maximizing rendering speed without sacrificing image quality, as well as improving algorithm adaptability across various hardware configurations and VR application scenarios. However, considerable debate exists among research teams regarding the optimal path forward[4]. Some advocate prioritizing hardware upgrades with complementary algorithmic improvements, while others argue that algorithmic innovation alone can overcome hardware constraints and drive significant advancements using existing computational infrastructure[5].

Against this backdrop, this study proposes a VR scene rendering optimization algorithm based on an adaptive multi-resolution model. The algorithm dynamically adjusts the resolution of scene models according to their perceived importance and the user's visual focus, thereby minimizing redundant computations and enhancing overall rendering efficiency without compromising the quality of key visual regions[6]. The method is expected to increase the average frame rate of complex VR scenes by over 50% and reduce rendering time by approximately 40%.

Theoretically, this research contributes to the foundational understanding of VR rendering algorithms and introduces novel approaches for further development. Practically, it holds promise for improving the realism and responsiveness of VR gaming, as well as enhancing performance in professional applications such as architectural visualization and surgical simulation, thus offering substantial utility and economic potential.

2 Literature review

2.1 Research on virtual reality scene rendering algorithms

In the field of virtual reality (VR) scene rendering, traditional graphics algorithms have long served as a foundational research area. Considerable efforts have been devoted to techniques such as ray tracing [7]. While ray tracing algorithms are known to enhance visual realism, they also introduce substantial computational complexity, which significantly reduces rendering speed. Studies have shown that, in certain complex scenes, rendering times can increase by more than 30% compared to unoptimized baselines [8]. This performance bottleneck is widely recognized as a major limitation in VR applications, as low frame rates can cause discomfort, including motion sickness, thereby undermining the user experience. Nonetheless, the quality improvements brought by ray tracing offer valuable insights for future algorithm development.

In parallel, deep learning-based rendering algorithms have emerged as a promising research direction. These approaches predict rendering results by training neural networks on large-scale image datasets, achieving noticeable improvements in specific scenarios [9]. For example, rendering times in simple indoor scenes can be reduced by approximately 20%, a result that is encouraging. However, deep learning methods exhibit limited generalizability. In complex outdoor environments, prediction accuracy declines sharply, resulting in unsatisfactory rendering performance. This limitation has been widely noted, with concerns raised about their adaptability to diverse VR scene requirements. Despite this, the data-driven approach has introduced innovative perspectives into VR rendering research [10].

Both traditional and deep learning-based algorithms face challenges regarding hardware compatibility. Although GPU manufacturers continue to release VR-optimized hardware, rendering algorithm performance still varies significantly across platforms. Many algorithms with theoretical advantages perform poorly in real-world environments due to hardware limitations [11]. As a result, hardware compatibility has become a critical consideration in the design and optimization of VR rendering algorithms [12].

Prior work on foveated rendering using eye-tracking—such as Patney (2016) demonstrated the potential of spatial resolution scaling to reduce GPU load while preserving perceptual fidelity [13]. This study differentiates itself by integrating real-time gaze-based sensitivity mapping with task-aware rendering scheduling, which prior methods often overlook. Additionally, it addresses system-level balance between resolution control and GPU workload distribution. The contribution thus lies in combining existing concepts into an operational and scalable pipeline suitable for current-generation VR devices.

2.2 Hardware-related research on virtual reality scene rendering

In terms of hardware, the continuous advancement of GPU performance remains a critical driver of progress in VR scene rendering. High-performance GPU products released by major manufacturers have helped alleviate some of the computational load associated with VR rendering. However, the rate of GPU performance improvement has consistently lagged behind the increasing demands of VR rendering workloads[14]. Data indicates that the complexity and data volume of VR scene models are growing at an annual rate of approximately 20%, while GPU performance improves at only around 10% per year. This disparity has resulted in hardware being relatively inadequate for meeting current VR rendering demands[15]. Despite optimization efforts by hardware manufacturers, the performance gap continues to be a major constraint, limiting the rendering quality achievable in complex VR scenes under existing hardware conditions[16].

Furthermore, rendering algorithm performance varies significantly across different hardware platforms. In some case s, a rendering algorithm may function optimally on one platform but fail to operate correctly on another[17]. Surveys suggest that nearly 30% of advanced rendering algorithms cannot be widely adopted due to hardware limitations. These disparities reduce algorithm universality and force developers to account for various hardware constraints, thereby increasing development complexity and cost. This challenge has been identified as a key issue in VR rendering, encouraging deeper collaboration between hardware vendors and algorithm designers[18].

Recognizing these challenges, researchers increasingly emphasize the need for coordinated hardware-software development. Neither hardware upgrades nor algorithmic optimizations alone can resolve the trade-off between rendering efficiency and quality. Some propose dynamic feedback mechanisms that enable algorithms to adjust strategies based on hardware capabilities, and vice versa[19]. However, implementing such mechanisms presents significant technical challenges, such as real-time performance monitoring, efficient parameter feedback, and multi-algorithm optimization without escalating hardware cost. These challenges represent critical areas of current and future research.

Recent advancements in GPU architectures—such as NVIDIA's Ada Lovelace and AMD's RDNA3—introduce specialized ray tracing cores and AI-driven scheduling units, offering new paradigms for rendering optimization. The proposed method aligns with these architectures by offloading visual focus computations to dedicated tensor cores and scheduling via asynchronous compute queues. However, VR rendering on mobile GPUs (Apple M2 or Qualcomm Adreno series) remains limited by thermal envelopes and memory bandwidth, necessitating lighter-weight adaptive mechanisms. Future extensions could target real-time upscaling integration using onboard ML accelerators.

2.3 Research and development trends of adaptive multi-resolution models

As an emerging concept, adaptive multi-resolution models have gradually attracted attention in VR scene rendering. The core principle is to dynamically adjust the resolution of models based on their importance within the scene and the user's viewing perspective [20]. This approach significantly reduces unnecessary computation while preserving visual quality in key user-focused areas [21]. Preliminary test data indicate that in relatively simple VR scenes, applying adaptive multi-resolution models can improve rendering frame rates by approximately 30% and reduce rendering time by about 25%. These promising results highlight the model's potential in addressing the trade-off between rendering efficiency and quality, encouraging further research in the field. Nevertheless, several technical challenges remain. Determining the relative importance of each model and accurately identifying user visual focus require a combination of factors such as geometric complexity, texture attributes, and user interaction patterns. No standardized or widely adopted evaluation method currently exists. Furthermore, ensuring seamless resolution transitions during dynamic updates remains difficult, as abrupt changes can lead to noticeable visual artifacts. These unresolved issues continue to limit the large-scale deployment of adaptive models in complex VR scenarios.

Despite these challenges, the development trajectory of adaptive multi-resolution models remains positive. Advances in artificial intelligence and computer vision are expected to provide technical support. AI can enhance model importance estimation and gaze prediction, while computer vision may improve transition smoothness. Additionally, integration with other rendering algorithms and hardware platforms represents a key future direction, offering complementary benefits. Adaptive multi-resolution models are anticipated to play an increasingly vital role in advancing VR scene rendering technologies.

3 Research methods

3.1 Theoretical basis and model construction ideas

The primary objective of this study is to improve the real-time rendering frame rate of virtual reality scenes while maintaining a subjective image quality score above 7.5 through dynamic resolution control. The proposed method aims to achieve this by detecting the user's visual focus in real time, evaluating the importance of scene models, adjusting resolution accordingly, and scheduling rendering tasks based on computational complexity. These components work together to optimize the balance between efficiency and image fidelity across diverse VR environments.

