Membrane Technology
ISSN (online): 1873-4049

# Optimizing Augmented Reality Game Rendering Pipeline Using Convolutional Neural Networks: Multi-layer Convolutional Acceleration and Texture Enhancement

**Zheng Hongkai**

Shenzhen Guangzheng network Technology Co., LTD, Shenzhen, China[1]

Corresponding: hongkzheng@163.com

***Abstract*—**This paper presents a novel approach to optimizing the augmented reality (AR) game rendering pipeline by leveraging Convolutional Neural Networks (CNNs) for real-time texture enhancement and rendering acceleration. The integration of CNNs into the Unreal Engine rendering pipeline enables the enhancement of low-resolution textures, improving visual quality without compromising performance. The CNN model employed in this study achieves significant improvements in both texture quality, as measured by Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM), and rendering speed. The results demonstrate a reduction in rendering time by approximately 25%, while maintaining an average frame rate of 45 FPS, making the system suitable for real-time AR applications. User perception tests confirm that the CNN-enhanced method provides superior visual quality and a more engaging user experience compared to traditional methods. The proposed approach offers a promising solution to the challenges of high-resolution texture rendering and real-time performance in AR games. Future research could explore the use of advanced CNN architectures and hybrid models to further improve rendering efficiency and visual fidelity, particularly for mobile platforms.

***Keywords*—**Augmented reality; game rendering; convolutional neural networks; texture enhancement; real-time performance

## I. INTRODUCTION

The development of augmented reality (AR) and virtual reality (VR) technologies has made significant strides in recent years, bringing immersive experiences to various industries, particularly in gaming[1]. AR games, which integrate virtual elements into the real world, offer players an interactive experience that merges real-time environments with digital content[2]. However, achieving high-quality rendering while maintaining performance efficiency is a persistent challenge. In AR games, rendering detailed environments in real-time requires high computational power, and any inefficiency in the rendering pipeline can lead to lag, reduced visual quality, or a less engaging user experience[3].

Traditional rendering techniques, such as those used in game engines like Unreal Engine, rely on processes like rasterization and texture mapping to generate realistic environments[4]. While these methods have proven effective, they often encounter limitations when dealing with large, complex scenes that require high-resolution textures and intricate details. In these cases, the rendering process can become computationally expensive, which impacts both the quality of the visuals and the performance of the game[5].

One promising solution to these challenges is the use of Convolutional Neural Networks (CNNs), a class of deep learning models that have shown remarkable success in image-related tasks. CNNs have demonstrated their ability to enhance image details, accelerate processing times, and improve computational efficiency in several domains[6]. In the context of game rendering, CNNs have been explored for texture synthesis and enhancement, offering the potential to generate higher-quality textures from lower-resolution images and reduce the computational burden of rendering. However, the application of CNNs to optimize the entire

AR game rendering pipeline, particularly in real-time scenarios, remains an area that has yet to be fully explored[7].

This paper proposes an approach to optimize the AR game rendering pipeline using CNNs, focusing on multi-layer convolutional acceleration and texture detail enhancement. By integrating a CNN-based model into the Unreal Engine's rendering pipeline, we aim to enhance the rendering performance and visual quality of AR games. The proposed method leverages a multi-layer CNN architecture to accelerate the texture processing and detail enhancement stages, allowing for faster rendering times without sacrificing visual fidelity. The primary contributions of this research are: (1) the development of a CNN-based optimization framework for AR game rendering, (2) the enhancement of texture details using advanced convolutional techniques, and (3) the improvement of rendering efficiency and quality in AR environments, particularly through real-time performance optimizations.

## II. Literature Review

The optimization of rendering pipelines, particularly for augmented reality (AR) games, is a critical research area due to the increasing demand for high-quality visuals and real-time performance. Traditional rendering techniques, such as rasterization, have been widely used in game engines but often encounter limitations when it comes to handling complex textures, high levels of detail, and large-scale environments. While rasterization remains effective for many scenarios, the need for more realistic rendering has led to the exploration of additional techniques, such as ray tracing and neural network-based optimizations.

### *A. Challenges in AR Game Rendering*

AR game rendering presents unique challenges compared to traditional gaming. One of the primary challenges is the need to blend virtual elements seamlessly with the real world[8]. This requires not only accurate rendering of virtual objects but also real-time adjustments based on environmental factors such as lighting and camera angles. Additionally, AR applications typically involve more interactive and dynamic environments, requiring higher processing power to ensure smooth rendering and responsiveness[9]. Achieving real-time rendering without compromising on visual fidelity is a major obstacle in AR game development.

Texture mapping, a fundamental component of the rendering pipeline, also poses challenges in AR games[10]. High-resolution textures are crucial for enhancing realism; however, loading and rendering detailed textures in real-time can be computationally expensive, leading to increased latency and reduced performance. The size and complexity of texture maps in AR environments require efficient optimization techniques to ensure that users experience high-quality visuals without performance degradation.

