Achieving Lightweight Super-Resolution for Real-Time Computer Graphics

Yu Wen[1*], **Chen Zhang**[1*], **Chenhao Xie** [2†], **Xin Fu**[1]

[1]University of Houston,
[2]Beihang University
{ywen2, zchen50, xfu8}@central.uh.edu, xiechenhao@buaa.edu.cn

Abstract

Image super-resolution (SR) is essential for bridging the gap between modern hardware and real-time computer graphics (CG) applications. It reduces CG workload by allowing low-resolution rendering, with original quality restored later via mathematical operations or machine learning. However, recent learning-based SR methods often rely on complex models, demanding high computational resources and undermining the benefits of reduced rendering workload.

Our qualitative and quantitative analysis of the SR process and rendering reveals that readily accessible rendering information can significantly enhance neural network design by serving as additional features. To capitalize on this, we propose CGSR, an optimization framework designed for lightweight real-time super-resolution. CGSR utilizes rendering information to boost both network extensibility and efficiency. It utilizes progressively available rendering information from the pipeline, which arrives earlier than the rendered frame, enabling pre-processing and masking of latency. These features are then integrated into a selected SR network backbone to form a CG-enhanced network. This network is further optimized and refined into a CG-optimized version using neural architecture search (NAS). To improve runtime performance, CGSR also employs rendering-aware hybrid pruning, which dynamically prunes the network based on temporal rendering data. Evaluation results show that CGSR significantly reduces parameter size, multi-add operations, and inference time while maintaining high SR quality across various backbone SR networks.

Project Page —
https://github.com/UH-ECOMS-Lab/CGSR

Introduction

Image super-resolution (SR) focuses on enhancing the quality of an image when upscaling from low resolution to high resolution. The technique has gained popularity in various fields, including medical, media sharing, and surveillance (Mahapatra, Bozorgtabar, and Garnavi 2019; Zhang et al. 2010), as it may restore a high-quality image from different degradation origins such as compression, sub-sampling, and low-resolution rendering.

In the context of real-time computer graphics (CG) and metaverse, the image SR technique fully harnesses its strengths to support the 3D rendering process by reducing the high computational demand introduced by the increased display resolution and the photo-realistic effects demand. For example, the modern virtual reality (VR) system not only requires high-resolution rendering for both right and left eyes but also demands a high framerate to compensate for the famous screen door effect (Wikipedia 2024) and reduce the simulation sickness (LaViola Jr 2000; Kolasinski 1995). The newest VR headsets, like the Meta Oculus Quest 3, already raise the bar to 2064 x 2208 resolution and up to 120Hz framerate (Meta 2023), resulting in extremely high computational overhead. By introducing the SR techniques, the real-time CG workload can be performed at a lower resolution to reduce the overhead, and a later SR process can restore the quality of the rendered frame.

Motivated by the requirements, the SR techniques have developed rapidly in the past decades from the simplest interpolation to filter-based approaches. In recent years, learning-based approaches become the mainstream of the SR, as such approaches successfully fill the high-frequency details missing in the low-resolution input images (Ha et al. 2018; Li et al. 2020; Yang et al. 2019b). To satisfy the photo-realistic quality requirement of the emerging 3D games and metaverse applications, recent ML-based methods focus on how to further improve the SR quality and reduce the errors introduced during the SR process such as the over-smoothed edges and screen flickers. Therefore, several novel techniques were introduced, including residual networks (Kim, Lee, and Lee 2016; Lim et al. 2017; Yu et al. 2018; Ahn, Kang, and Sohn 2018) and pixel attention design (Zhao et al. 2020). However, although these ML-based SR methods can overcome certain quality loss issues, the increasing model size results in extra workload and latency of the SR process which weakens the performance improvement and limits their usage in real-time CG applications.

In this paper, we present CGSR, an optimization framework for lightweight SR that leverages real-time rendering information to enhance the SR process. CGSR significantly reduces parameter size and multi-add operations while preserving or even improving SR quality compared to unopti-

[*]These authors contributed equally.
[†]Corresponding Author.

mized SR backbones. Our method utilizes information such as depth, normal directions, and object edges from the rendering pipeline. This information, which is available progressively before the final frame is rendered, allows for preprocessing without adding latency. We integrate these features into various SR backbones through an additional information transform block, avoiding the need for full network retraining. An evolutionary neural architecture search (NAS) guided by geometry properties then performs parameter reduction, optimizing both network architecture and parameters efficiently. Finally, CGSR employs rendering-aware hybrid pruning, using temporal information from previous and current frames to identify and prune less important kernel weights while maintaining SR quality. Through these strategies, CGSR achieves up to a 77% reduction in parameter size and performs only 22% of the multi-add operations compared to original SR backbones. The SR quality of CGSR is maintained or improved, with no visible degradation. Additionally, CGSR significantly reduces inference time across various SR backbones, making it ideal for real-time CG and metaverse applications. Our main technical contributions are summarized as follows:

- We observe that the quality error of SR is closely related to a lack of recognition of rendering information, which can be extracted from the rendering pipeline.
- We integrate the rendering information into various SR backbones through an information transform block with minimal overhead. Additionally, we use an evolutionary-based NAS approach, guided by rendering information, to optimize these SR backbones to meet the lightweight requirements of real-time CG applications.
- To further leverage rendering information during runtime, we propose a rendering-aware hybrid pruning method that prunes unimportant weights on the fly to accommodate the continuously changing content.
- Our case study demonstrates that CGSR reduces parameter size to 23% and multi-add operations to 22% of those in state-of-the-art networks. Additionally, CGSR significantly reduces inference time across many SR backbones, highlighting its effectiveness in optimizing performance across diverse network architectures.

Related Works

Based on the main techniques, the SR could be classified into 2 main groups: the filter-based methods, and the learning-based methods. The filtering-based methods utilize different mathematics methods to generate high-resolution pixels, while the learning-based methods analyze the whole image or video for specific features and add the high-resolution details accordingly.

