Journal of Database Management
Volume 34 • Issue 3

Neural Super-Resolution in Real-Time Rendering Using Auxiliary Feature Enhancement

Zhihua Zhong, Zhejiang University City College, China
Guanlin Chen, Zhejiang University City College, China*
Rui Wang, Zhejiang University City College, China
Yuchi Huo, Zhejiang University City College, China

ABSTRACT

As the demand for high quality and high resolution in real-time rendering grows, superresolution is on its way to becoming a necessary component in modern real-time rendering applications (e.g., video games). The superresolution technique allows graphic applications to save computational costs by rendering at a lower resolution and reconstructing a high-resolution result. Nvidia introduced DLSS to the market as the first superresolution application in 2020, and NSRR was published on Siggraph the same year. Each of these approaches has shown powerful capabilities and is well suited to the needs of the industrial sector. In this paper, the authors propose the optimization potential of superresolution algorithms by introducing feature enhancement and feature caching modules and attempt to improve the current algorithms.

KEYWORDS

Deep Learning, Rendering, Superresolution

INTRODUCTION

The GPU was invented to meet the demands of massively parallel computation tasks from computer graphics, so a mutual relationship exists between the development of GPU performance and computer graphics technology. Display devices have developed in recent years, including 4K/8K high-resolution monitors and virtual reality headsets, which demand higher resolutions and refresh rates than before. The exponential growth in resolution places a tremendous burden on the GPU, while the GPU computing power can only grow linearly. Because modern hardware is far from meeting the actual requirements, it is inevitable to keep rendering at low resolution and obtain high resolution through some means.

DOI: 10.4018/JDM.321544 *Corresponding Author

While superresolution is a relatively new topic in computer graphics, there are other methods of rendering at a resolution different from the target. In fact, when dealing with the aliasing problem, an approach called Super-Sampling Anti-Aliasing (SSAA) solves it by sampling at a higher resolution and downsampling to the original resolution. It can be seen that antialiasing and superresolution are both reducible to undersampling, the same basic graphical problem. Aliasing results from insufficient sampling, whereas superresolution is expected to reconstruct the result with a much lower resolution. As a result, advanced antialiasing methods can be modified to solve superresolution problems.

The G-buffers are visual data structures that are packed with various scene information and can be easily accessed in real-time rendering applications. In most cases, buffers are produced in just a few milliseconds per frame in a single draw call. Using G-buffers effectively can help the network predict high-resolution results better.

RELATED WORK

Image and Video Superresolution

Superresolution in traditional image processing is an ill-posed problem. There is no additional information input between the input low-resolution image and the output high-resolution image. The algorithm cannot increase the entropy of the data and can only some way guess the pixel color of the high-resolution image according to the input image data. Neural network machine learning methods that have emerged in recent years can be seen as an attempt to obtain the network to learn additional information from the sample and learn a correspondence between low-resolution pixels and high-resolution pixels.

Most traditional image super-resolution algorithms use interpolation to generate the value of new pixels using low-resolution pixel information in a certain adjacent area. The most common interpolation algorithm is to use indirect B-spline convolution kernel interpolation to obtain the spatial value of new pixels. The higher the order of the B-spline convolution kernel, the larger the perceptual domain of the convolution kernel, and the more pixels will be used to guide the interpolation result.

The final interpolation result will also become low-frequency accordingly. Among them, 0-order, 1-order, and 3-order B-splines are commonly used B-spline convolution kernel orders, corresponding to nearest neighbor interpolation method, bilinear interpolation method, and bicubic interpolation method.

The nearest neighbor interpolation method does not attempt to add any detail information and directly copies the value of the nearest low-resolution pixel point to the new pixel point.

Bilinear interpolation is a method of linearly interpolating four adjacent pixels (two points from a one-dimensional perspective). Compared with the nearest neighbor interpolation method, the result of bilinear interpolation is more blurred at high-frequency edges. The bicubic interpolation method proposed by Keys constructs a cubic function to interpolate based on more adjacent pixel information in an attempt to capture detail information that bilinear interpolation cannot capture. When the order of the B-spline convolution kernel tends to infinity, the convolution kernel tends to be a Gaussian function, and high-frequency information is lost more severely. Therefore, indirect B-spline convolution kernel interpolation methods often obtain blurry results. However, B-spline convolution kernel interpolation methods with deconvolution operations can prevent high-frequency information loss during interpolation and obtain high-resolution results with sharp edges although they cannot supplement missing high-frequency information. Li et al. proposed a super-resolution algorithm that interpolates based on local covariance coefficients, which can effectively process high-frequency information and get sharp edges of objects.