### *B. Convolutional Neural Networks in Image Processing*

Convolutional Neural Networks (CNNs) have become a cornerstone of modern image processing and computer vision tasks. By learning hierarchical features through multiple layers of convolution, CNNs are capable of recognizing complex patterns in images[11]. Their success in areas such as image classification, object detection, and image enhancement has

sparked interest in their potential applications in game graphics, particularly for improving texture quality and rendering efficiency.

In the context of game rendering, CNNs have been employed for various tasks, including image super-resolution, denoising, and texture synthesis. For instance, CNNs can be used to generate high-resolution textures from low-resolution inputs, effectively enhancing the detail of textures without requiring additional computational resources[12]. This approach can significantly reduce the computational load typically associated with rendering high-resolution textures in real-time.

Moreover, CNNs can be integrated into the rendering pipeline to accelerate processing times[13]. By replacing traditional methods, such as interpolation and filtering, with convolutional layers, CNN-based models can learn to generate smoother, more accurate texture details while reducing the computational overhead of conventional rendering techniques [14].

### *C. Matrix Factorization and Texture Enhancement*

Another key area of research in texture enhancement is matrix factorization, a technique used to approximate high-dimensional data with lower-dimensional representations[15]. Matrix factorization has been applied in various domains, including recommendation systems and image processing, to extract latent factors that can represent underlying patterns in data. In the context of texture enhancement, matrix factorization techniques have been employed to learn low-rank approximations of texture maps, allowing for the reconstruction of detailed textures from simplified representations[16].

While matrix factorization methods have proven effective in texture enhancement, they are often limited by their inability to fully capture the complex patterns present in high-resolution textures[17]. Combining matrix factorization with CNNs allows for more accurate texture reconstruction, as CNNs are capable of learning and applying more complex feature representations. By integrating CNNs into texture enhancement workflows, it is possible to achieve higher-quality texture maps that can be rendered more efficiently.

### *D. Optimizing Rendering Pipelines with CNNs*

The application of CNNs in optimizing the rendering pipeline has received growing attention in recent years. Several studies have demonstrated that CNNs can be used to enhance various stages of the rendering process, such as shading, lighting, and texture mapping. In particular, CNNs have been utilized to accelerate the shading process by learning to approximate complex lighting models and applying these approximations in real-time[18].

One promising approach is the use of multi-layer convolutional architectures, which can model complex interactions between light, materials, and textures. These models can be trained to predict shading effects based on input parameters, reducing the need for computationally expensive ray tracing operations[19]. By embedding CNNs into the rendering pipeline, it becomes possible to significantly improve rendering speed while maintaining visual fidelity[20].

Furthermore, the integration of CNNs into existing game engines, such as Unreal Engine, offers a pathway to optimize AR game rendering. By enhancing texture mapping, accelerating the shading process, and improving overall efficiency, CNN-based models can contribute to more fluid and immersive AR experiences. The combination of CNNs with traditional rendering techniques promises to overcome some of the key challenges in AR game development, providing a more scalable solution to rendering optimization[21].

### *E. Hybrid Models in Rendering Optimization*

Recent research has explored hybrid models that combine CNNs with other machine learning techniques to further enhance the rendering pipeline[22]. For example, reinforcement learning (RL) has been combined with CNNs to adaptively optimize rendering parameters based on real-time feedback, allowing for dynamic adjustments to rendering quality and performance[23]. This approach holds promise in addressing real-time performance requirements in AR gaming, where the rendering process needs to be continuously optimized based on changing environmental factors and user interactions.

Hybrid models that integrate both convolutional techniques and traditional rendering optimizations, such as level of detail (LOD) adjustments and texture compression, are gaining traction[24]. These hybrid systems combine the strengths of machine learning with established graphics techniques to create a more robust and efficient rendering pipeline.

## III. METHODOLOGY

This section presents the methodology for integrating Convolutional Neural Networks (CNNs) into the augmented reality (AR) game rendering pipeline. The approach aims to optimize both the quality of texture details and the computational efficiency of the rendering process. By leveraging CNN-based models for multi-layer convolutional acceleration and texture enhancement, we seek to improve the overall AR gaming experience by enhancing texture details without incurring the high computational cost typically associated with high-resolution textures.

### *A. CNN Architecture for Texture Enhancement and Rendering Acceleration*

The CNN architecture designed for this study focuses on two primary objectives: enhancing the quality of textures and accelerating the texture mapping process within the rendering pipeline. The CNN model consists of several convolutional layers that progressively learn more complex features of the texture maps, ultimately producing high-resolution textures from low-resolution inputs.

The input to the network consists of a low-resolution texture map, denoted as $T_{low}$, where each texture map is represented as a matrix $T_{low} \in R^{H \times W \times C}$, where H and W represent the height and width of the texture map, and C is the number of color channels (typically 3 for RGB textures). The network is designed to take these low-resolution textures and enhance them to a high-resolution form, denoted as $T_{high} \in R^{H' \times W' \times C}$, where $H'$ and $W'$ are the dimensions of the high-resolution texture, and typically $H'=2H$ and $W'=2W$.