Filter-based Super-resolution

A straightforward method to generate a high-resolution image is performing an interpolation to fill up the color of the new pixels. Although the interpolation methods can be very fast, the simple interpolation computation usually produces visible errors that significantly degrade the SR quality. For example, the nearest interpolation is prone to generate jagged edges, while the bilinear and cubic interpolation tend to blur the edges and lose high-frequency details (Thévenaz, Blu, and Unser 2000; Han 2013; Siu and Hung 2012).

To overcome the drawbacks of interpolation without overcomplicating the process, several filtering-based methods were proposed during the past decades, including the Wiener filter-based (Hardie 2007; Cruz et al. 2017; Hardie and Barnard 2012) and Kalman filter-based (Takehara et al. 2000; Kanakaraj, Nair, and Kalady 2017; Rahimi et al. 2019) SR. Recently, the FidelityFX Super Resolution (FSR) (AMD 2024) introduced by AMD achieves a high framerate with acceptable quality. Generally, due to the simplicity of the algorithms, the filtering-based SR methods ensure a good performance with acceptable quality loss. However, such methods often introduce visible quality loss such as screen flickers and ghosting, and also struggle to handle the high-dynamic or foggy scenes. Thus, the filter-based approaches need to be further combined with other post-processing to achieve a highly realistic result.

CNN-based Super-resolution

Unlike the filter-based SR, the learning-based SR can achieve high potential quality as the ML process may add more details that are absent from the low-resolution version. The early SRCNN (Dong et al. 2015) designed a simple 3-layer convolutional neural network to perform the SR computation. Many machine-learning-based methods have been proposed in recent years based on different network designs. VDSR (Kim, Lee, and Lee 2016) learns the residual between the low and high-resolution input for better accuracy. EDSR (Lim et al. 2017) also utilizes a residual network for this task. And PAN (Zhao et al. 2020) introduced pixel attention to reduce the parameter size of the network further. Recent methods leverage temporal and auxiliary rendering information to enhance real-time rendering. Neural Supersampling (Xiao et al. 2020) integrates features from previous frames, depth, and motion vectors to optimize super-resolution. Mob-FGSR (Yang et al. 2024) uses motion vectors and depth maps to achieve accurate frame generation and super-resolution through splat-based motion estimation. Extranet (Guo et al. 2021) uses geometry buffers and motion vectors to generate accurate, ghosting-free extrapolated frames, enabling low-latency temporal supersampling.

Recently, the NTIRE challenge (Lugmayr, Danelljan, and Timofte 2021; Lugmayr et al. 2022; Li et al. 2023) on super-resolution has introduced various innovative approaches focusing on both performance and efficiency. In addition, RepSR (Wang, Dong, and Shan 2022) introduces a novel VGG-style network using structure re-parameterizable block and batch normalization to boost super-resolution performance. While these methods share a focus on efficiency, they mainly explore specific network structures tailored for single-image super-resolution on real-world datasets. In contrast, our approach leverages rendering information with detailed geometric features to improve CG image resolution and address the less-explored spatial domain of computer graphics for real-time applications, providing an orthogonal optimization to previous work.

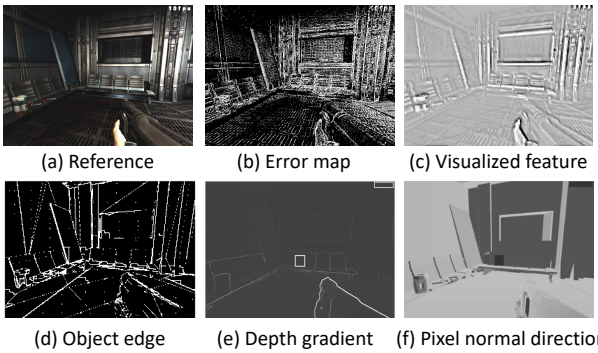


Figure 1: Analysis of the limitation of SOTA SR and the relationship between rendering information and SR error.

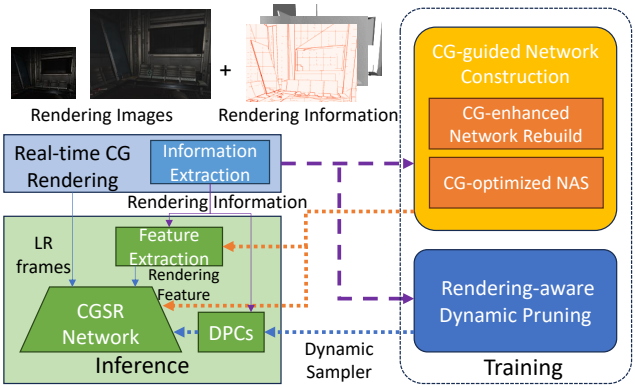


Figure 2: The overall workflow of the Lightweight CG Guided Real-time Super Resolution (CGSR) design. DPCs: dynamic pruning components.

Motivation

Although recent SR networks have significantly enhanced image quality, their increasing complexity in extracting 3D spatial and geometric features from input images leads to excessive parameters and computations. However, real-time CG applications require not only exceptional image quality but also high performance. As a result, this added complexity translates into longer processing times, making these SR networks impractical for real-time use.

In real-time CG applications, rendering converts 3D models into 2D images that are then fed into SR networks, making these processes inherently coupled. Therefore, designing real-time SR networks should be closely integrated with the rendering process. To achieve high-quality, low-latency SR in real-time CG, it is essential to consider both SR and rendering together.

To investigate the relationship between SR and rendering, we start with error analysis and feature visualization experiments. Error analysis reveals how the SR network performs across different regions of a rendered frame, pinpointing areas where the network struggles. This analysis enables us to trace back to specific elements of the rendering process associated with these challenging areas. Feature visualization then provides a direct comparison of how effectively the SR network interprets and processes different scene elements, allowing a clear assessment of its understanding relative to the original rendering information. Our error analysis reveals an uneven distribution of SR errors (Fig.1-(b)), with pronounced errors concentrated around specific regions. This pattern suggests that the network struggles to capture and process detailed spatial and geometric features. To investigate further, feature visualization (Fig.1-(c)) provides additional insight into these spatial and geometric details. The visualization confirms that the network's tendency to process input images uniformly across all regions exacerbates errors in areas with complex spatial and geometric features. This indicates that the network's resource allocation is inadequate for handling these challenging regions effectively.