Superresolution tasks can generally be divided into two types depending on the type of data: single-image superresolution (SISR) and video superresolution.

In addition, there are algorithms that accomplish super-resolution without using interpolation. Irani et al. propose a back-projection method that allows super-resolution results to better retain

information from the original input by accounting for the differences introduced by the upsampling method. Noting that sharpness in natural images is relatively stable and does not change with resolution, Sun et al. propose a super-resolution based on the gradient of the image edges method. Glasner et al. exploit the self-similarity of the image to find similar detail information within the image to supplement the rest of the insufficient detail. There are also methods that try to avoid the situation where image details need to be estimated. Avidan et al. noticed that the super-resolution work can be done by increasing the number of pixels in unimportant regions, by defining an energy function that gradually adds imperceptible pixels to the image at the lowest energy location for the human eye, and finally outputs a high-resolution image that stretches the low-frequency regions and leaves the high-frequency regions unaltered.

The traditional methods to solve the SISR problem are interpolation methods, such as bilinear, bicubic, edge-directed and other interpolation methods, and methods using self-similarity to reconstruct the details of high-resolution images. On the other hand, machine learning methods, from SRCNN (Dong et al., 2015), the earliest three-tier convolutional network, to SRResNet (Ledig et al., 2017), which began to introduce residual networks, now have a number of ways to solve this problem. Except for the scenario where some complex networks may learn features directly from datasets and therefore provide extra information for superresolution, neural networks are essentially just a higher-order interpolation function. Unlike SISR, video hyperresolution improves the effect of reconstructing high-resolution images by mining adjacent frames in temporal domains, in addition to using data in spatial domains.

Real-Time Rendering

The primary objective for real-time rendering is to ensure real-time capability, but both anti-aliasing and superresolution require a larger number of samples, which adversely impacts the real-time capability in most cases. Similar to video, real-time rendering relies on the persistence of vision, playing discrete images in the temporal domain at a rapid rate. The two methods are, however, fundamentally different in a number of ways.

First, although frames played in video are also discrete in the temporal domain, the information recorded in each frame is sampled over a certain period of time. The film is exposed to the light on the scene for a specified period before a frame is depicted. In general, rendering can only draw a picture in a single instant. Thus, a sense of fluency can be obtained by using a video with 24 frames per second (FPS). In contrast, it demands at least 30 FPS to avoid obvious frame drops and even more than 144 FPS to surpass the refresh rate of human eyes. In view of this difference, real-time rendering is presented with a tougher challenge that requires it to complete more complex tasks in a much shorter period of time than video.

Second, video records the ground truth directly in the spatial domain. At a certain location in a video frame, all light is recorded in that area. The rendering process, however, is limited to computing only one incident light at a time, which means taking a discrete sample in the spatial domain. In accordance with the Nyquist theorem, the original signal cannot be reconstructed if the sampling rate is less than twice the maximum frequency component of that signal. This phenomenon is known as aliasing, and it is precisely for this reason that antialiasing techniques are needed.

Antialiasing

Solve the problem of aliasing, low-pass anti-aliasing filters are used in signal processing, where the original signal is first low-pass filtered to remove the high-frequency information and then reconstructed, i.e., the effects caused by aliasing can be avoided. In contrast, in graphical plotting, an anti-aliasing algorithm is used to low-pass filter the plotted samples to obtain the desired anti-aliasing results.

Super-Sampling Anti-Aliasing (SSAA) achieves low-pass filtering of the original signal by increasing the number of samples, drawing multiple times at different locations within a pixel, and

averaging the drawing results. The oversampling antialiasing algorithm solves the aliasing problem using the simplest way of increasing the number of plots, which makes the plotting workload increase dramatically, and using the oversampling antialiasing algorithm usually leads to difficulties in securing real-time performance.

Multi-Sampling Anti-Aliasing (MSAA) also uses the same way to increase the number of samples to solve the sampling problem, but the Multi-Sampling Anti-Aliasing algorithm goes further to exploit the characteristics of the sampling problem in the graphics drawing, almost all the sampling in the drawing occurs at the edges of the object, and the color changes in the same object within a pixel is not very frequent. Therefore, the multisampling anti-walking algorithm blends the colors of different objects based on the area of covered pixels only at the edges of the objects, significantly reducing the number of samples that have no significant impact on the final drawing results.