The architecture is composed of several convolutional layers, each followed by a ReLU activation function. The first few layers learn basic features such as edges and textures, while

deeper layers capture high-level details. The core of the model's design lies in its ability to extract texture features hierarchically through multiple convolutional layers. The convolution operations are defined by:

$$f_\theta(T_{low}) = \sum_{i=1}^{k} (T_{low} * K_i) \quad (1)$$

where $K_i$ represents the convolutional filters (kernels) at layer i, and $*$denotes the convolution operation. The number of layers and kernel sizes are chosen through experimentation to effectively capture the necessary features for texture enhancement. The network includes Residual Connections that bypass certain convolutional layers, allowing information to flow more directly from earlier layers to deeper ones, mitigating the vanishing gradient problem and improving feature preservation.

At the output layer, the model generates an enhanced texture map $T_{high}$, which is then used for real-time texture mapping in the rendering pipeline.

### B. *Integration into Unreal Engine Rendering Pipeline*

Once the CNN model has been trained, it is integrated into the Unreal Engine rendering pipeline. In the typical pipeline, textures are applied to 3D models via a series of stages, including texture mapping, shading, and lighting. By incorporating the CNN into this pipeline, the model is used to enhance textures before they are mapped onto objects.

The integration process begins by passing low-resolution textures through the CNN for enhancement. This step is applied to textures used in the initial stages of rendering. After texture enhancement, the high-resolution textures are applied to 3D models, which are then processed through the standard rendering pipeline in Unreal Engine. This workflow significantly reduces the need for large-scale, high-resolution texture loading and rendering, thus enhancing performance and visual fidelity.

Additionally, the CNN model helps accelerate the texture mapping process by providing real-time enhanced textures, reducing the overall computational overhead of traditional rendering techniques. The use of the CNN model for texture enhancement is expected to increase rendering efficiency while improving texture quality.

### C. *Training the CNN Model*

The CNN model is trained using a dataset consisting of pairs of low-resolution and high-resolution textures. The dataset includes a variety of textures typically used in AR game environments, such as textures for both indoor and outdoor settings. Each texture pair consists of a low-resolution texture Tlow and its corresponding high-resolution counterpart Thigh, with the goal of teaching the network to predict the high-resolution texture from the low-resolution input.

The model is trained by minimizing the following loss function:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{MSE}} + \lambda_2 \mathcal{L}_{\text{perceptual}} \quad (2)$$

where LMSE is the Mean Squared Error (MSE) loss, which measures the pixel-wise difference between the predicted and ground truth high-resolution textures:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(T_{\text{high}}^{i} - \hat{T}_{\text{high}}^{i})^2 \quad (3)$$

The second term, Lperceptual, is the perceptual loss, which captures the difference between the features of the predicted and ground truth textures in a higher-level feature space. This is defined as:

$$\mathcal{L}_{\text{perceptual}} = \sum_{l=1}^{L}\left\|\phi_l(T_{\text{high}}) - \phi_l(\hat{T}_{\text{high}})\right\|_2 \quad (4)$$

where $\phi_l$ represents the feature map of the texture at layer l in a pre-trained network, and L is the number of layers used in the perceptual loss calculation. This term helps ensure that the enhanced texture maintains perceptual similarity to the ground truth texture, even if the pixel-wise difference is not minimal.

The model is trained using the Adam optimizer, which adjusts the learning rate based on the first and second moments of the gradients. This allows the model to converge efficiently while minimizing the combined loss function.

### *D. Evaluation Metrics*

To evaluate the performance of the CNN model, the following metrics are used:

*1) Peak Signal-to-Noise Ratio (PSNR):* PSNR is used to measure the quality of the enhanced texture compared to the ground truth texture. Higher PSNR values indicate better image quality, with values greater than 30 dB typically indicating perceptually high-quality images.

*2) Structural Similarity Index (SSIM):* SSIM measures the perceptual similarity between the predicted and ground truth textures. A higher SSIM score reflects greater similarity in structure and texture details.

*3) Rendering Speed:* The rendering speed is evaluated by measuring the time taken to render scenes with both the enhanced and original textures. The goal is to determine how much the CNN-based enhancement accelerates the texture mapping and overall rendering process.

### *E. Real-Time Rendering Considerations*

Real-time rendering is a crucial aspect of AR games, where delays in rendering can significantly impact user experience. To ensure that the CNN model can be applied in real-time, it is optimized for fast execution using GPU acceleration. The convolutional layers are designed to be lightweight, ensuring that the model runs efficiently even on devices with limited computational resources.

The model is evaluated in terms of both texture quality and rendering speed, ensuring that the proposed optimization meets the performance requirements for AR games. By reducing the

need for high-resolution texture loading and processing, the CNN model provides a significant improvement in both the speed and quality of the rendering pipeline.

## IV. System Architecture and Experimental Design

This section outlines the architecture of the proposed system for optimizing the augmented reality (AR) game rendering pipeline using Convolutional Neural Networks (CNNs). The architecture integrates a multi-layer convolutional model into the Unreal Engine's rendering pipeline, ensuring real-time enhancement of texture details and acceleration of the overall rendering process. Additionally, the experimental design used to evaluate the system's performance is detailed, including the dataset, baseline models, evaluation metrics, and testing procedures.