Building on our initial findings, we further analyzed the decomposed rendering pipeline and found that several specific types of inherent spatial and geometric information align with the observed error distribution. This includes normal vectors (Fig.1-(f)), which are computed during the geometry stage to assist with light interaction; depth information (Fig.1-(e)), stored in the depth buffer to manage fragment visibility; and object edges (Fig.1-(d)), derived from depth data (Chen et al. 2014; Schmeing and Jiang 2014; Abid Hasan and Ko 2016; Wen et al. 2023). These data points are naturally generated and utilized throughout the rendering process. As this information is available earlier in the rendering pipeline and stored in GPU cache or memory, it can be directly accessed and integrated into the SR process, effectively masking latency by overlapping it with other pipeline stages. By utilizing this readily available data, we can enhance the efficiency and effectiveness of SR, providing valuable insights into the process while maintaining practical applicability.

Method

Framework Overview

To achieve efficient super-resolution, we propose an optimization framework for lightweight SR called "CGSR" that couples the SR network with rendering information generated during the runtime of real-time CG applications. The CGSR framework is adaptable to various CNN-based SR network backbones, with each selected SR network following a feature extraction-reconstruction paradigm. This approach demonstrates high generalization and versatility, offering a universal optimization solution for SR in the real-time CG domain. The CGSR framework consists of two main phases: CG-guided NAS network construction and rendering-aware hybrid pruning, as illustrated in Fig. 2. We focus primarily on object edges, normal directions, and depth maps, as these elements strongly correlate with SR errors, as discussed in the previous section.

In the first phase, we integrate the rendering information into a pre-trained SR network through an additional rendering information transform block, avoiding the need for full network retraining. This integration constructs a

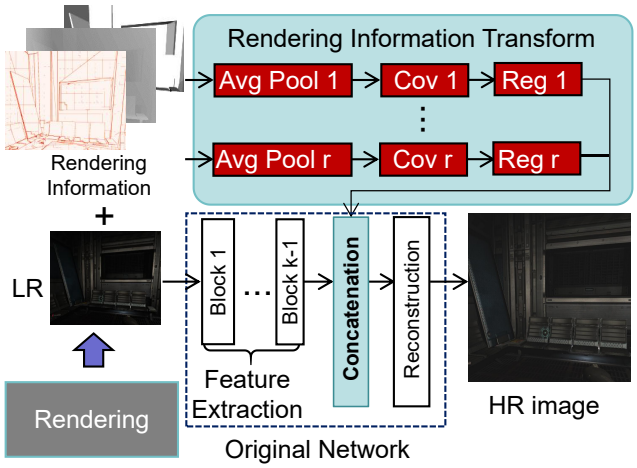


Figure 3: The structure of the rendering information transform block.

CG-enhanced network, which we then refine into the CG-optimized Network using neural architecture search (NAS) for parameter and channel shrinking. The NAS process selects the most efficient configuration based on the integrated feature, achieving co-optimization of both architecture and parameters. The second phase involves rendering-aware hybrid pruning. Using real-time 3D rendering information from CG applications, we first train a dynamic pruning component to identify rendering patterns and prune unimportant kernel weights. This is followed by a fast static pruning process to achieve high pruning rates and maintain SR quality.

CG-guided NAS Network Construction

Based on our discussion in motivation, rendering information is crucial to the SR process as it provides essential spatial and structural details that impact both quality and efficiency. However, incorporating this information directly as inputs requires the network to learn the details from scratch, which can be error-prone and complex. To address this, we propose using rendering information as an intermediate feature. This approach enhances efficiency and accuracy by allowing for precise corrections with ground truth data at critical stages of processing.

Specifically, we introduce a rendering information transform block that utilizes pooling, convolution, and regularization layers to convert rendering information into usable features. These features are resized and spatially aligned to seamlessly integrate with the network's feature maps. In typical SR neural networks, the feature extraction phase processes low-resolution images to extract meaningful intermediate features, while the reconstruction phase uses these features to generate high-resolution images. We position the processed rendering features between the feature extraction and reconstruction phases, ensuring they effectively guide the transformation process. This placement optimizes the transition from low-resolution to high-resolution images, resulting in a CG-enhanced network. The overall structure of this feature mapping block is illustrated in Fig. 3.

To further optimize the architecture and parameters of the selected backbone SR network, we apply NAS to refine the CG-enhanced network after integration. The search space is designed to explore various SR block configurations, allowing for channel and layer combinations that meet real-time CG requirements. During the NAS process, the rendering information is used as an additional loss term. Specifically, we evaluate each subnet by calculating a combined loss that includes two components: one for the transformation of low-resolution images to high-resolution images, and another for transforming low-resolution geometry properties to high-resolution counterparts. This combined loss guides the subnet selection to optimize both image quality and geometric accuracy. Thus, we approach NAS as a co-optimization task, aiming to minimize both SR quality loss and geometry properties loss while ensuring validation accuracy. We use an evolutionary search algorithm (Vikhar 2016; Slowik and Kwasnicka 2020; Zhang et al. 2011) to generate P new candidates via mutation and crossover. Each candidate undergoes a few epochs of training with a warm start from the original network using the following loss function:

$$L_t = ||HR_{i,r} - SR_{i,r}||_2 + \sum_{i=1}^{r} ||R_i|| + \sum_{k=1}^{K} ||S_k||, \quad (1)$$

where HR_i denotes the high-resolution image ground truth, SR_i represents the original image. HR_r and SR_r are the high and low-resolution rendering information, respectively. In practice, the loss for rendering information may be used with an additional weight to balance its impact relative to the image quality loss. R_i denotes the regularization layer for the feature conversion channel and S_k represents the channels to be shrunk in the original network. The iterative NAS process selects the H most efficient candidates with minimal parameter size and minimal combined image and rendering information loss, where H is a ratio of P. The process continues until the number of candidates is below a pre-defined threshold, with the smallest parameter size candidate being chosen as the CG-optimized SR Network.