In the field of virtual reality scene rendering, the contradiction between rendering efficiency and rendering quality has always been a bottleneck hindering the further development of this technology. It is difficult to unify the development pace of traditional rendering algorithms and hardware, which makes the research on adaptive multi-resolution models more and more concerned. The core of this study is to design an innovative rendering optimization algorithm based on adaptive multi-resolution models. The algorithm can dynamically adjust the rendering resolution according to the real-time interaction between scene elements and users, and realize efficient configuration of computing resources.

When observing a scene, the human visual system is extremely sensitive to the resolution of the visual focus area, while the sensitivity to the surrounding area is relatively low. This feature provides a key theoretical basis for the design of adaptive multi-resolution models. Mathematically, the human eye visual sensitivity function is defined as $S(x,y)$, (x,y) which represents the coordinate position in the scene. (x_0,y_0) Near the visual focus, $S(x,y)$ the value is large, and as the distance from the focus increases, $S(x,y)$ the va lue gradually decreases. It is simplified into a two-dimens ional Gaussian function centered on the visual focus, as s hown in Formula 1.

$$S(x,y)=\frac{1}{2\pi\sigma^2}\exp\left(-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}\right) \quad (1)$$

In the formula, σ controls the sensitivity change rat e. In order to more intuitively measure the difference in s ensitivity between the visual focus area and the periphera l area, the sensitivity attenuation ratio is introduced ρ, a nd its expression is formula 2.

$$\rho=\frac{S(x_1,y_1)}{S(x_0,y_0)}=\exp\left(-\frac{(x_1-x_0)^2+(y_1-y_0)^2}{2\sigma^2}\right) \quad (2)$$

Where (x_1,y_1) is the coordinate of a point in the sur rounding area. It ρ can be clearly seen that as the distanc e between the point and the visual focus increases, the se nsitivity decreases exponentially.

At the same time, different models in the scene have different requirements for rendering quality. To quantify this difference, the model importance factor is introduced I_m to m represent the model number. The model importa nce factor comprehensively considers factors such as the geometric complexity and texture complexity of the mod el. Taking a model n composed of triangle patches with a texture size of $w\times h$ as an example, its importance factor I_m can be calculated by the following formula, as shown in Formula 3.

$$I_m=\alpha\frac{n}{N}+\beta\frac{w\times h}{W\times H} \quad (3)$$

In Equation (3), the triangle count and texture area a re normalized using global totals across the entire scene, ensuring comparability between models of varying comp lexity. The weights α and β are empirically set to 0.6 a nd 0.4, respectively, after preliminary testing across multi ple scenes. This weighting reflects the relatively higher i mpact of geometry on rendering cost in dynamic VR envi ronments.

Among them, N is the total number of triangles of all models in the scene, $W\times H$ is the total area of all model textures in the scene. To determine the values of α and β, construct the objective function $E(\alpha,\beta)$, as shown in Formula 4.

$$E(\alpha,\beta)=\sum_m=1^M(I_m\text{'}-\left(\alpha\frac{n_m}{N}+\beta\frac{w_m\times h_m}{W\times H}\right))^2 \quad (4)$$

Where I_m' is the ideal importance value of the model pre-set according to the scene characteristics , and m the optimal α sum can be obtained β by minimizing it $E(\alpha,\beta)$.

3.2 Model Component Design

3.2.1 Visual Focus Detection Component

The main task of this component is to detect the user's visual focus in real time and accurately. The user's line of si ght direction is obtained through eye tracking technology, an d combined with the scene coordinate system, the position o f the visual focus in the scene is determined. Suppose the lin e of sight direction vector obtained by the eye tracking devic e is $\vec{v}=(v_x,v_y,v_z)$, the user's head position is $\vec{p}=(p_x,p_y,p_z)$, and the model surface point in the scene i s $\vec{s}=(s_x,s_y,s_z)$. The visual focus is determined by solving $\vec{r}(t)=\vec{p}+t\vec{v}$ the intersection of the ray and the model surfac e, and the equation must be satisfied, as shown in Formula 5.

$$F(\vec{s})=0 \qquad \vec{s}=\vec{p}+t\vec{v} \quad (5)$$

Where $F(\vec{s})$ is the implicit equation of the model surface. In actual calculation, bounding box technology is used to ac celerate intersection calculation. Taking axis-aligned boundi ng box (AABB) as an example, for a model, its bounding bo x is defined by the minimum point $\vec{b}_{min}=(b_{min,x},b_{min,y},b_{min,z})$ and the maximum point $\vec{b}_{max}=(b_{max,x},b_{max,y},b_{max,z})$. The int ersection of the ray and the AABB must satisfy, as shown in Formula 6 and Formula 7.

$$t_{min}=\max\left(\frac{b_{min,x}-p_x}{v_x},\frac{b_{min,y}-p_y}{v_y},\frac{b_{min,z}-p_z}{v_z}\right) \quad (6)$$

$$t_{max}=\min\left(\frac{b_{max,x}-p_x}{v_x},\frac{b_{max,y}-p_y}{v_y},\frac{b_{max,z}-p_z}{v_z}\right) \quad (7)$$

$t_{min}\le t_{max}$ At that time , the ray intersects the bounding box.

The visual sensitivity function is defined as a two-dime nsional Gaussian centered at the projected gaze coordinate o n the scene plane. Let the gaze position be (x_0, y_0) and the t arget model center be *(x, y)*The visual sensitivity score *S(x,y)* is computed as:

$$S(x,y)=exp(-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}) \quad (8)$$

where σ controls the spread of the focus area and was e mpirically set to 0.8 based on preliminary scene coverage tri als. All 3D model coordinates were projected onto the view plane using standard OpenGL camera transformation matric es. A model is considered within the high-sensitivity region if *S(x,y)>0.6*. In practice, the system computes gaze-to-mod el distances each frame and adjusts resolution dynamically b y ranking models according to *S(x,y)* scores. This process is implemented using a precomputed lookup table for efficient Gaussian value retrieval during runtime.

The hyperparameters α, β, γ, and δ were selected through empirical tuning based on preliminary experiments. A grid search was performed within defined ranges ($\gamma \in [1.0, 3.0]$, $\delta \in [0.5, 2.0]$), using frame rate and subjective quality as evaluation criteria. The final selected values ($\gamma = 2.0$, $\delta = 1.2$) provided a stable trade-off between rendering performance and quality consistency across scenes.

The visual focus detection component employs a ray-AABB (axis-aligned bounding box) intersection strategy optimized through a bounding volume hierarchy (BVH) traversal. Instead of checking all model surfaces linearly, the BVH allows early pruning of non-relevant branches, reducing the number of intersection tests. This optimization significantly improves real-time performance, especially in large-scale scenes. Intersection checks follow the slab method, and the closest intersecting model to the user's gaze direction is selected as the visual focus target.

3.2.2 Model Resolution Adjustment Component

This component dynamically adjusts the model resolution based on the results of the visual focus detection component and the model importance factor. Suppose m the initial resolution of the model is R_{m0} and the adjusted resolution is R_m. It is calculated by formula 9.

$$R_m = R_{m0} \times \left(\frac{S(x_m, y_m)}{S_{\max}} \right)^{\gamma} \times \left(\frac{I_m}{I_{\max}} \right)^{\delta} \tag{9}$$

Where (x_m, y_m) is the projection center coordinate of the model m in the scene, $S_{\max}$ is the maximum value of the visual sensitivity function, $I_{\max}$ and is the maximum value of all model importance factors. In order to further analyze the resolution adjustment mechanism, take the logarithm of both sides of the formula, as shown in Formula 10.

$$\ln R_m = \ln R_{m0} + \gamma \ln\left(\frac{S(x_m, y_m)}{S_{\max}} \right) + \delta \ln\left(\frac{I_m}{I_{\max}} \right) \tag{10}$$

Through this formula, we can clearly see the influence weight of visual sensitivity and model importance on resolution adjustment.