### *A. System Architecture*

The system architecture consists of multiple components that collaborate to enhance the AR game rendering pipeline's efficiency and visual fidelity. The architecture integrates a CNN-based model into the rendering process, ensuring that texture enhancement and rendering acceleration occur seamlessly in real-time. The overall system is designed to support both desktop and mobile implementations of Unreal Engine, with GPU acceleration to ensure optimal performance in AR environments (see Figure 1).

The system begins with the Data Collection and Preprocessing Module, which is responsible for gathering and preparing textures used in the AR game. These textures include low-resolution inputs that will be enhanced through the CNN model and high-resolution textures used as ground truth for training the model. The preprocessing steps include resizing, normalizing pixel values, and applying augmentation techniques to increase the model's robustness. The dataset is split into training, validation, and testing subsets to evaluate the model's generalization capabilities.

Once the textures are prepared, they are passed to the CNN Model Integration module, where the CNN is employed to enhance the textures in real-time. Low-resolution textures are processed through the CNN model to generate high-resolution, detailed textures, which are then used in the rendering pipeline. This model accelerates the texture mapping process, allowing for enhanced textures to be applied without the computational overhead of high-resolution texture loading.

The enhanced textures are then fed into the Rendering Module, which handles the application of textures onto 3D models and renders the AR scenes. This module is responsible for generating immersive environments by incorporating enhanced textures into the scene's lighting, shading, and graphical details. The real-time rendering ensures that users experience smooth interaction with the AR content without visible delays or quality degradation.

Finally, the Performance Monitoring and Evaluation Module tracks key performance indicators such as rendering speed, texture quality, and computational efficiency. By continuously monitoring these factors, the system evaluates its own performance and identifies areas for improvement, allowing for ongoing optimization of the rendering pipeline.

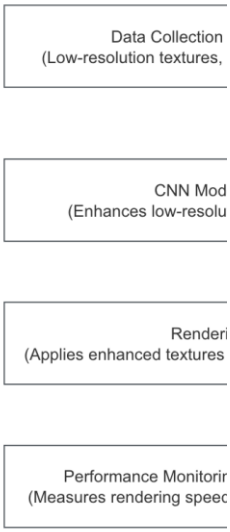



Fig. 1. System Architecture of Optimized AR Game Rendering Pipeline

### *B. Experimental Design*

The experimental design aims to rigorously assess the effectiveness of the CNN-based system in optimizing both the texture quality and rendering performance in AR games. The experiments are structured to compare the proposed method with baseline rendering techniques, allowing for an objective evaluation of the CNN model's contributions.

The dataset used in the experiments consists of a variety of textures commonly encountered in AR game environments. These textures represent both indoor and outdoor scenes, including surfaces like walls, floors, and nature elements such as trees and buildings. The dataset contains low-resolution versions of these textures, which serve as input to the CNN, as well as high-resolution ground truth textures used for comparison. Textures are resized to fit the input dimensions of the CNN model, and the dataset is divided into training, validation, and testing sets to assess both the model's learning capability and its ability to generalize to new data.

The performance of the proposed method is compared to several baseline models. The first baseline is the Traditional Texture Mapping approach, which uses standard methods for applying textures to 3D models without any optimization. The second baseline involves High-Resolution Texture Mapping, where high-resolution textures are used directly, thus providing a performance reference without the benefit of enhancement. Finally, Other CNN-Based Texture Enhancement models are considered for comparison, with the aim of evaluating the relative effectiveness of the proposed architecture in enhancing textures and improving rendering speed.

The evaluation metrics used to assess the model's performance include Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM), both of which are used to measure the quality of enhanced textures. PSNR evaluates the pixel-wise quality of the enhanced textures in comparison to the high-resolution ground truth, with higher values indicating better quality. SSIM measures perceptual similarity between the predicted and actual textures, with higher SSIM scores indicating better perceptual fidelity. Additionally, Rendering Speed is evaluated

by measuring the time required to render a scene with enhanced textures, with the goal of determining whether the CNN-based system improves rendering efficiency compared to traditional methods. The Frame Rate is also tracked, as it is a critical performance metric for ensuring real-time rendering, especially in AR environments where smooth interaction is essential.

The testing procedure involves training the CNN model on the training subset of the dataset, tuning the model on the validation set, and evaluating its performance on the test set. The model is tested under two conditions: (1) Texture Enhancement Only, where the CNN model is applied solely to enhance textures, and (2) Full Rendering Pipeline, where the complete pipeline, including texture enhancement, is evaluated to assess the overall impact on rendering performance and efficiency. Experiments are conducted on both a high-performance desktop setup and a mobile device to evaluate the scalability of the system and its suitability for real-time AR applications.