Rendering-Aware Dynamic Pruning

While offline NAS has proven highly effective, further improvements in SR efficiency can be achieved through dynamic, pattern-based pruning guided by real-time rendering information. Not all pixels in a frame contribute equally to SR; for example, only a small percentage may contain critical details like object edges (Fig.1-(d)). Online rendering information can highlight these key areas, indicating which corresponding weights are most crucial for pruning. Unlike traditional pruning methods that rely solely on weight magnitude (Li et al. 2016; Lee et al. 2020; Li et al. 2018), we introduce rendering-aware dynamic pruning. This method incorporates the specific rendering characteristics of CG applications, efficiently pruning weights based on current and previous rendering information, ensuring the network adapts to evolving content in real time.

The key to achieving dynamic pruning lies in determining the importance of each weight by regularizing it with ren-

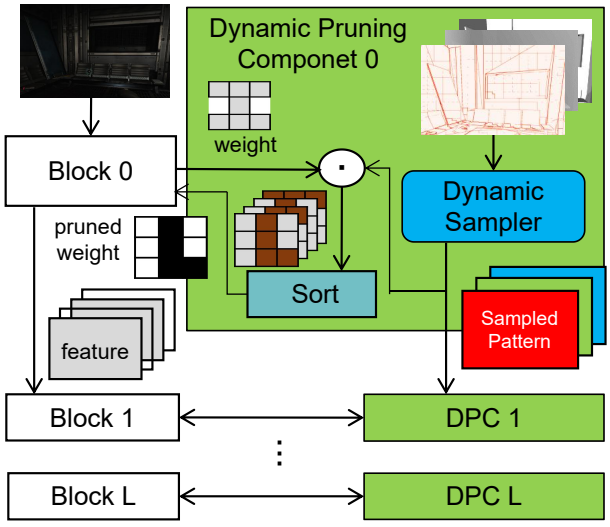


Figure 4: The structure of rendering-aware dynamic pruning component.

dering information. To this end, we introduce a rendering-aware dynamic pruning component (DPC) for each block, as illustrated in Fig. 4. The DPC comprises a dynamic sampler, which calculates a weighted average of the inputs to generate a regularization pattern tailored to each network block while ensuring dimensional consistency. The dynamic sampler serves to preserve the "knowledge" embedded in the rendering information while also capturing the unique characteristics of various CG applications.

For a network with s blocks, the feature map from the l-th-blocks can be expressed as:

$$H_l(x) = h_{l-1}(h_{l-2}(...h_0(x_0))) \tag{2}$$

where x_0 is the input frame, $H_l(x)$ represents the feature map of l-th-blocks, and h_k is propagate function in the k-th block that projects the feature maps from the $(k-1)$-th block to the k-th block.

Similarly, for the dynamic sampler, we define the regularizing pattern for the l-th-layer as:

$$F_l(x) = f_{l-1}(f_{l-2}(...f_0(r_0))) \tag{3}$$

where r_0 represents the rendering information, $F_l(x)$ denotes the regularizing pattern after the l-th-layer, and f_k is the sampling function for the k-th-layer. We set the output size of f_k to match x_k, ensuring that each regularizing pattern corresponds to a specific feature map.

The DPC generates a set of regularized weights by applying the regularizing pattern to block weights. It then performs frequency-based sorting of these weights to determine their relative spatial importance, recording the most important weights in a pattern pool. During inference, the block weights are regularized using the current rendering information in the DPC and pruned based on the weight pattern.

To prune the network while preserving accuracy, we partially train and fine-tune the dynamic sampler for a specific CG application. The loss function is defined as follows:

$$L_p = ||HR_i - SR_i||_2 + \sum_{l=1}^{K} ||W_P^l||, \tag{4}$$

where W_P^i denotes the less important weight identified at the l-th-layer.

Experiment

Dataset and Implementation

Traditionally, training an efficient SR network relies on large datasets specific to each application to enhance SR quality. However, due to the absence of publicly available datasets containing real-time rendering information, we took the initiative to pioneer the generation of such datasets. Using a popular GPU simulator (del Barrio et al. 2006), we created a large-scale dataset comprising thousands of real game frames with integrated rendering information. This approach has additional benefits: both low-resolution and high-resolution images are raw renders, with low-resolution images not derived from high-resolution ones, ensuring data authenticity and precision. Additionally, it guarantees that the rendering information of both resolutions is ground truth, maintaining accuracy throughout every stage of the process.

To validate our design across various applications, we selected three real-world game traces supported by the simulator: Doom 3, Quake 4, and The Chronicles of Riddick. For each frame, we captured rendered color, depth, pixel normal vectors, and object edges. These traces were obtained from random periods during actual game sessions or built-in performance tests. For each game trace, we render thousands of frames at various resolutions (320x240, 640x480, 1280x960, 1600x1200), ensuring consistent graphics configuration and filtering methods across all settings for accurate rendering results. 10% percent of the frames, uniformly sampled from the entire period, are reserved as test sets to evaluate the performance and SR quality of CGSR. Additionally, 70% of the frames are used for training, while the remaining 20% are set aside for validation.

The CGSR framework is implemented in PyTorch 1.5.0 and integrated into the selected backbone to create the CG-enhanced network. For architecture and parameter shrinking, we use a population size of 50 and a candidate selection ratio of 0.2. In each generation, candidate networks are trained for 5 epochs using an Adam optimizer with a cosine annealing learning rate that decays from $1e-3$ to $1e-6$. The CG-optimized network is then trained for 30 epochs. During rendering-guided hybrid pruning, the CG-optimized network is fine-tuned for 2 epochs in each dynamic pruning run. The final network is evaluated on 10 NVIDIA Tesla V100 GPUs.