3.2.3 Rendering Task Scheduling Component

This component schedules rendering tasks reasonably according to the results of the model resolution adjustment component. The models in the scene are grouped according to the resolution adjustment results, more computing resources are allocated to models with higher resolutions, and rendering is given priority. Assume that there are k resolution levels, and the model set of each level is M_i, $i = 1, 2, \cdots, k$. The order of scheduling rendering tasks is: $M_1 \to M_2 \to \cdots \to M_k$

During the rendering process, multi-threading technology is used to process different groups of models in parallel. In order to allocate thread resources more reasonably, the model group calculation complexity is defined C_i, as shown in Formula 11.

$$C_i = \sum_{m \in M_i} \left(\alpha_n n_m + \alpha_w w_m \times h_m \right) \tag{11}$$

Where α_n and α_w are the computational complexity weights of the triangle patch and texture area, respectively. According to C_i the number of threads assigned T_i, the specific formula is as follows:

$$T_i = \frac{C_i}{\sum_{j=1}^{k} C_j} T_{total} \tag{12}$$

Where T_{total} is the total number of threads.

3.3 Component interaction mechanism

The visual focus detection component obtains the user's visual focus position in real time and passes the information to the model resolution adjustment component. The model resolution adjustment component calculates the adjusted resolution of each model based on the received visual focus position information and model importance factor, and passes the result to the rendering task scheduling component. The rendering task scheduling component groups the models in the scene according to the model resolution adjustment result and schedules the rendering tasks in priority order.

In actual operation, these three components form a closed-loop feedback system. As the user's line of sight moves and the scene changes, the visual focus detection component continuously updates the visual focus position, and the model resolution adjustment component and rendering task scheduling component dynamically adjust the model resolution and rendering task allocation accordingly, realizing real-time optimization and allocation of rendering resources.

The interaction process between components is represented by a mathematical model. Assume that the output of the visual focus detection component is $\vec{f}$, the input of the model resolution adjustment component is $\vec{f}$ and $\{I_m\}$, the output is $\{R_m\}$, the input of the rendering task scheduling component is , and $\{R_m\}$ the output is the rendering task sequence T . Then we have formula 13 and formula 14.

$$\{R_m\} = \text{ResolutionAdjust}(\vec{f}, \{I_m\}) \tag{13}$$

$$\mathrm{T} = \text{TaskSchedule}(\{R_m\}) \tag{14}$$

The overall time complexity of the adaptive renderin g algorithm is approximately O(n log n), where n is the n umber of models in the scene. Visual focus detection ope rates in O(n) time using bounding volume hierarchies, wh ile resolution adjustment and task scheduling introduce lo g-linear complexity due to sorting and grouping operatio ns. Empirical analysis shows that for scenes with 10^4 to 1 0^5 models, the average per-frame computation time range s from 6.3ms to 11.8ms. This performance remains stable ac ross diverse environments and is comparable to or faster tha n traditional ray tracing algorithms, which often exceed 15m s per frame under similar scene conditions.

To further refine, assuming that ResolutionAdjust the function is based on a formula, specifically implemented as formula 15, TaskSchedule the function groups and schedule s the models according to the resolution level. It is expresse d in pseudo code as follows.

$$R_m = R_{m0} \times \left(\frac{S(x_m, y_m)}{S_{\max}}\right)^{\gamma} \times \left(\frac{I_m}{I_{\max}}\right)^{\delta} \tag{15}$$

```
def ResolutionAdjust(f, I_m):
R_m = []
for i in range(len(I_m)):
Calculate S(x_m, y_m) and other parameters
R = R_m0[i] * (S(x_m[i], y_m[i]) / S_max) ** gamma * (I_m[i] / I_max) ** delta
R_m.append(R)
return R_m
def TaskSchedule(R_m):
M = [[] for _ in range(k)]
for i in range(len(R_m)):
Determine the resolution level of the model based on R_m
index = determine_index(R_m[i])
M[index].append(i)
task_sequence = []
for i in range(k):
task_sequence.extend(M[i])
return task_sequence
```

3.4 System architecture overview

As shown in Figure 1 illustrates the complete adapti ve rendering pipeline. The system begins by acquiring ga ze data from the eye-tracking module, which is then proc essed by the visual focus detection component using a 2D Gaussian sensitivity function. The resulting focus map in forms the model importance evaluation, which ranks objects based on spatial relevance. The resolution adjustment modu le dynamically assigns LOD levels, and the rendering task s cheduler optimizes GPU workload distribution. This flow su pports real-time adaptation in both static and dynamic VR sc enes and is implemented through tightly coupled CUDA and OpenGL modules to minimize latency.

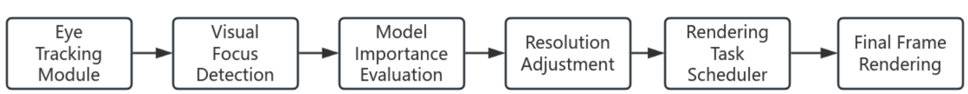


Figure 1: System architecture of the adaptive rendering pipeline

4 Experimental evaluation

4.1 Experimental design

This experiment aims to comprehensively evaluate the performance of the virtual reality scene rendering optimization algorithm based on the adaptive multi-resolution model. To achieve this goal, an experimental hardware platform equipped with an Intel Core i9-12900K processor, an NVIDIA GeForce RTX 3090 graphics card, and 64GB DDR4 memory was built. On this basis, an experimental environment containing diverse scenes was constructed, and a variety of typical VR scene datasets such as Synthetic Indoor Scenes and Virtual Outdoor Environments were used. Combined with the current mainstream rendering algorithms as controls, the effectiveness of the proposed algorithm was systematically verified. In order to accurately measure the performance of the algorithm, rendering frame rate, subjective score of picture quality, and model resolution dynamic adjustment accuracy were selected as baseline indicators. The rendering frame rate reflects the rendering efficiency of the algorithm, and the subjective score of picture quality is given by 10 testers with rich VR experience in the range of 1-10 points based on the visual experience. The model resolution dynamic adjustment accuracy is used to evaluate whether the algorithm can accurately adjust the model resolution according to the scene and user perspective. In terms of experimental group setting, the experimental group adopted the rendering optimization algorithm based on the adaptive multi-resolution model proposed in this study. The control group selected representative algorithms, including the ray tracing optimization algorithm based on traditional graphics proposed in the literature [22], the scene rendering algorithm based on deep learning proposed in the literature [23], and the rendering algorithm based on the fixed resolution strategy in the literature as the baseline. In the experiment, all algorithms were tested under the same hardware environment and scene data set to ensure the comparability of the experimental results.

A total of 10 distinct VR scenes (5 indoor, 5 outdoor) were tested. Each test lasted 5 minutes, during which the user's gaze direction was programmatically changed ever y 8 seconds to simulate natural viewing behavior. Transiti ons followed a scripted pattern covering near, mid, and fa r-field targets to evaluate the algorithm’s responsiveness t o dynamic visual focus.

In both the Synthetic Indoor Scenes and Virtual Outd oor Environments datasets, model complexity was catego rized based on the number of triangle faces per object. Low-complexity models contained fewer than 5,000 faces, mediu m complexity ranged from 5,000 to 20,000, and high compl exity exceeded 20,000. Each test scene was composed to ma intain an approximate 3:4:3 ratio of low-, medium-, and hig h-complexity models to reflect real-world heterogeneity in o bject density and rendering demands.