Fast Approximate Anti-Aliasing (FXAA), on the other hand, accomplishes the anti-sampling work by post-processing without increasing the drawing effort. The Fast Approximate Anti-Aliasing algorithm generates an approximate antialiasing result quickly by detecting the edges of the object and blending the nearby pixels based on the characteristic that the aliasing occurs at the edges of the object.

Temporal Anti-Aliasing (TAA), inspired by the optical flow method in images, takes advantage of the nature of high frame rates with little variation between consecutive frames applied in real time, and reprojects samples from historical frames to the current frame for blending. The time-domain antialiasing algorithm and its various variants are the current SOTA antialiasing algorithms. The time-domain antialiasing algorithm solves the aliasing problem well while not increasing the number of samples additionally. However, the time-domain inverse sampling algorithm also suffers from ghosting and flickering due to the lack of information in the occluded region of the historical frames.

This kind of algorithm takes advantage of the temporal coherency of rendering application results, where continuous rendering results in the temporal domain mostly remain continuous, with only a few occasions where high-frequency variations occur. Even though this rarely occurs, it can seriously undermine the functionality of real-time applications. An extensive number of heuristics have been proposed to mitigate the negative effects of this situation, but these approaches affect the details of the results to a certain extent. Since the rendering application still preserves the original virtual scene, extra information such as depth map and motion vector can be acquired for a much lower cost than video, in which the scene information has already deteriorated.

The QW-Net proposed by Thomas et al. in 2020 uses a neural network to accomplish the mixing of current and historical frames in TAA, allowing more flexibility in balancing error correction strength and detail loss when reusing historical frames The QW-Net introduced by (Thomas et al., 2020) uses a neural network to accomplish the blending of current and previous frames in TAA, allowing more flexibility in balancing error correction strength and detail loss when reusing previous frames. This method proposes a new quantized image reconstruction neural network, which improves the quality of the reconstructed images while ensuring real-time performance.

Real-time Superresolution

The need for higher resolution and more photorealistic results necessitates the reduction of the real-time sampling rate and some means of mitigating its drawback. The industry was introduced to a new technique in 2016 known as CheckerBoard Rendering, which renders only one pixel for every four pixels and then interpolates the result of those pixels. In 2020, Unreal Engine updated temporal antialiasing upsampling (TAAU) as its superresolution solution. In the same year, Deep-Learned Supersampling 2.0(DLSS) was released by Nvidia, as well as Neural Supersampling for RealTime Rendering (NSRR) (Xiao et al., 2020), both of which use machine learning to enhance superresolution.

DLSS uses deep learning technology to upscale images from a lower resolution to a higher one while maintaining the same level of visual quality. It's a type of video rendering technique that looks to boost framerates by rendering frames at a lower resolution than displayed and using deep learning algorithms to upscale them back to the native resolution. FidelityFX Super Resolution(FSR) which

is an upscaling technology developed by AMD that uses advanced optimization techniques to help boost frame rates in some games without requiring users to upgrade their graphics card and can run on any graphics card. In addition, Intel released Xe Super Sampling (XeSS) which is an upscaling feature of Intel Arc Alchemist graphics cards. It works by rendering your game at a lower resolution and then upscaling it using machine learning and dedicated AI hardware found inside the GPU. XeSS can run on dedicated XMX cores or on general hardware that supports DP4a instruction.

Neural Networks for Reconstruction

The application of neural networks to rendering problems has gained increasing attention in recent years. A particular area of development has been the denoising of Monte Carlo renderings, which is aimed at eliminating the effects of distributed noise. (Chaitanya et al., 2017) introduced a technique that directs the extraction of the denoised image using a network based on U-Net (Ronneberger et al., 2015). They were the first to demonstrate interactive results at a temporally stable rate by introducing persistent connections within a UNet network. Due to the U-Net system's multiscale architecture, the system offers an expansive receptive field at an affordable price, and this is important for denoising sparsely sampled images.

During real-time applications, inputs to reconstruction techniques such as TAA are denoised via filters specialized for effect types. The use of neural networks for image reconstruction has previously been demonstrated by Marco Salvi using a UNet with warped feedback loops. A recurrent U-Net was also employed by (Kaplanyan et al., 2019). to reconstruct the peripheral image for foveated rendering from relatively sparse samples. The computing and storage costs of neural networks are greatly reduced when they are quantized to use reduced-precision arithmetic. In the case of HDR images, a quantized network can result in a significant loss in image quality. (Thomas et al., 2020) introduced QW-Net, a quantized neural network for image reconstruction, which guarantees high-quality output while using quantization to reduce computational consumption.