Case Study: Performance of CG-optimized PAN

To evaluate our CGSR framework, we use the advanced lightweight PAN (Zhao et al. 2020) as a case study due to its high-quality outputs and efficient, compact architecture, which balances quality and computational demands for real-time applications. Its feature extraction-reconstruction de-

Scale ×2	Doom3			Quake 4			tCoR		
Model	Params(k)	M-A(G)	PSNR/SSIM	Params(k)	M-A(G)	PSNR/SSIM	Params(k)	M-A(G)	PSNR/SSIM
FSR	NA	NA	33.56/0.9167	NA	NA	29.14/0.8204	NA	NA	29.14/0.8204
SRCNN	57	53	33.56/0.9167	57	53	32.42/0.8937	57	18	35.32/0.9501
EDSR	1518	458	35.91/0.9355	1518	458	32.96/0.9137	1518	152	37.42/0.9578
PAN	261	71	36.16/0.9437	261	71	33.15/0.9188	261	24	38.01/0.9591
CGSR-P	62	15	36.11/0.9436	54	14	33.07/0.9151	64	6	38.16/0.9563
Scale ×3	Doom3			Quake 4			tCoR		
Model	Params(k)	M-A(G)	PSNR/SSIM	Params(k)	M-A(G)	PSNR/SSIM	Params(k)	M-A(G)	PSNR/SSIM
FSR	NA	NA	27.64/0.8191	NA	NA	24.56/0.8840	NA	NA	26.65/0.9249
SRCNN	57	119	29.14/0.8204	57	119	28.18/0.7856	57	89	30.61/0.8538
EDSR	1518	362	32.94/0.8693	1518	362	29.15/0.8421	1518	272	31.39/0.8799
PAN	261	88	34.62/0.9153	261	88	30.18/0.8913	261	66	31.53/0.8843
CGSR-P	62	21	34.55/0.9039	60	20	30.20/0.8911	64	16	31.51/0.8715

Table 1: Quantitative comparison of the network performance in Parameter Size and Multi-Adds number; and SR quality in PSNR and SSIM

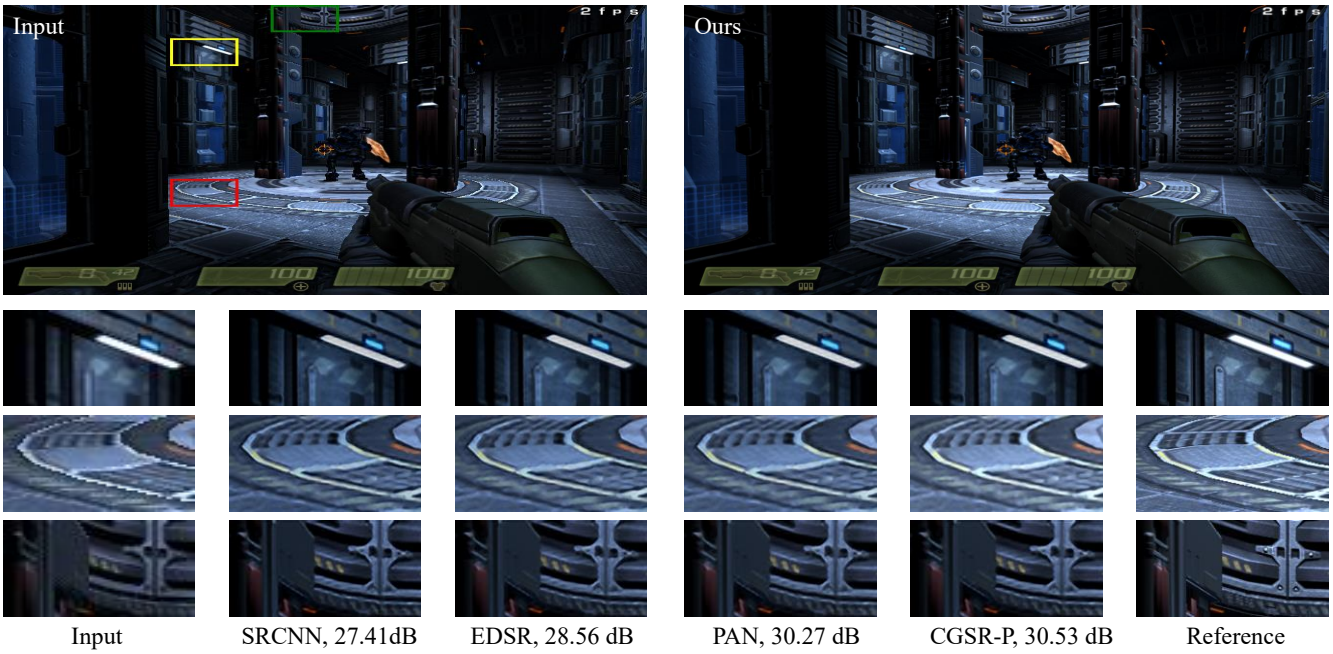
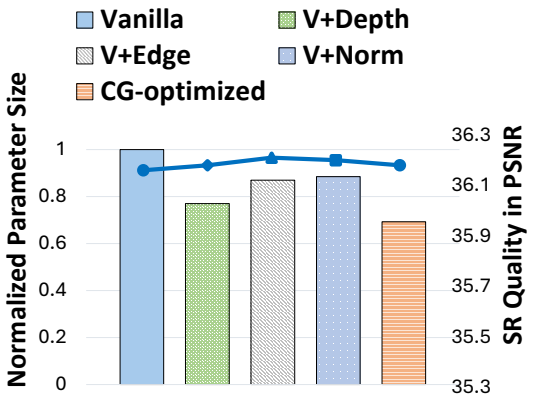


Figure 5: Visual results of the CGSR-P versus various SR networks on the Quake 4 trace.