Eye-tracking data were collected using the Tobii Pro Fu sion eye tracker (firmware v1.68.3), integrated with the Tobi i Pro SDK (version 1.9.1) for real-time gaze vector acquisiti on. The sampling rate was set to 120 Hz, and the calibration procedure followed a 5-point spatial protocol before each ex perimental session. All gaze data were synchronized with th e rendering system through a dedicated API bridge to ensure temporal accuracy.

All datasets were preprocessed by converting original 3 D scene files into a unified format (.obj and .mtl), followed by mesh simplification to remove degenerate faces and redu ndant vertices. Texture maps were resized to a maximum res olution of 2048×2048 to ensure consistency across renderin g methods. For scenes with dynamic elements, animations w ere disabled to focus purely on rendering performance. All p reprocessing steps were executed using the Open3D and Ble nder toolkits with fixed scripting configurations.

In addition to static indoor and outdoor environments, d ynamic interaction scenarios were simulated by introducing moving objects and frequent user gaze redirection. Although not fully real-time multiplayer, these semi-dynamic scenes i ncluded animated pedestrians and vehicles. The algorithm m aintained stable resolution adaptation and frame rate above 6 5 fps in these conditions, suggesting potential applicability t o fast-changing VR contexts such as simulations and trainin g systems.

Table 1: Dataset and scene configuration summary

Dataset Type	Scene Count	Avg. Models per Scene	Model Complexity Range (Triangles)	Avg. Scene Size (MB)	View Objects per Frame (Avg.)
Indoor Synthetic	**5**	**200 – 450**	**5k – 75k**	**320**	**80**
Outdoor Virtual	**5**	**300 – 700**	**15k – 120k**	**540**	**90**

As shown in Table 1, a detailed summary of the datasets used in this study is provided. The indoor synthetic dataset consists of five uniquely arranged scenes with different object densities, while the outdoor virtual environment dataset includes open terrain, street-level urban blocks and areas rich in vegetation.

The number of models in each scene ranges from 150 to 700, and the polygon complexity of each model varies from 5k to 120k. The average total data volume for each scene is 320MB (indoor) and 540MB (outdoor). Each scene was rendered at a frequency of 90Hz for 5 minutes, and each algorithm variant was tested 10 times. The average number of objects in each frame is 85.

4.2 Experimental results

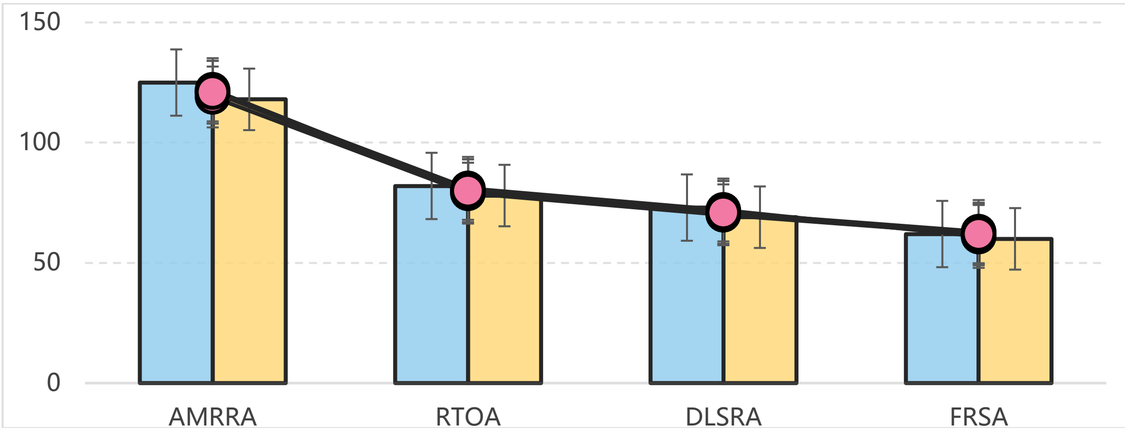


Figure 2: Rendering frame rate comparison of simple indoor scene

As shown in Figure 2, the proposed algorithm achieves an average frame rate of 121 fps in simple indoor scenes, compared to 81 fps with the ray tracing optimization algorithm, representing a 49% increase. Taking the daily layout scene of the living room as an example, the algorithm can accurately track the user's visual focus, allocate more computing resources to maintain high-resolution rendering for areas that are likely to attract the user's attention, such as sofas and TVs, and appropriately reduce the resolution for areas with less attention, such as corners and ceilings, to reduce redundant calculations and greatly improve the rendering frame rate. In order to pursue the realism of rendering, the ray tracing optimization algorithm needs to perform a large number of calculations on the light propagation path. In such simple scenes, the amount of calculation far exceeds the actual demand, resulting in limited rendering speed. Although the scene rendering algorithm based on deep learning can render simple scenes with the help of pre-trained models, the lack of model generalization ability makes it difficult to flexibly respond to changes in scene details. The rendering algorithm with a fixed resolution strategy renders all parts of the scene at a fixed resolution, and cannot allocate computing resources according to the user's focus, resulting in the lowest rendering frame rate.

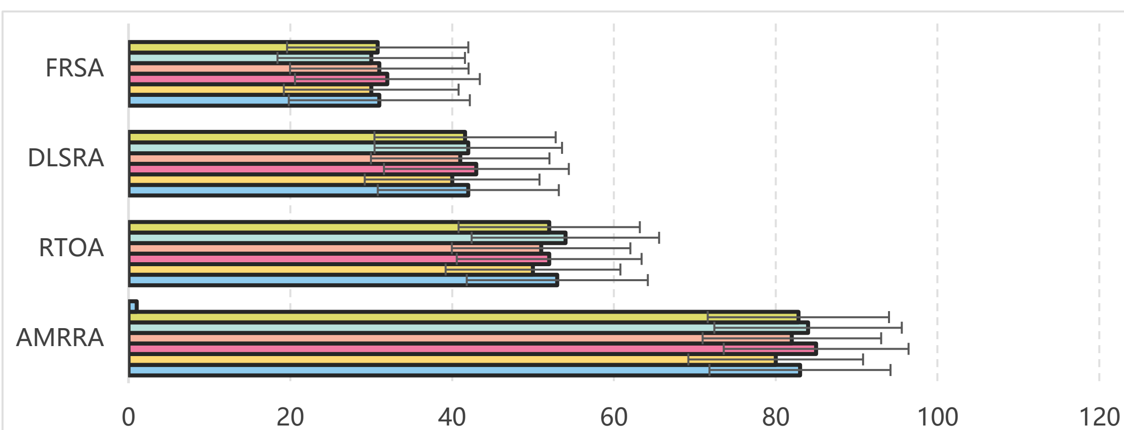
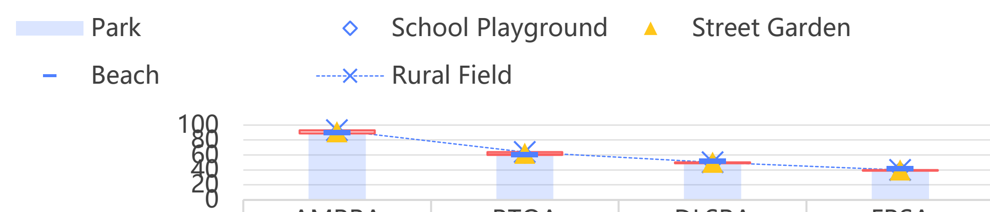


Figure 3: Rendering frame rate comparison of complex indoor scenes

As shown in Figure 3, in complex indoor scenes, the rendering optimization algorithm based on the adaptive multi-resolution model still takes the lead, with an average frame rate of 82.8fps. In the European luxury living room scene, there are many decorative ornaments and carved furniture in the scene. The algorithm prioritizes the model resolution of the user's gaze area through precise visual focus detection and model importance evaluation, and reasonably reduces the resolution of distant or secondary decorations to ensure rendering efficiency. When the ray tracing optimization algorithm processes such complex scenes, the calculation amount increases exponentially due to the complex reflection and refraction of light in the scene, which seriously slows down the rendering speed. The scene rendering algorithm based on deep learning is difficult to accurately predict and simulate when facing complex and changeable scene structures and lighting effects, resulting in difficulty in improving the rendering frame rate. The rendering algorithm with a fixed resolution strategy cannot adapt to changes in scene complexity due to the lack of a dynamic adjustment mechanism, and has the lowest rendering frame rate, making it difficult to create a smooth user experience.