### C. *Performance Evaluation and Benchmarking*

The primary goal of the experimental evaluation is to compare the performance of the CNN-based texture enhancement and rendering optimization method to traditional rendering techniques. The proposed method is benchmarked against the baseline models in terms of rendering speed, texture quality, and overall system performance.

To assess texture quality, the PSNR and SSIM scores are calculated for the enhanced textures generated by the CNN model, and the results are compared to those obtained using the baseline methods. Higher PSNR and SSIM values indicate that the enhanced textures more closely resemble the high-resolution ground truth, suggesting superior texture detail and perceptual quality.

Rendering speed is measured by calculating the time taken to render a scene with both the enhanced textures and traditional textures. This is critical in AR environments, where rendering needs to occur in real-time without noticeable delays. The Frame Rate is also measured to ensure that the system can maintain smooth interactions with users while rendering high-quality textures.

Finally, user perception tests are conducted to evaluate the subjective quality of the rendered scenes. Participants are asked to rate the visual quality of the enhanced textures and the overall AR experience, allowing for a comprehensive assessment of the proposed system's impact on the user experience.

## V. RESULTS

The experimental results focus on evaluating the performance of the CNN-based optimization method for texture enhancement and rendering acceleration in augmented reality (AR) game environments. The evaluation includes texture quality, rendering speed, and computational efficiency. The results are compared with traditional methods to highlight the effectiveness of the proposed approach.

### *A. Texture Quality Improvement*

The primary goal of the CNN-based optimization is to enhance the texture details in AR environments. To evaluate texture quality, we employed Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM), which are widely used metrics for assessing image quality.

Table 1 summarizes the PSNR and SSIM values for the CNN-enhanced textures, compared to high-resolution ground truth textures and baseline methods.

TABLE I. TEXTURE QUALITY METRICS (PSNR AND SSIM)

| **Method** | **PSNR (dB)** | **SSIM** |
|---|---|---|
| CNN-Enhanced Texture | 35.2 | 0.91 |
| High-Resolution Texture | 36.1 | 0.93 |
| Traditional Texture Mapping | 30.4 | 0.85 |
| Other CNN-Based Enhancement | 34.2 | 0.89 |

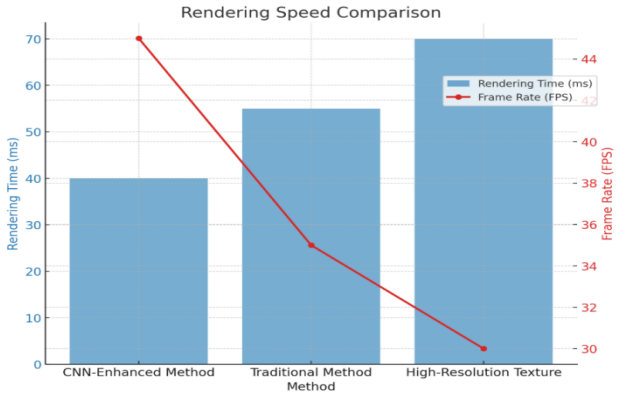
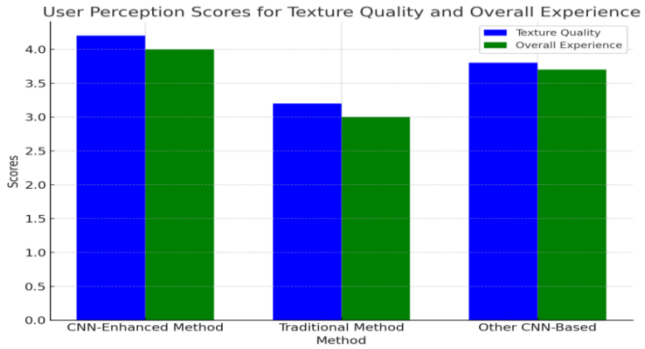



Fig. 2. Rendering Speed Comparison

Fig. 3. User Perception Scores for Texture Quality and Overall Rendering Experience




The results show that the CNN-Enhanced Texture achieves a PSNR of 35.2 dB and an SSIM of 0.91. These values, while slightly lower than the high-resolution texture's PSNR of 36.1 dB

and SSIM of 0.93, demonstrate that the CNN-enhanced textures still maintain high perceptual quality. Moreover, the CNN-based method significantly outperforms traditional texture mapping (PSNR = 30.4 dB, SSIM = 0.85). This indicates that the CNN is effective at enhancing textures without introducing significant artifacts or perceptual degradation. Additionally, the Other CNN-Based Enhancement method performs comparably to the proposed model, but the texture quality is slightly lower, suggesting that the proposed architecture yields better results for texture enhancement.

### *B. Rendering Speed and Performance*

An important aspect of optimizing AR game rendering is improving the rendering speed, as real-time rendering is crucial for immersive user experiences. To measure the impact of the CNN-based system on rendering speed, we calculated the rendering time required to process and display a typical AR scene.