METHOD

In this section, we will describe our improved superresolution algorithm. We will first discuss the main solution ideas for real-time superresolution algorithms and explore potential improvements from past work. Next, we will present our proposed network structure and improvement scheme.

Problem Setting

Similar to the Anti-Aliasing problem, in real-time rendering, superresolution needs to address the issue of how to recover high-quality, high-resolution results while maintaining a low sampling rate for the current frame. TAA provides an important idea for solving this problem: reusing samples captured in historical frames to complement the undersampling of the current frame.

The 2020 work NSRR follows this idea by using historic frame information to reconstruct the current frame high-resolution results. By combining with machine learning methods, which have become popular in recent years, NSRR projects each frame of information into hidden space and upsamples it to the target resolution. A reweighting network will then adjust the warped history frame data in hidden space. In the end, all the feature maps are fed into a reconstruction network to obtain the final result. The NSRR pioneered the use of neural networks to resolve the dilemma of ghosting and missing details in traditional heuristic methods. However, the dataset that NSRR uses is primarily based on VR headset input, with relatively small and smooth camera movement characteristics. Therefore, the NSRR does not perform as well when faced with the challenge of more complex camera movements.In rendering, it is the shading task that is the greatest burden for each pixel, which can be shown by the successful use of MSAA. This reveals that in addition to the motion vector and depth map, there is additional information available at low cost that can be utilized by the network to improve its performance.

Basic Idea

In modern real-time rendering applications, G-buffer are data prepared by the rendering pipeline before shading, which are rich in various scene information and can be easily accessed. G-buffers include various data information needed for rendering, which can include material information such as albedo, metallicity, roughness, and specular, as well as geometric information such as position, surface and shading normal, depth, and motion vectors. Despite being rich in information, G-buffers can be readily available at a very low cost for real-time rendering. All these G-buffers can usually be produced simultaneously in a single draw call, at the cost of just a few milliseconds per frame. G-buffers are generated and saved in many deferred rendering applications, and even with forward rendering many real-time rendering applications still generate G-buffers for various purposes. Therefore, providing networks with reliable and cheap G-buffers can help networks better reconstruct high-resolution results almost for free. It is important for the network design to comprehensively extract useful features from the additional auxiliary G-buffer inputs, which can be much beneficial for synthesizing high-resolution results from these fertile geometric and material information.

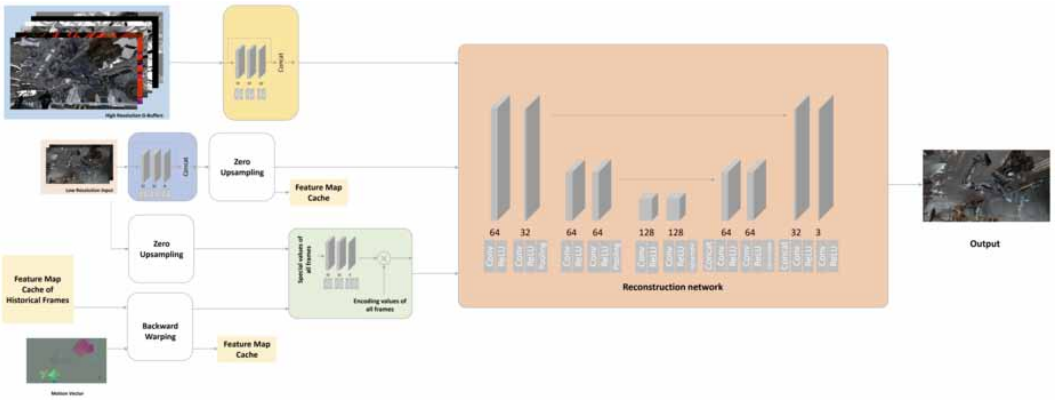
To acquire G-buffers, rendering application first project all the vertices into the NDC space (normalized device coordinate space) by a perspective transformation, and scatter the vertex information from rasterized triangle meshes of the scene into their corresponding projected pixels by interpolation. In this way, material information attached by UV mapping to each mesh vertex is accessible by each pixel via the same interpolation. Then rendering application cache, the information required in lighting stage into several writeable texture in GPU. These operations are highly accelerated in modern GPU, which can be performed in parallel by numerous GPU rendering cores in a very short time. The computational cost of generating G-buffers is associated with the complexity of the geometry features of the scene.