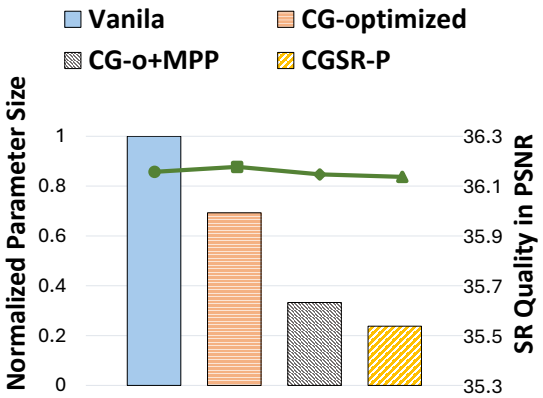
sign aligns well with our framework's requirements, making it ideal for assessment. The CGSR-optimized PAN is referred to as CGSR-P in this section. We compare CGSR-P with several networks, including the pioneering SRCNN (Dong et al. 2015), and the residual block-based VDSR (Kim, Lee, and Lee 2016) and EDSR (Lim et al. 2017), evaluating parameters such as size, multi-add operations, and SR quality. All networks are rebuilt and trained on our dataset without geometry properties. SR quality is measured using the peak signal-to-noise ratio (PSNR) (Dwivedi, Yadav, and Gupta 2023) for detail and structural similarity index (SSIM) (Wang et al. 2004) for perceptual quality, with higher values indicating better SR quality.

Network Size and SR Quality Table. 1 compares the CGSR-P with other networks at both ×2 and ×3 SR scales. Performance and quality metrics are averaged throughout each real-world game trace. The results demonstrate that the CGSR framework effectively reduces the size of the backbone network while maintaining high SR quality. For instance, when using PAN as the backbone, the optimized CGSR-P network achieves an average reduction of 77% in parameter size and 78% in multi-add operations, while delivering comparable SR quality.

Visual Comparison To visually compare user perception, we present an SR frame generated by the CGSR-P network alongside frames from other SR networks and the ground truth high-resolution frame in Fig. 5. We highlight specific



(a) Various rendering information



(b) Various pruning approaches

Figure 6: Parameter size and SR quality when employing various rendering information and pruning methods.

areas with detailed zoom-ins: object edges (yellow), high-angle normal directions (red), and sudden depth changes (green). Our method not only enhances sharpness by incorporating object edge information but also improves texture reconstruction by leveraging normals and depth, which offer crucial directional details for perceptually accurate restoration. The results demonstrate that our method closely approximates the ground truth, outperforming other methods that tend to introduce errors in these critical regions.

Ablation Study To assess the impact of each graphics property, we sequentially integrate them into the CGSR-P network. Fig. 6-(a) displays normalized parameter sizes as bars and corresponding accuracy changes as a line chart. The results show that various types of rendering information contribute to reducing parameter sizes to different extents, with the combination of all three types yielding an average performance gain of 37%. Specifically, depth significantly reduces model parameters by providing comprehensive geometric information, while normals mainly improve large-angle surfaces, offering a smaller but noticeable enhancement. Since edges have a strong impact on user perception, focusing on them alone yields the greatest quality improvement, consistent with findings in previous work (Gu et al. 2022). Additionally, the accuracy analysis demonstrates that rendering information helps the network effectively retain learned knowledge, even with reduced parameter sizes.

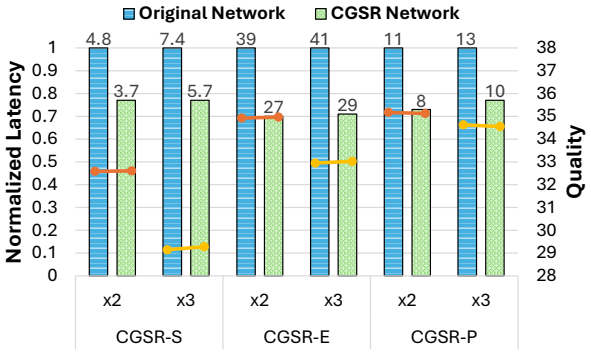


Figure 7: Evaluation across various SR networks.

Similarly, we validate our CGSR design by evaluating each phase's effectiveness, as shown in Fig. 6-(b). We compare the vanilla PAN, CG-optimized PAN without pruning, CG-optimized PAN with magnitude-based pruning (MMP), and CG-optimized PAN with our rendering-aware hybrid pruning (CGSR-P). Our CG-guided NAS network reduces parameter size by 37% while maintaining or improving quality. Rendering-aware hybrid pruning further reduces parameters by an additional 33% compared to magnitude-based pruning. The line chart shows that our pruning approach results in less accuracy loss, with a maximum of under 0.05 dB compared to the original network, demonstrating the precision and effectiveness of our design.

Evaluation on Different Backbones

For real-time CG applications, minimizing SR network inference time is crucial for responsive user interactions and meeting rendering pipeline constraints. In this section, we evaluate the latency improvements and quality of the CGSR framework across various backbones, demonstrating its effectiveness for real-time scenarios and its adaptability to networks with different characteristics and complexities. The optimized versions of these networks are labeled CGSR-S (SRCNN backbone), CGSR-E (EDSR backbone), and CGSR-P (PAN backbone).

Figure 7 shows the normalized inference times and corresponding PSNR quality for the Doom 3 trace, before and after applying the CGSR framework. The results demonstrate a consistent reduction in latency across various backbones, achieving an average decrease of 30% while maintaining similar quality. In lightweight networks like SRCNN, the benefits are modest due to its already efficient design and minimal redundancy. However, CGSR still reduces latency slightly without compromising SR quality, showing that even optimized networks can gain from the framework. In contrast, CGSR's impact is more pronounced in complex networks like EDSR, where it significantly cuts inference time by effectively pruning redundant computations.

This highlights CGSR's strength in optimizing complex architectures, making them viable for real-time CG applications without sacrificing accuracy. Specifically, CGSR-P further improves the already efficient PAN, reducing inference times from 11 ms to 8 ms for the $\times 2$ model and from 13 ms to 10 ms for the $\times 3$ model. The performance improvements ensure the network meets the critical latency deadlines required for real-time CG applications, thereby enhancing the overall user experience.