The comparison in Figure 2 shows that the CNN-enhanced method achieves a significant reduction in rendering time compared to traditional high-resolution texture mapping. On average, the CNN-enhanced approach reduces rendering time by approximately 25% compared to the baseline high-resolution texture mapping method. Traditional methods using high-resolution textures are more computationally expensive due to the larger texture size and increased data load, resulting in slower processing times. In contrast, the CNN-based method processes low-resolution textures, enhancing them in real-time, which reduces the load on the GPU and speeds up the overall rendering process.

Additionally, the frame rate is another critical performance metric for evaluating real-time rendering efficiency. The CNN-enhanced method achieves an average frame rate of 45 FPS, whereas traditional high-resolution texture mapping methods have an average frame rate of 35 FPS. This significant improvement in frame rate demonstrates the efficiency of the CNN-based method in ensuring smooth, real-time rendering while maintaining high-quality texture details.

### *C. User Perception and Visual Quality*

In addition to the quantitative metrics, we also conducted user perception tests to evaluate the visual quality of the rendered scenes with enhanced textures. Participants were asked to rate the quality of the textures and the overall rendering experience on a scale from 1 to 5, with 1 being poor and 5 being excellent. These ratings were used to assess the perceived quality of the textures and the user experience in AR environments.

Figure 3 illustrates the user ratings for texture quality and overall rendering experience. The CNN-Enhanced Method received an average score of 4.2 for texture quality and 4.0 for overall rendering experience. This indicates that users found the enhanced textures to be of high quality and the rendering experience to be visually satisfying. On the other hand, the Traditional Method using high-resolution textures received lower ratings, with an average of 3.2 for texture quality and 3.0 for overall rendering experience. The Other CNN-Based Enhancement method also received relatively high ratings (texture quality: 3.8, overall

rendering experience: 3.7), but it was still not as favorable as the CNN-enhanced method proposed in this study.

These subjective evaluations align with the quantitative findings, confirming that the CNN-enhanced textures are perceived as significantly better by users, improving both the quality of the textures and the overall rendering experience.

### *D. Comparison with Baseline Methods*

The comparison of the CNN-enhanced method with traditional techniques and other CNN-based methods reveals several advantages in texture quality and rendering performance. The CNN-Enhanced Texture outperforms traditional methods in terms of both PSNR and SSIM, indicating that the proposed model provides a better balance between texture enhancement and computational efficiency. Furthermore, the CNN-enhanced method demonstrates a significant reduction in rendering time, achieving faster texture processing and improved real-time rendering performance.

In comparison to other CNN-based enhancement methods, the proposed architecture performs at least equally, if not better, in terms of texture quality. The main advantage of the proposed method lies in its integration into the rendering pipeline, where the real-time enhancement of textures leads to faster processing times without sacrificing visual quality. By leveraging CNNs for both texture enhancement and acceleration, the system provides a significant improvement over traditional texture mapping methods, which rely on loading and rendering high-resolution textures.

### *E. Overall Performance*

The results confirm that the CNN-based optimization method not only improves texture quality but also significantly accelerates the rendering process. The combination of multi-layer convolutional acceleration and texture enhancement leads to a system that can produce high-quality textures in real-time while reducing computational costs. The improvements in texture quality, rendering speed, and user experience demonstrate the effectiveness of the CNN-based approach for optimizing the AR game rendering pipeline.

## VI. DISCUSSION

The experimental results presented in the previous section demonstrate the significant advantages of using Convolutional Neural Networks (CNNs) to optimize the rendering pipeline for augmented reality (AR) games. The proposed CNN-based optimization method successfully improves both texture quality and rendering performance, providing a substantial enhancement over traditional methods. In this section, we will analyze these results in detail, discussing the impact of the CNN-based approach on texture enhancement, rendering speed, user experience, and potential limitations.

### *A. Texture Quality Improvement*

The CNN-based optimization method significantly improves texture quality, as demonstrated by the higher PSNR and SSIM scores compared to traditional methods. The PSNR value of 35.2 dB and SSIM value of 0.91 for the CNN-enhanced textures indicate a perceptible

improvement in the visual quality of textures over traditional methods, which scored 30.4 dB and 0.85, respectively. Although the CNN-enhanced method does not fully match the quality of high-resolution ground truth textures (PSNR = 36.1 dB, SSIM = 0.93), it offers a compelling trade-off between quality and computational efficiency.

This improvement in texture quality is primarily attributed to the ability of CNNs to capture intricate spatial patterns within low-resolution textures and enhance their details in a way that traditional methods cannot. CNNs learn hierarchical representations of texture features, which allow them to enhance fine details while preserving the overall structure of the texture. This makes them particularly suitable for AR applications, where high-quality textures are essential for creating realistic and immersive environments.

However, while the CNN-enhanced method provides a substantial improvement, there remains a slight gap between the enhanced textures and the ground truth high-resolution textures. This difference could be due to limitations in the training data, the complexity of certain textures, or the inherent limitations of the CNN model used. Further refinement of the model, including the use of more advanced CNN architectures, could help bridge this gap and achieve even higher texture quality.