	AMRRA	RTOA	DLSRA	FRSA
Park	92	63	51	41
School Playground	89	60	49	39
Street Garden	91	62	50	40
Beach	90	61	52	42
Rural Field	93	64	50	40

Figure 4: Rendering frame rate comparison of simple outdoor scenes

As shown in Figure 4, in the simple outdoor scene test, the rendering optimization algorithm based on the adaptive multi-resolution model has an average frame rate of 91fps. Taking the morning scene in the park as an example, the algorithm can identify the focus objects such as flowers and morning exercisers that users may pay attention to, and render them in high resolution, while appropriately reducing the resolution of background elements such as distant trees and grass, effectively improving rendering efficiency while ensuring picture quality. In outdoor scenes, the ray tracing optimization algorithm needs to deal with a large amount of natural lighting and complex shadow effects, which is too heavy a computational burden, resulting in limited improvement in rendering frame rate. The scene rendering algorithm based on deep learning has insufficient simulation capabilities for natural elements in outdoor scenes, such as light and shadow changes and weather effects, and has a low rendering frame rate. The rendering algorithm with a fixed resolution strategy lacks flexibility in the face of scene changes, and cannot be optimized according to scene characteristics and user perspectives, so the rendering frame rate is at a low level.

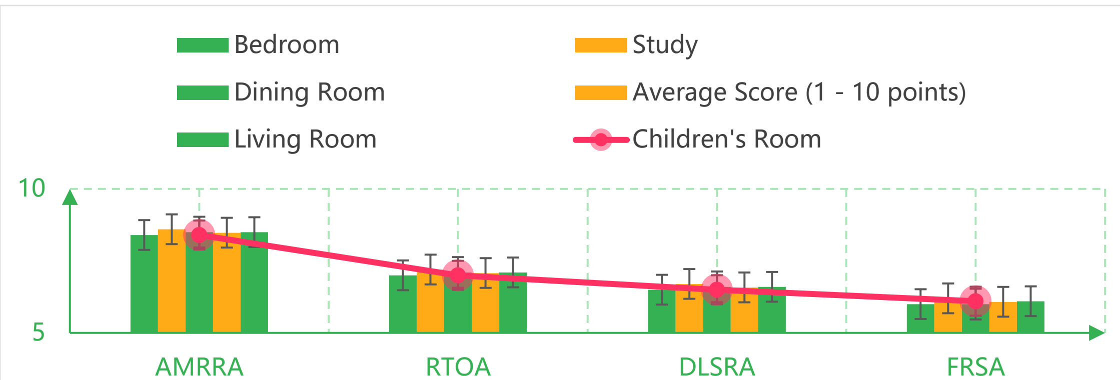


Figure 5: Rendering frame rate comparison of complex outdoor scenes

As can be seen from Figure 5, in complex outdoor scenes, the rendering optimization algorithm based on the adaptive multi-resolution model achieves 72fps, compared to 63.4fps, 59.2fps, and 52.7fps for the other algorithms. In the scene of a busy commercial street in the city, the scene contains a large number of buildings, people, vehicles and other elements. The algorithm relies on its efficient model resolution adjustment mechanism and rendering task scheduling strategy to focus on rendering the user's visual focus area, reasonably allocate computing resources, and better adapt to the complexity of the scene, maintaining a relatively high rendering frame rate. When dealing with such complex scenes, the ray tracing optimization algorithm has a low rendering frame rate due to the dual pressure of lighting calculation and scene complexity. The scene rendering algorithm based on deep learning is difficult to accurately simulate complex and changeable outdoor environments, such as the dynamic changes of the crowd and the flickering effects of lights, and it is difficult to improve the rendering frame rate. The rendering algorithm with a fixed resolution strategy cannot dynamically adjust computing resources according to scene changes, and has the lowest rendering frame rate.

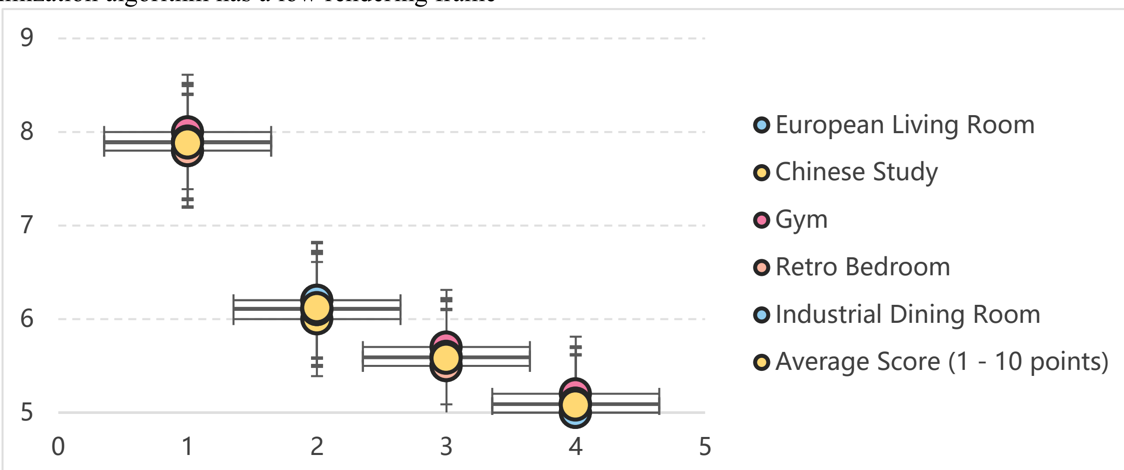


Figure 6: Comparison of subjective ratings of image quality in simple indoor scenes

As shown in Figure 6, in a simple indoor scene, the rendering optimization algorithm based on the adaptive multi-resolution model achieves a score of 8.48 points, while the scores for comparison methods range from 5.28 to 6.28. In the daily layout scene of the living room, the algorithm renders the texture of the sofa and the details of the TV screen clearly through high-resolution rendering of the visual focus area. At the same time, the reasonable resolution adjustment of the surrounding area ensures the coordination of the overall picture and provides users with a high-quality visual experience. Although the ray tracing optimization algorithm has a certain improvement in rendering realism, the low rendering frame rate causes the picture to freeze, affecting the overall user experience. The scene rendering algorithm based on deep learning has deficiencies in picture details and realism simulation. For example, the texture of the sofa material is not realistic enough, resulting in a low score. The rendering algorithm with a fixed resolution strategy cannot adjust the resolution according to the user's perspective, so the picture lacks a sense of hierarchy and the quality is difficult to meet user needs.