In contrast to the very cheap G-buffer acquisition process, the subsequent lighting (or shading) stage, is very time costly, because modern rendering pipelines strive for synthesizing photo realistic rendering appearance, which requires a lot of complex light transport computations and simulation of physically based material reflections and refractions. The computational cost in lighting stage is also relevant to the amount and property of light sources, which determine the final visual quality of rendering result. So, in modern rendering scenes, artists design more complicated light sources to produce more photographic realistic result, which burdening lighting stage rather than G-buffer generating. In this paper, we propose to accelerate this costly lighting stage by superresolution. By using superresolution method enhanced by cheap auxiliary feature, the computational cost in lighting stage, which is the main rendering cost in modern scenario, can be cutdown sharply. Enhanced by superresolution, the rendering pipeline only needs to render a low resolution image and feeds it into the superresolution framework to synthesize final rendering results with high resolution and visual quality. The acceleration ratio is proportional to the superresolution ratio. In this paper, we provide 2x2 and 4x4 superresolution ratios, which can help to reduce the rendering cost by 4 times or even 16 times respectively. The overhead of our superresolution framework is far less than the saved high resolution rendering time, due to the cheapness of G-buffer acquisition and the lightweight of our designed neural network. Our experiments show that with the help of auxiliary G-buffer information as the neural network input, the superresolution framework can produce more high-quality results. Our method outperforms all the baseline superresolution methods with a lower cost to achieve state-of-the-art.

Network Architecture

Our network structure is improved on the basis of NSRR. Figure 1 is an overview of our network structure. With reference to NSRR's framework of feature-reweighting network and reconstruction network implemented with U-Net, we propose a feature enhancement module and an historical feature caching acceleration module. The feature enhancement module provides inexpensive and reliable feature information extracted from G-Buffer for reconstructing the network, while the feature caching acceleration module provides acceleration by caching the historical feature maps to reduce the repetitive operations of the network.

Figure 1. Network architecture of our approach



Feature Enhancement

To add the G-buffers information to the network, we designed an additional feature extraction network on top of the NSRR network. By feeding the G-buffers information into the network, the information on which the high-resolution results of the network output are based is increased. We use UE4's builtin G-buffers, including base color, metallic, normal, roughness, and specular, as shown in Figure 2. The feature extraction network for G-buffers utilizes the same structure as the one used for low-resolution samples.

As the feature maps are extracted from the G-buffers, they will be concatenated and sent to the reconstruction network along with the upsampled feature maps of low-resolution samples and the warped feature maps of previous frames. In the end, the reconfiguration network produces the final result.

Feature Caching

The NSRR network structure converts only the current frame to the YCrCb color space, which entails that the feature extracted from the current frame is exclusively available in the current frame superresolution process. With the help of the feature enhancement module, we can significantly improve the quality of the results. By extracting in the same color space, it may be possible to use the extracted feature information as the historical information of the next frame. In addition, the redundant processing of accumulative backward warping in NSRR can also be removed after the historical feature map cache has been established.