Limitation

In our experiments, our framework demonstrated significant performance improvements and real-time capabilities across various backbone networks, while notably reducing memory and computational resource requirements. However, as with other architectural optimization approaches, our methods of architecture shrinking and dynamic weight pruning might encounter workload balance issues due to potential limitations in hardware scheduling. With specialized hardware and optimized algorithm scheduling, further performance gains can be achieved. Additionally, our approach can be combined with other optimization techniques, such as quantization (Yang et al. 2019a), magnitude-based pruning (Yang, Chen, and Sze 2017), and data parallelism (Jia, Zaharia, and Aiken 2019), to further enhance SR network performance. We intend to explore these possibilities in future work.

In this study, our primary goal is to develop a versatile framework for enhancing the performance and efficiency of SR networks, rather than exhaustively exploring all potentially relevant rendering information. Consequently, we utilize a straightforward method to generate rendering data from an RTL-level GPU simulator (del Barrio et al. 2006). This approach, while effective, limits the types of rendering information available and may introduce errors in the training data. Future work will focus on a more in-depth exploration of rendering engines to extract additional beneficial information and address these limitations.

Conclusion

In this paper, we present CGSR, an optimization framework for lightweight super-resolution (SR) that integrates real-time rendering information into the SR process. This framework addresses the gap between the computational demands of emerging real-time CG applications and the capabilities of current SR networks. While most state-of-the-art SR networks only utilize a few basic outputs from the CG process, our approach leverages the strong relationship between SR and geometry properties. CGSR achieves significant improvements over various SR backbones in both parameter size and multi-add operations while preserving high SR quality. Additionally, the CG-correlated design of our framework allows it to be extended to incorporate other rendering information, providing a robust foundation for future advancements in neural SR for real-time computer graphics.

Acknowledgments

The work of Yu Wen, Chen Zhang, and Xin Fu is supported by NSF grants CCF-2130688 and CNS-2107057. The work of Chenhao Xie is supported by the Fundamental Research Funds for the Central Universities.

References

Abid Hasan, S. M.; and Ko, K. 2016. Depth edge detection by image-based smoothing and morphological operations. *Journal of Computational Design and Engineering*, 3(3): 191–197.

Ahn, N.; Kang, B.; and Sohn, K.-A. 2018. Fast, accurate, and lightweight super-resolution with cascading residual network. In *Proceedings of the European conference on computer vision (ECCV)*, 252–268.

AMD. 2024. AMD FidelityFX Super Resolution. https://www.amd.com/en/technologies/fidelityfx-super-resolution.

Chen, W.; Yue, H.; Wang, J.; and Wu, X. 2014. An improved edge detection algorithm for depth map inpainting. *Optics and Lasers in Engineering*, 55: 69–77.

Cruz, C.; Mehta, R.; Katkovnik, V.; and Egiazarian, K. O. 2017. Single image super-resolution based on Wiener filter in similarity domain. *IEEE Transactions on Image Processing*, 27(3): 1376–1389.

del Barrio, V.; Gonzalez, C.; Roca, J.; Fernandez, A.; and E, E. 2006. ATTILA: a cycle-level execution-driven simulator for modern GPU architectures. In *2006 IEEE International Symposium on Performance Analysis of Systems and Software*, 231–241.

Dong, C.; Loy, C. C.; He, K.; and Tang, X. 2015. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2): 295–307.

Dwivedi, S.; Yadav, R. N.; and Gupta, L. 2023. Chapter 8 - A comparative analysis of image restoration techniques. In Rajput, S. S.; Khan, N. U.; Singh, A. K.; and Arya, K. V., eds., *Digital Image Enhancement and Reconstruction*, Hybrid Computational Intelligence for Pattern Analysis, 173–211. Academic Press. ISBN 978-0-323-98370-9.

Gu, J.; Cai, H.; Dong, C.; Zhang, R.; Zhang, Y.; Yang, W.; and Yuan, C. 2022. Super-Resolution by Predicting Offsets: An Ultra-Efficient Super-Resolution Network for Rasterized Images. In Avidan, S.; Brostow, G.; Cissé, M.; Farinella, G. M.; and Hassner, T., eds., *Computer Vision – ECCV 2022*, 583–598. Cham: Springer Nature Switzerland. ISBN 978-3-031-19800-7.

Guo, J.; Fu, X.; Lin, L.; Ma, H.; Guo, Y.; Liu, S.; and Yan, L.-Q. 2021. ExtraNet: Real-Time Extrapolated Rendering for Low-Latency Temporal Supersampling. *ACM Trans. Graph.*, 40(6).

Ha, V. K.; Ren, J.; Xu, X.; Zhao, S.; Xie, G.; and Vargas, V. M. 2018. Deep learning based single image super-resolution: A survey. In *Advances in Brain Inspired Cognitive Systems: 9th International Conference, BICS 2018, Xi'an, China, July 7-8, 2018, Proceedings 9*, 106–119. Springer.

Han, D. 2013. Comparison of commonly used image interpolation methods. In *Conference of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013)*, 1556–1559. Atlantis Press.

Hardie, R. 2007. A fast image super-resolution algorithm using an adaptive Wiener filter. *IEEE Transactions on Image Processing*, 16(12): 2953–2964.

Hardie, R. C.; and Barnard, K. J. 2012. Fast super-resolution using an adaptive Wiener filter with robustness to local motion. *Optics express*, 20(19): 21053–21073.

Jia, Z.; Zaharia, M.; and Aiken, A. 2019. Beyond Data and Model Parallelism for Deep Neural Networks. *Proceedings of Machine Learning and Systems*, 1: 1–13.

Kanakaraj, S.; Nair, M. S.; and Kalady, S. 2017. SAR image super resolution using importance sampling unscented Kalman filter. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(2): 562–571.

Kim, J.; Lee, J. K.; and Lee, K. M. 2016. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1646–1654.

Kolasinski, E. M. 1995. Simulator sickness in virtual environments. *DTIC*.

LaViola Jr, J. J. 2000. A discussion of cybersickness in virtual environments. *ACM Sigchi Bulletin*, 32(1): 47–56.