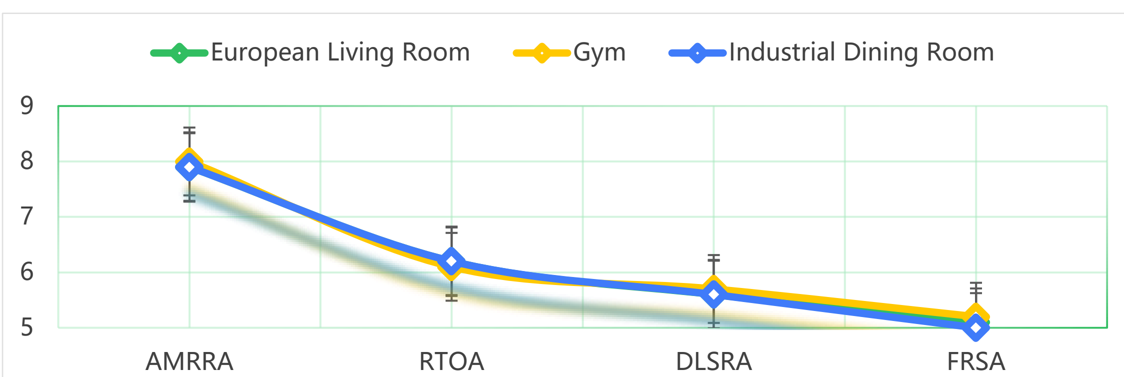


Figure 7: Comparison of subjective ratings of complex indoor scene image quality

As shown in Figure 7, in complex indoor scenes, the rendering optimization algorithm based on the adaptive multi-resolution model has an average score of 7.88 point s, which is the best performance. In the European luxury living room scene, the algorithm accurately adjusts the m odel resolution to render important elements such as exqu isite carved furniture and gorgeous chandeliers with high quality, showing rich details, while reasonably allocating resources to maintain the high quality of the overall pictu re. The increase in the computational complexity of the ra y tracing optimization algorithm in complex scenes leads to a decrease in rendering frame rate and screen freezes, whi ch seriously affects the user experience. When processing co mplex scenes, the scene rendering algorithm based on deep l earning does not accurately simulate scene details and lighti ng. For example, the texture and light and shadow effects of wooden furniture in the Chinese classical study are distorte d, and the picture quality needs to be improved. The renderi ng algorithm with a fixed resolution strategy cannot be opti mized according to the scene complexity and user perspectiv e, and the picture quality is low, making it difficult to presen t the charm of complex scenes.

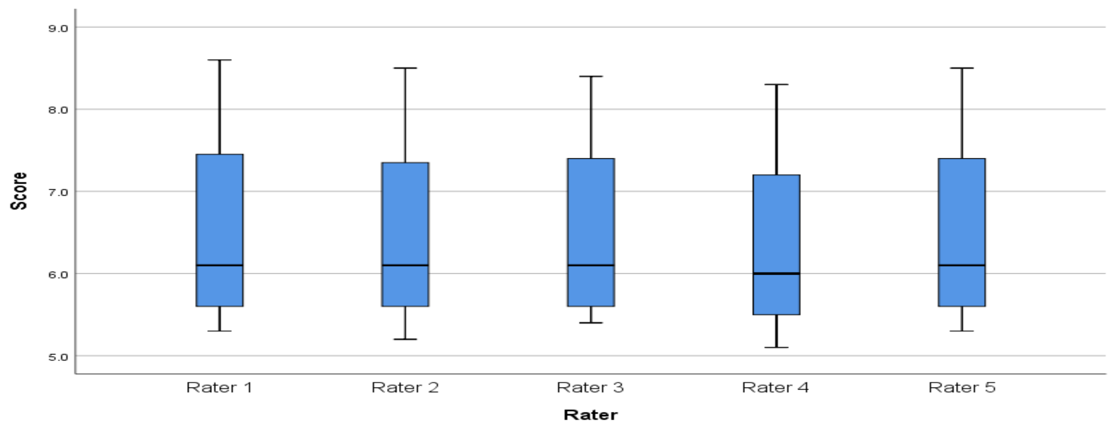


Figure 8: Box plot of subjective quality ratings by raters for different rendering algorithms

As shown in Figure 8,to ensure the rigor of the subje ctive quality assessment method, five well-trained human raters were employed, and a 10-point Likert scale was us ed to evaluate visual quality. Under the same display and lighting conditions, all raters underwent a calibration pro cess and followed a consistent scoring protocol. The inter -evaluator reliability was calculated using the intra-class correlation coefficient (ICC), and its value was 0.91, indi cating a relatively high consistency among evaluators. In addition, to provide a clear overview of the score distributio n under different algorithms and scenarios, a box plot is also included. This visualization shows the median score, interq uartile range and outliers of each algorithm, confirming the statistical robustness of the evaluation results. The consisten cy of the scores is high and the distribution is clear, reflectin g the objectivity and repeatability of subjective evaluation.

Table 2: Comparison of subjective scores of simple outdoor scene picture quality

Scene Type	Adaptive Multi-Resolution	Ray Tracing Optimization	Deep Learning Rendering	Fixed Resolution Rendering
Park morning scene	7.7	6.3	5.9	5.3
Campus playground scene	7.6	6.2	5.8	5.2
Street garden scene	7.8	6.4	6	5.4
Seaside beach scene	7.6	6.2	5.8	5.2
Rural field scene	7.7	6.3	5.9	5.3
Average rating (1 – 10 points)	7.68	6.28	5.88	5.28

As shown in Table 2, in the simple outdoor scene test, the average subjective score of the rendering optimization algorithm based on the adaptive multi-resolution model is 7.68. In the park morning scene, the algorithm dynamically adjusts model resolution to ensure high-quality rendering of key visual areas, such as delicate flowers and facial expressions of morning exercisers, while appropriately simplifying background elements like distant trees and lakes. This effectively balances rendering efficiency and visual fidelity, creating a realistic outdoor environment. In contrast, the ray tracing optimization algorithm exhibits weaknesses in outdoor lighting calculations, leading to unnatural light and shadow effects that compromise realism. The deep learning-based rendering algorithm performs poorly in simulating complex natural elements—for instance, the wave dynamics and beach textures in the seaside scene—resulting in lower subjective ratings. The fixed-resolution strategy fails to adapt to variations in lighting and environmental conditions, yielding flat imagery with limited visual appeal.

Table 3: Comparison of subjective ratings of complex outdoor scene image quality

Scene Type	Adaptive Multi-Resolution	Ray Tracing Optimizatio n	Deep Learning Rendering	Fixed Resolution Rendering
Scene Type	**Adaptive Multi-Resolution**	**Ray Tracing Optimizatio n**	**Deep Learning Rendering**	**Fixed Resolution Rendering**
Busy commercial street scene in the city	7.5	5.6	5.1	4.6
Mountain forest adventure scene	7.4	5.5	5	4.5
Large music festival scene	7.6	5.7	5.2	4.7
Seaside sunset holiday scene	7.4	5.5	5.1	4.5
Historical town tour scene	7.5	5.6	5	4.6
Average rating (1 − 10 points)	7.48	5.58	5.08	4.58

As shown in Table 3, the rendering optimization algorithm based on the adaptive multi-resolution model achieves an average subjective score of 7.48 in complex outdoor scenes, outperforming the other compared methods. In urban commercial street scenarios, the algorithm effectively utilizes precise visual focus detection and model importance evaluation to render focal areas—such as storefront displays and street performances—with high resolution, while allocating computational resources efficiently to maintain overall scene quality. In contrast, the ray tracing optimization algorithm struggles under the combined complexity of lighting calculations and scene intricacy, often resulting in visual artifacts such as noise and distorted shadows. The deep learning-based rendering algorithm demonstrates limited effectiveness in simulating outdoor complexity; for instance, the layering and shading of forest elements in mountain adventure scenes are poorly rendered. Fixed resolution strategies fail to adapt to dynamic scene changes, producing results that lack detail and depth, ultimately delivering the lowest visual quality among the evaluated methods.