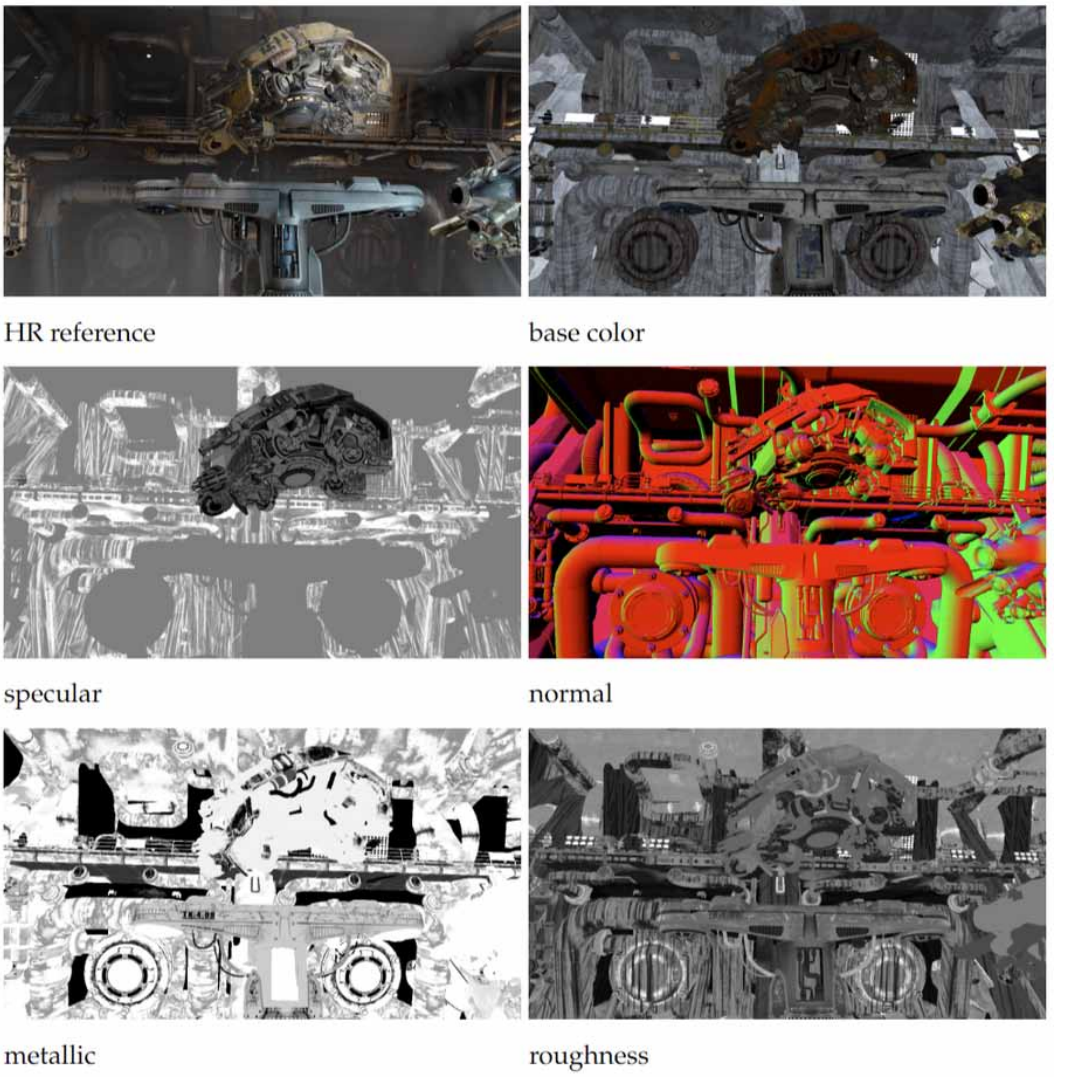
The feature cache contains only four copies of the feature maps in the previous screen space, as shown in Figure 3. The superresolution process can be initiated by simply backward warping the feature maps once at the same time, thereby providing the feature maps in the current screen space. Afterwards, the latest three feature maps from the warped results are cached in the feature cache in conjunction with the current frame feature map.

The feature caching module enables a superresolution workflow, as shown in Figure 4, to consist of only feature extraction, backward warping, reassignment, and finally reconstruction. Consequently, a large number of redundant operations are reduced while integrating parts of the operations into a more compact structure.

Implementation and Datasets

The datasets we prepare are from the cinematic scene INFILTRATOR, which is publicly available for UE4. INFILTRATOR consists of a dark indoor scene in the first half and an open city scene in the second half. The first half of the film, which we are using as the training and validation dataset,

Figure 2. Built-in G-buffers in UE4 and corresponding low-resolution samples as well as high-resolution reference



features a large amount of metallic materials and fast camera movements. The city scene, which we use as the testing dataset, in the second half contains detailed vistas with slow camera movement. We render 5120 × 2880 pixels with 8xMSAA and downscale to 1280 × 720 as a reference. For the lower resolution inputs, MSAA is turned off, and a bias is added to the mip level of the texture sampler, as indicated by equation 1. For training, we divide the images into overlapped patches with a resolution of 512 × 512 pixels, whereas for validation and testing, we use full frames with a resolution of 1280 × 720 pixels.

Our networks are trained using PyTorch. Training optimization is conducted using the ADAM method (Kingma et al., 2014) with default hyperparameters, a learning rate of 1e-4, a batch size of 16, and 100 epochs. Each network takes approximately a day to train on 8 Tesla V100 GPUs.