Lee, J.; Park, S.; Mo, S.; Ahn, S.; and Shin, J. 2020. Layer-adaptive sparsity for the magnitude-based pruning. *arXiv preprint arXiv:2010.07611*.

Li, G.; Qian, C.; Jiang, C.; Lu, X.; and Tang, K. 2018. Optimization based layer-wise magnitude-based pruning for DNN compression. In *IJCAI*, 2383–2389.

Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; and Graf, H. P. 2016. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*.

Li, X.; Wu, Y.; Zhang, W.; Wang, R.; and Hou, F. 2020. Deep learning methods in real-time image super-resolution: a survey. *Journal of Real-Time Image Processing*, 17: 1885–1909.

Li, Y.; Zhang, Y.; Timofte, R.; Van Gool, L.; Yu, L.; Li, Y.; Li, X.; Jiang, T.; Wu, Q.; Han, M.; et al. 2023. NTIRE 2023 challenge on efficient super-resolution: Methods and results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1922–1960.

Lim, B.; Son, S.; Kim, H.; Nah, S.; and Mu Lee, K. 2017. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 136–144.

Lugmayr, A.; Danelljan, M.; and Timofte, R. 2021. NTIRE 2021 learning the super-resolution space challenge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 596–612.

Lugmayr, A.; Danelljan, M.; Timofte, R.; Kim, K.-w.; Kim, Y.; Lee, J.-y.; Li, Z.; Pan, J.; Shim, D.; Song, K.-U.; et al. 2022. NTIRE 2022 challenge on learning the super-resolution space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 786–797.

Mahapatra, D.; Bozorgtabar, B.; and Garnavi, R. 2019. Image super-resolution using progressive generative adversarial networks for medical image analysis. *Computerized Medical Imaging and Graphics*, 71: 30–39.

Meta. 2023. Unwrap mixed reality with Meta Quest 3. https://www.meta.com/quest/quest-3.

Rahimi, A.; Moallem, P.; Shahtalebi, K.; and Momeni, M. 2019. Using Kalman filter in the frequency domain for multi-frame scalable super resolution. *Signal Processing*, 155: 108–129.

Schmeing, M.; and Jiang, X. 2014. Edge-aware depth image filtering using color segmentation. *Pattern Recognition Letters*, 50: 63–71.

Siu, W.-C.; and Hung, K.-W. 2012. Review of image interpolation and super-resolution. In *Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference*, 1–10. IEEE.

Slowik, A.; and Kwasnicka, H. 2020. Evolutionary algorithms and their applications to engineering problems. *Neural Computing and Applications*, 32: 12363–12379.

Takehara, K.; Adrian, R.; Etoh, G.; and Christensen, K. 2000. A Kalman tracker for super-resolution PIV. *Experiments in Fluids*, 29(Suppl 1): S034–S041.

Thévenaz, P.; Blu, T.; and Unser, M. 2000. Image interpolation and resampling. *Handbook of medical imaging, processing and analysis*, 1(1): 393–420.

Vikhar, P. A. 2016. Evolutionary algorithms: A critical review and its future prospects. In *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, 261–265.

Wang, X.; Dong, C.; and Shan, Y. 2022. RepSR: Training Efficient VGG-style Super-Resolution Networks with Structural Re-Parameterization and Batch Normalization. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, 2556–2564. New York, NY, USA: Association for Computing Machinery. ISBN 9781450392037.

Wang, Z.; Bovik, A.; Sheikh, H.; and Simoncelli, E. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612.

Wen, Y.; Xie, C.; Song, S. L.; and Fu, X. 2023. Post0-VR: Enabling Universal Realistic Rendering for Modern VR via Exploiting Architectural Similarity and Data Sharing. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 390–402.

Wikipedia. 2024. Screen-door effect. https://en.wikipedia.org/wiki/Screen-door_effect.

Xiao, L.; Nouri, S.; Chapman, M.; Fix, A.; Lanman, D.; and Kaplanyan, A. 2020. Neural Supersampling for Real-Time Rendering. *ACM Trans. Graph.*, 39(4).

Yang, J.; Shen, X.; Xing, J.; Tian, X.; Li, H.; Deng, B.; Huang, J.; and Hua, X.-s. 2019a. Quantization networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7308–7316.

Yang, S.; Zhu, Q.; Zhuge, J.; Qiu, Q.; Li, C.; Yan, Y.; Xu, H.; Yan, L.-Q.; and Jin, X. 2024. Mob-FGSR: Frame Generation and Super Resolution for Mobile Real-Time Rendering. In *ACM SIGGRAPH 2024 Conference Papers*, SIGGRAPH '24. New York, NY, USA: Association for Computing Machinery. ISBN 9798400705250.

Yang, T.-J.; Chen, Y.-H.; and Sze, V. 2017. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5687–5695.

Yang, W.; Zhang, X.; Tian, Y.; Wang, W.; Xue, J.-H.; and Liao, Q. 2019b. Deep learning for single image super-resolution: A brief review. *IEEE Transactions on Multimedia*, 21(12): 3106–3121.

Yu, J.; Fan, Y.; Yang, J.; Xu, N.; Wang, Z.; Wang, X.; and Huang, T. 2018. Wide activation for efficient and accurate image super-resolution. *arXiv preprint arXiv:1808.08718*.

Zhang, J.; Zhan, Z.-h.; Lin, Y.; Chen, N.; Gong, Y.-j.; Zhong, J.-h.; Chung, H. S.; Li, Y.; and Shi, Y.-h. 2011. Evolutionary computation meets machine learning: A survey. *IEEE Computational Intelligence Magazine*, 6(4): 68–75.

Zhang, L.; Zhang, H.; Shen, H.; and Li, P. 2010. A super-resolution reconstruction algorithm for surveillance images. *Signal Processing*, 90(3): 848–859.

Zhao, H.; Kong, X.; He, J.; Qiao, Y.; and Dong, C. 2020. Efficient image super-resolution using pixel attention. In *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, 56–72. Springer.