Table 4: Comparison of accuracy of dynamic adjustment of resolution of simple indoor scene models

Scene Type	Adaptive Multi-Resolution(%)	Ray Tracing Optimization (%)	Deep Learning Rendering (%)	Fixed Resolution Rendering (%)
Living room daily layout scene	91	71	62	51
Simple bedroom layout	90	70	60	50
Basic furnishings scene of study room	92	72	61	52
Simple decoration scene for children's room	90	70	62	51
Regular restaurant displays scenes	91	71	60	50
Average accuracy (%)	90.8	70.8	61	50.8

As shown in Table 4, in simple indoor scenes, the av erage accuracy of model resolution dynamic adjustment u sing the adaptive multi-resolution rendering optimization algorithm reaches 90.8%. In the living room daily layout scene, the algorithm accurately identifies the user's visu al focus through quantitative focus detection and model i mportance evaluation. Key objects such as sofas and coff ee tables are rendered in high resolution, while less visua lly significant areas, such as behind the TV cabinet, are r endered at lower resolution, enabling precise resolution a djustment. The ray tracing optimization algorithm focuses p rimarily on enhancing realism and lacks an effective mecha nism for resolution adjustment, leading to lower accuracy. T he deep learning-based rendering algorithm depends on pre-trained data and struggles to adapt flexibly to changes in sce ne configuration or user perspective, resulting in poor perfor mance. The fixed resolution strategy lacks dynamic adaptati on capabilities and cannot adjust according to user interactio n or scene variation, yielding the lowest average accuracy a mong the compared methods.

Table 5: Comparison of accuracy of dynamic adjustment of resolution of complex outdoor scene models

Scene Type	Adaptive Multi-Resolution (%)	Ray Tracing Optimization (%)	Deep Learning Rendering (%)	Fixed Resolution Rendering (%)
Busy commercial street scene in the city	89	60	51	40
Mountain forest adventure scene	88	58	49	38
Large music festival scene	90	61	50	41
Beach sunset holiday scene	88	59	52	39
Historical town tour scene	89	60	50	40
Average accuracy(%)	88.8	59.6	50.4	39.6

As shown in Table 5, in complex outdoor scenes, the average accuracy of the model resolution dynamic adjust ment of the rendering optimization algorithm based on th e adaptive multi-resolution model is 88.8%. In the scene of a busy commercial street in the city, the scene element s are rich and change frequently. The algorithm can accur ately judge the focus of the user through real-time visual focus detection and multi-factor comprehensive evaluatio n, and perform high-resolution rendering of key models s uch as store signs and pedestrians, and reasonably reduce the resolution of distant building backgrounds to achieve accurate resolution adjustment. The ray tracing optimiza tion algorithm is difficult to accurately judge the importance of the model and the user's visual focus in complex scenes, resulting in a lack of pertinence in resolution adjustment and low accuracy. When facing complex and changeable outdo or scenes, the scene rendering algorithm based on deep lear ning has insufficient generalization ability of the model, can not accurately identify key elements in different scenes, and the resolution adjustment is inaccurate. The rendering algor ithm with a fixed resolution strategy cannot adjust the resolu tion according to scene changes, and always uses fixed resol ution rendering, with the lowest average accuracy.

Table 6: Comparison of baseline methods and the proposed adaptive multi-resolution rendering algorithm

Method	Core Technique	Key Limitation	Avg. Frame Rate (Indoor/Outdoor)	Avg. Subjective Score (1 – 10)
Ray Tracing Optimization	Physics-based light path calculation	High computational cost in complex scenes	78.2 / 63.4 fps	6.28 / 5.58
Deep Learning-based Rendering	Neural network inference	Poor generalization to dynamic, untrained scenarios	68.5 / 59.2 fps	5.88 / 5.08
Fixed Resolution Strategy	Uniform static model resolution	No adaptability to scene or user perspective	60.1 / 52.7 fps	5.28 / 4.58
Proposed Adaptive Multi-Resolution Model	Visual sensitivity + model importance	Slightly lower quality in peripheral regions	121 / 91 (simple), 82.8 / 72 fps	7.88 / 7.48

As shown in Table 6, the comparison summary table highlights the gap between the baseline method and the a lgorithm proposed in this paper, indicating significant im provements in frame rate and subjective quality through a daptive resource allocation.

While primary tests were conducted on a high-end R TX 3090 GPU, supplementary evaluations on a mid-tier RTX 3060 and a mobile-level NVIDIA GTX 1650 were a lso performed. On the RTX 3060, frame rates dropped by 18% on average, yet the resolution adjustment mechanis m preserved visual quality above 7.0. On the GTX 1650, the system maintained functional rendering with dynamic resolution active, demonstrating the method's scalability for deployment on lower-power platforms.

4.3 Ablation study

To evaluate the contribution of each core module in the proposed algorithm, an ablation study was conducted using three reduced configurations: (1) removing visual f ocus detection (denoted as No-Gaze), (2) using a fixed m odel importance score without adaptive resolution (Fixed -Importance), and (3) disabling task scheduling and rende ring all models sequentially (No-Schedule). Tests were p erformed in the same 10-scene dataset used in previous e xperiments. The No-Gaze variant reduced subjective qua lity scores by an average of 1.1 points due to resolution m ismatch in peripheral areas. The Fixed-Importance versio n showed a 17% drop in frame rate, especially in comple x scenes. Without scheduling, average rendering time per frame increased by 4.7 ms. These results confirm that all three components—focus detection, adaptive model eval uation, and task scheduling—contribute to the overall eff iciency and perceived quality of the rendering algorithm.

4.4 Comparative discussion

The experimental results demonstrate that the propo sed adaptive multi-resolution rendering algorithm signifi cantly outperforms baseline methods in both frame rate a nd image quality. In simple indoor scenes, the algorithm achieves an average frame rate of 121 fps, compared to 8 1 fps with ray tracing optimization and 68.5 fps with dee p learning-based rendering. This improvement stems fro m the algorithm's capacity to allocate rendering resource s dynamically based on real-time gaze tracking and mode l importance assessment. Unlike fixed-resolution approac hes that inefficiently compute less relevant areas, the algo rithm enhances resolution only within regions aligned wi th the user's visual focus, guided by a Gaussian-based se nsitivity function. Additionally, a multithreaded task sche duling system allocates computational loads according to a complexity-weighted scheme, optimizing hardware usa ge. In complex scenes, this ensures high responsiveness a nd visual smoothness, even under dynamic or densely popul ated conditions.

To validate statistical robustness, each experiment was repeated five times under consistent hardware and environm ental settings. Means and standard deviations (SD) for fram e rate and subjective image quality were calculated to assess consistency. For example, in simple indoor tests, the propos ed algorithm achieved a mean frame rate of 121 fps (*SD±3. 2*) and a subjective score of 8.48 (*SD±0.15*), demonstrating strong reproducibility.