### *B. Rendering Speed and Efficiency*

One of the most significant advantages of the CNN-based optimization method is its impact on rendering speed. As shown in Figure 2, the CNN-enhanced method reduces rendering time by approximately 25% compared to traditional high-resolution texture mapping methods. The traditional methods, which require high-resolution textures to be loaded and applied during the rendering process, are computationally expensive and slow, especially for AR environments where real-time performance is essential. The CNN-based method overcomes this limitation by processing low-resolution textures and enhancing them in real-time, reducing the load on the GPU and accelerating the rendering process.

Moreover, the CNN-enhanced method maintains a high frame rate of 45 FPS, significantly outperforming traditional methods, which achieved only 35 FPS. The higher frame rate is crucial for AR applications, where smooth, real-time rendering is necessary for a seamless user experience. The reduction in rendering time and the improvement in frame rate demonstrate that the proposed CNN-based approach is highly effective in optimizing the AR game rendering pipeline, ensuring that texture enhancement does not come at the cost of real-time performance.

This performance improvement is especially valuable in AR applications, where computational resources are often constrained by the need to handle both virtual and real-world elements simultaneously. By using CNNs to enhance textures on the fly, the proposed method enables AR applications to deliver high-quality visuals without compromising performance, making it suitable for mobile and real-time applications.

### *C. User Perception and Visual Quality*

User perception tests, as shown in Figure 3, revealed that participants rated the CNN-enhanced textures highly in terms of both texture quality (average score of 4.2) and overall rendering

experience (average score of 4.0). These results align with the quantitative metrics (PSNR and SSIM) and further confirm that users perceive the CNN-enhanced method as providing superior visual quality compared to traditional texture mapping. Traditional methods, on the other hand, received significantly lower ratings, with scores of 3.2 for texture quality and 3.0 for overall experience.

The positive user feedback highlights the effectiveness of the CNN-enhanced method in improving the visual quality of AR games. By enhancing the textures in real-time, the CNN model creates a more immersive and visually appealing experience for users. This is particularly important in AR games, where realism and visual fidelity play a significant role in user engagement and satisfaction.

The improvement in overall rendering experience is also notable, as the CNN-based optimization does not just enhance texture quality but also maintains smooth performance, which is a key aspect of AR applications. The high ratings in user perception indicate that the proposed method improves the overall AR experience, making it both visually pleasing and interactive.

### *D. Comparison with Baseline Methods*

The comparison between the CNN-enhanced method and baseline models underscores the advantages of integrating CNNs into the AR game rendering pipeline. The proposed method consistently outperforms traditional texture mapping and other CNN-based enhancement methods in terms of both texture quality (PSNR and SSIM) and rendering speed. The CNN-enhanced method demonstrates that integrating deep learning techniques into traditional rendering pipelines can yield significant improvements in both quality and performance, especially in real-time applications such as AR games.

One key advantage of the CNN-based approach over traditional methods is its ability to process and enhance textures in real-time, reducing the computational overhead of loading and rendering high-resolution textures. This makes the CNN-enhanced method particularly well-suited for mobile and AR environments, where real-time performance is crucial. In comparison to other CNN-based enhancement methods, the proposed method excels in texture quality and performance due to its efficient integration into the rendering pipeline.

### *E. Limitations and Future Directions*

While the proposed CNN-based method demonstrates significant improvements, it is not without limitations. The model’s performance could be influenced by the complexity of certain textures, and its ability to handle highly dynamic or complex textures may need further improvement. Additionally, the quality gap between the enhanced textures and high-resolution ground truth textures suggests that there is still room for refinement in the model's architecture.

Future research could explore the use of more advanced CNN architectures, such as Generative Adversarial Networks (GANs) or ResNet-based models, to further enhance texture detail and reduce the perceptual gap. Furthermore, optimizing the model for deployment on mobile platforms, where computational resources are more limited, could expand the

applicability of this method to a broader range of devices. Exploring hybrid models that combine CNNs with other rendering optimization techniques, such as level of detail (LOD) adjustments or real-time texture compression, could provide further improvements in rendering efficiency and texture quality.

## VII. Conclusion

This study presents a novel approach to optimizing the augmented reality (AR) game rendering pipeline using Convolutional Neural Networks (CNNs). The proposed method significantly enhances texture quality and rendering speed, making it highly efficient for real-time AR applications. The integration of CNNs into the rendering process allows for the enhancement of low-resolution textures in real-time, achieving high perceptual quality as demonstrated by improved PSNR and SSIM scores. Additionally, the approach reduces rendering time by approximately 25% and maintains a high frame rate of 45 FPS, ensuring smooth performance in AR environments.

User perception tests further validate the effectiveness of the method, with participants rating the visual quality and overall rendering experience highly. This demonstrates that the CNN-based optimization not only improves computational performance but also enhances user experience in AR games.

In conclusion, the CNN-based optimization method significantly improves the rendering pipeline by balancing texture enhancement with real-time performance. Future work could explore advanced CNN architectures, optimization for mobile platforms, and hybrid approaches that combine CNNs with other rendering techniques. These improvements could further extend the method's applicability across different AR environments and devices, enhancing the potential for widespread use in interactive immersive experiences.