$$b' = b + \log_2\left(R_r \,/\, R_n\right) \tag{1}$$

Figure 3. Cached feature map

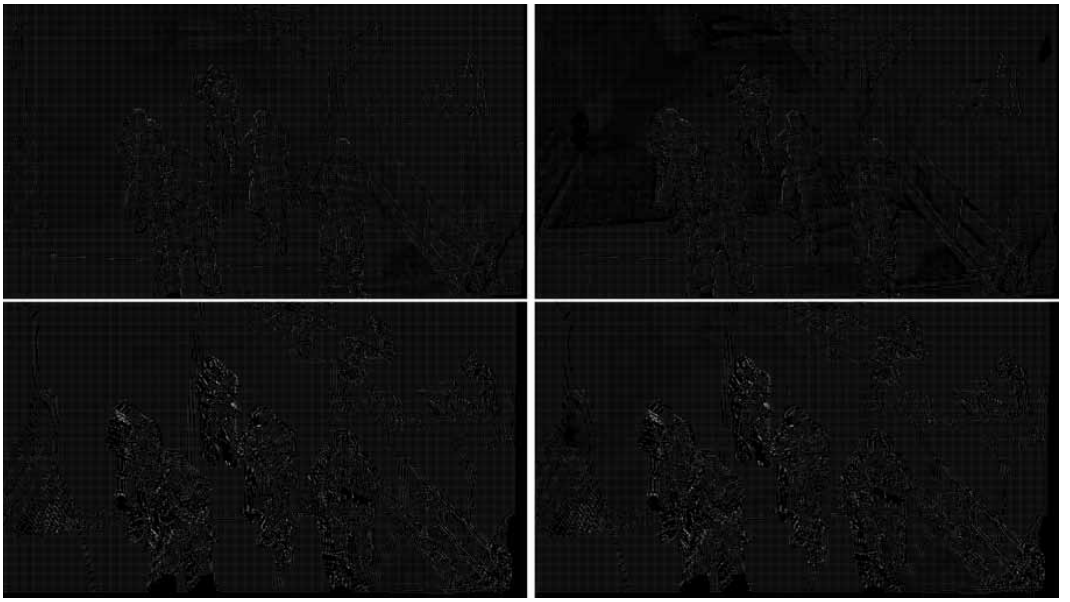
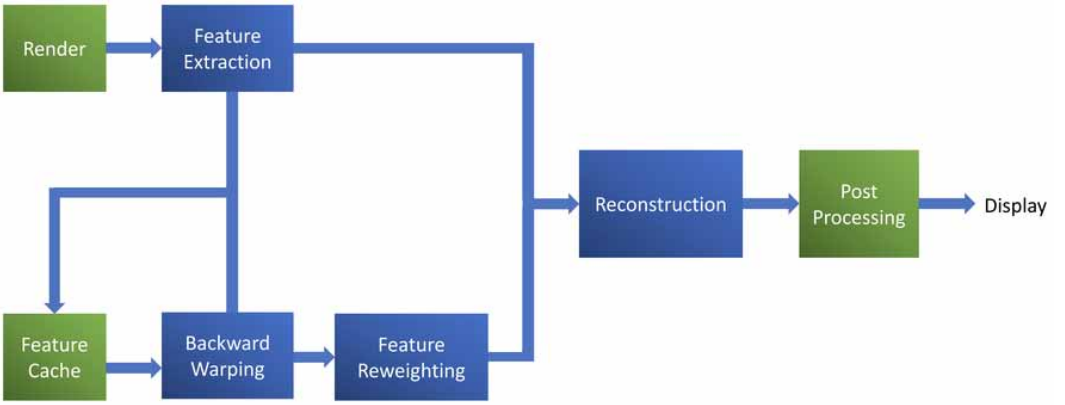