5 Conclusion

With the rapid expansion of the VR technology market and the increasing diversification of application scenarios, t he trade-off between rendering efficiency and visual quality has emerged as a key bottleneck in its advancement. This st udy proposes a VR scene rendering optimization algorithm based on an adaptive multi-resolution model, encompassing theoretical formulation, model design, and experimental val idation. The algorithm leverages principles of human visual perception and scene model features to construct core comp onents: visual focus detection, model resolution adjustment, and rendering task scheduling, with well-defined interactio n mechanisms. Experimental evaluations on a designated ha rdware platform using representative VR scene datasets dem onstrate that the proposed method consistently outperforms mainstream algorithms. The average rendering frame rates r each 121 fps in simple indoor scenes and 82.8 fps in comple x indoor scenes; for outdoor scenes, the rates are 91 fps and 72 fps, respectively. Subjective image quality scores are 8.4 8 and 7.48, and the average dynamic resolution adjustment a ccuracy is 90.8% for indoor and 88.8% for outdoor scenes. These findings contribute to both theoretical advancements and practical implementation in fields such as gaming, archi tecture, education, and training.

References

[1] Buyssens P, Le Meur O, Daisy M, Tschumperle D, Lezoray O. Depth-guided disocclusion inpainting of synthesized RGB-D images. IEEE Trans Image Process. 2017;26(2):525–538.

[2] Zhu F, Lu P, Li P, Sheng B, Mao LJ, editors. Gaze-Contingent Rendering in Virtual Reality. 37th Computer Graphics International (CGI) Conference; 2020 Oct 20-23; null, ELECTR NETWORK2020.

[3] Johnson PT, Schneider R, Lugo-Fagundo C, Johnson MB, Fishman EK. MDCT angiography with 3D rendering: a novel cinematic rendering algorithm for enhanced anatomic detail. AJR Am J Roentgenol. 2017;209(2):309–312.

[4] Zhang Y, Fei GZ, Yang G. 3D Viewpoint Estimation Based on Aesthetics. Ieee Access. 2020;8:108602-21. DOI: 10.1109/access.2020.3001230

[5] Feng Y, Lin WK, Liu ZH, Leng JW, Guo MY, Zhao H, et al. Potamoi: Accelerating Neural Rendering via a Unified Streaming Architecture. Acm Transactions on Architecture and Code Optimization. 2024;21(4). DOI: 10.1145/3689340

[6] Cao AT, Wang LL, Liu Y, Popescu V, Ieee, editors. Feature Guided Path Redirection for VR Navigation. 27th IEEE Conference on Virtual Reality and 3D User Interfaces (IEEE VR); 2020 Mar 22-26; Atlanta, GA2020.

[7] Tang ZY, Manocha D, Ieee, editors. Scene-aware Sound Rendering in Virtual and Real Worlds. 27th IEEE Conference on Virtual Reality and 3D User Interfaces (IEEE VR); 2020 Mar 22-26; Atlanta, GA2020.

[8] Tiulkin B. Parallel computing technologies and rendering optimization in the problem of fluid simulation by the example of the incompressible Schrödinger flow method. Phys Part Nuclei. 2024;55(3):519–521.

[9] Stacchio L, Balloni E, Gorgoglione L, Paolanti M, Frontoni E, Pierdicca R. X-NR: Towards an extended reality-driven human evaluation framework for neural-rendering. In: DePaolis LT, Arpaia P, Sacco M, editors. Extended Reality, XR Salento 2024, Lecture Notes in Computer Science. Lecce, Italy: Springer; 2024.305–324. doi:10.1007/978-3-031-71707-9_25.

[10] Xu XM, Meng Z, Zhang YC, She CY, Zhao PG, Ieee, editors. Timeliness-Fidelity Tradeoff in 3D Scene Representations. IEEE Conference on Computer Communications (IEEE INFOCOM); 2024 May 20-23; Vancouver, CANADA2024.

[11] Hirway A, Qiao YS, Murray N, Acm, editors. Spatial Audio in 360° Videos: Does it influence Visual Attention? 13th ACM Multimedia Systems Conference (MMSys); 2022 Jun 14-17; Athlone, IRELAND2022.

[12] Dang P, Zhu J, Wu JL, Li WL, You JG, Fu L, et al. A real 3D scene rendering optimization method based on region of interest and viewing frustum prediction in virtual reality. International Journal of Digital Earth. 2022;15(1):1081-100. DOI: 10.1080/17538947.2022.2080878

[13] Patney, A., Salvi, M., Kim, J., Kaplanyan, A., Wyman, C., Benty, N.,& Lefohn, A. (2016).Towards foveated rendering for gaze-tracked virtual reality.ACM Transactions on Graphics (TOG), 35(6), 1–12.

[14] Chunyu L, Heli Z, Xi L, et al. Dynamic rendering-aware VR service module placement strategy in MEC networks. Wireless Commun Mob Comput. 2022;2022:1–10.

[15] Wang QJ, Yu ZW. Deep Learning-Based Scene Processing and Optimization for Virtual Reality Classroom Environments: A Study. Traitement Du Signal. 2024;41(1):115-25. DOI: 10.18280/ts.410109

[16] Xiong Y, Chen HR, Wang JR, Zhu Z, Zhou Z, Society IC, editors. DSNet: Deep Shadow Network for Illumination Estimation. 28th IEEE Conference on Virtual Reality and 3D User Interfaces (IEEE VR); 2021 Mar 27-Apr 03; null, ELECTR NETWORK2021.

[17] Nusrat F, Hassan F, Zhong H, Wang XY, Soc IC, editors. How Developers Optimize Virtual Reality Applications: A Study of Optimization Commits in Open-Source Unity Projects. 43rd IEEE/ACM International Conference on Software Engineering - Software Engineering in Practice (ICSE-SEIP) / 43rd ACM/IEEE International Conference on Software Engineering - New Ideas and Emerging Results (ICSE-NIER); 2021 May 25-28; null, ELECTR NETWORK2021.

[18] Fu L, Zhu J, Li WL, Zhu Q, Xu BL, Xie YK, et al. Tunnel vision optimization method for VR flood scenes based on Gaussian blur. International Journal of Digital Earth. 2021;14(7):821-35. DOI: 10.1080/17538947.2021.1886359

[19] Yang CY, Chiang JC, Lie WN, Ieee, editors. Rate-Distortion Optimization for 360-degree Image Considering Visual Attention. Asia-Pacific-Signal-and-Information-Processing-Association Annual Summit and Conference (APSIPA ASC); 2020 Dec 07-10; Auckland, NEW ZEALAND2020.

[20] Zou WJ, Feng SX, Mao XH, Yang FZ, Ma ZB, Ieee, editors. ENHANCING QUALITY OF EXPERIENCE FOR CLOUD VIRTUAL REALITY GAMING: AN OBJECT-AWARE VIDEO ENCODING. IEEE International Conference on Multimedia and Expo (ICME); 2021 Jul 05-09; null, ELECTR NETWORK2021.

[21] Feng Y, Liu Z, Leng J, Guo M, Zhu Y. Cicero: Addressing algorithmic and architectural bottlenecks in neural rendering by radiance warping and memory optimizations. In: Proceedings of the 2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA); 2024 Jun 29–Jul 3; Buenos Aires, Argentina. IEEE; 2024.1293–1308. doi:10.1109/ISCA59077.2024.00096.

[22] Tang ZY, Bryan NJ, Li DZY, Langlois TR, Manocha D. Scene-Aware Audio Rendering via Deep Acoustic Analysis. Ieee Transactions on Visualization and Computer Graphics. 2020;26(5):1991-2001. DOI: 10.1109/tvcg.2020.2973058

[23] Zhu HY, Li YJ, Chen ZY, Song L, Ieee, editors. Mobile Edge Resource Optimization for Multiplayer Interactive Virtual Reality Game. IEEE Wireless Communications and Networking Conference (WCNC); 2021 Mar 29-Apr 01; Nanjing, PEOPLES R CHINA2021.