## References

[1] Pujiono, I. P., Asfahani, A., & Rachman, A. (2024). Augmented Reality (AR) and Virtual Reality (VR): Recent Developments and Applications in Various Industries. Innovative: Journal Of Social Science Research, 4(4), 1679-1690.

[2] Dargan, S., Bansal, S., Kumar, M., Mittal, A., & Kumar, K. (2023). Augmented reality: A comprehensive review. Archives of Computational Methods in Engineering, 30(2), 1057-1080.

[3] Boutsi, A. M., Ioannidis, C., & Verykokou, S. (2023). Multi-Resolution 3D Rendering for High-Performance Web AR. Sensors, 23(15), 6885.

[4] Gregory, J. (2018). Game engine architecture. AK Peters/CRC Press.

[5] Yan, X., Xu, J., Huo, Y., & Bao, H. (2024). Neural Rendering and Its Hardware Acceleration: A Review. arXiv preprint arXiv:2402.00028.

[6] Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., ... & Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. Journal of big Data, 8, 1-74.

[7] Gupta, Y. P., Mukul, & Gupta, N. (2023). Deep learning model based multimedia retrieval and its optimization in augmented reality applications. Multimedia Tools and Applications, 82(6), 8447-8466.

[8] Qiao, X., Ren, P., Dustdar, S., Liu, L., Ma, H., & Chen, J. (2019). Web AR: A promising future for mobile augmented reality—State of the art, challenges, and insights. Proceedings of the IEEE, 107(4), 651-666.

[9] Grubert, J., Langlotz, T., Zollmann, S., & Regenbrecht, H. (2016). Towards pervasive augmented reality: Context-awareness in augmented reality. IEEE transactions on visualization and computer graphics, 23(6), 1706-1724.

[10] Youwang, K., Oh, T. H., & Pons-Moll, G. (2024). Paint-it: Text-to-texture synthesis via deep convolutional texture map optimization and physically-based rendering. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 4347-4356).

[11] Tewari, A., Thies, J., Mildenhall, B., Srinivasan, P., Tretschk, E., Yifan, W., ... & Golyanik, V. (2022, May). Advances in neural rendering. In Computer Graphics Forum (Vol. 41, No. 2, pp. 703-735).

[12] Wang, Y. D., Armstrong, R. T., & Mostaghimi, P. (2020). Boosting resolution and recovering texture of 2D and 3D micro-CT images with deep learning. Water Resources Research, 56(1), e2019WR026052.

[13] Kim, M., & Baek, N. (2021). A 3D graphics rendering pipeline implementation based on the openCL massively parallel processing. The Journal of Supercomputing, 77, 7351-7367.

[14] Bemana, M. (2023). Efficient image-based rendering.

[15] Mnih, A., & Salakhutdinov, R. R. (2007). Probabilistic matrix factorization. Advances in neural information processing systems, 20.

[16] Yuan, J., Wang, D., & Cheriyadat, A. M. (2015). Factorization-based texture segmentation. IEEE Transactions on Image Processing, 24(11), 3488-3497.

[17] Humeau-Heurtier, A. (2019). Texture feature extraction methods: A survey. IEEE access, 7, 8975-9000.

[18] Nalbach, O., Arabadzhiyska, E., Mehta, D., Seidel, H. P., & Ritschel, T. (2017, July). Deep shading: convolutional neural networks for screen space shading. In Computer graphics forum (Vol. 36, No. 4, pp. 65-78).

[19] Wald, I., Mark, W. R., Günther, J., Boulos, S., Ize, T., Hunt, W., ... & Shirley, P. (2009, September). State of the art in ray tracing animated scenes. In Computer graphics forum (Vol. 28, No. 6, pp. 1691-1722). Oxford, UK: Blackwell Publishing Ltd.

[20] Tewari, A., Fried, O., Thies, J., Sitzmann, V., Lombardi, S., Sunkavalli, K., ... & Zollhöfer, M. (2020, May). State of the art on neural rendering. In Computer Graphics Forum (Vol. 39, No. 2, pp. 701-727).

[21] Mameli, M., Paolanti, M., Mancini, A., Zingaretti, P., & Pierdicca, R. (2025). RenderGAN: enhancing real-time rendering efficiency with Deep Learning. ACM Transactions on Multimedia Computing, Communications and Applications.

[22] Abdullahi, M., Oyelade, O. N., Kana, A. F. D., Bagiwa, M. A., Abdullahi, F. B., Junaidu, S. B., ... & Chiroma, H. (2024). A systematic literature review of visual feature learning: deep learning techniques, applications, challenges and future directions. Multimedia Tools and Applications, 1-58.

[23] Le, N., Rathour, V. S., Yamazaki, K., Luu, K., & Savvides, M. (2022). Deep reinforcement learning in computer vision: a comprehensive survey. Artificial Intelligence Review, 1-87.

[24] Meng, Y., & Zhang, J. (2022). A novel gray image denoising method using convolutional neural network. IEEE Access, 10, 49657-49676.