Figure 4. Workflow of our approach

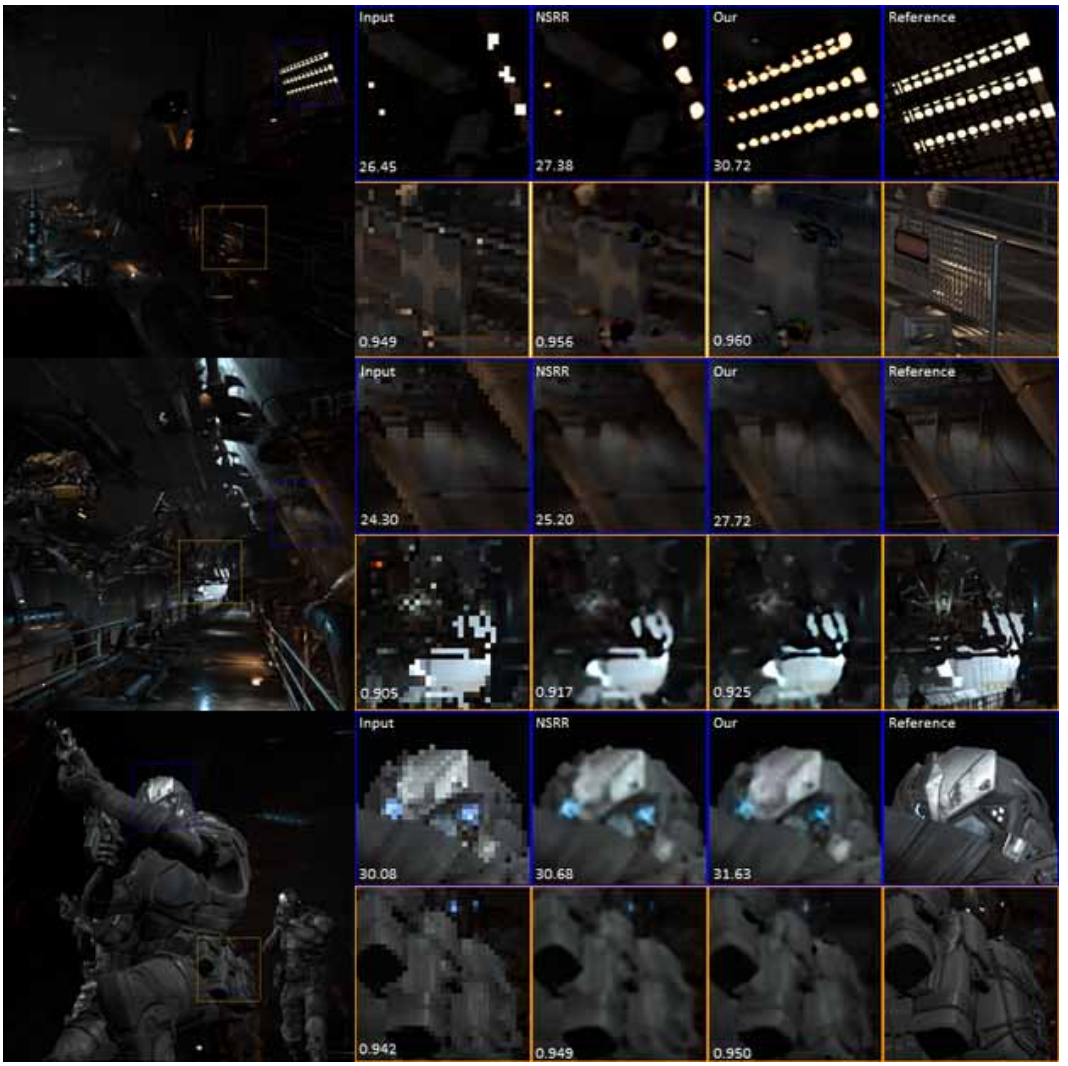


where b and b' are the Native bias and the final Mip bias, while R_r and R_n represent the rendering resolution and the Native resolution, respectively.

RESULT

In this section, we evaluate the performance of our method according to the obtained results. Furthermore, we examine how the high-resolution Gbuffers information helps the model to better predict the final high-resolution results.

Figure 5. Representative frames in our dataset, where the first two frames are from the validation dataset and the last frame is from the test dataset. We compared the NSRR with our 4x4 superresolution results, compared them with the low-resolution input and reference. In each frame we have selected two areas of interest in the results, which are framed in blue and orange. In addition, we show the PSNR metrics and SSIM values of the full figure in the lower left corner of the result



Quality Evaluation

As shown in Figure 5, we compare our method with the state-of-the-art real-time superresolution method NSRR. The NSRR network was re-implemented and trained on the same datasets as our method and uses the same training procedure as in our original method.

Two widely used photometric quality metrics are adopted to evaluate the results: the peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM) (Wang et al., 2004). For an assessment of a single image, PSNR and SSIM are well known. The higher the value, the better the image quality is.

Analysis

Frame 1 is a slowly advancing shot containing a wealth of detail. The area of interest in the blue box demonstrates the low sampling rate of the 4x superresolution low-resolution sample in the face of a high-frequency light source at medium distance, resulting in a large amount of detailed information being lost. At this point, it is difficult to recover this high-frequency information by relying only on the NSRR of low-resolution samples. In contrast, the network in our method can recover these high-frequency data through high-resolution G-buffers information. The orange area of interest demonstrates the network's ability to handle high-frequency geometric details at medium distances. The region of interest is a guardrail with a large number of holes, which are smaller than the size of a pixel in the low-resolution sample, so these details are completely lost in the low-resolution sample. Instead, this detailed information is preserved in the high-resolution G-buffers.

Frame 2 is a scene where the shots change more rapidly and there is more high-frequency information in the far field. How the network copes with the undersampling of material information is shown in the blue area of interest. NSRR resembles the result of bilinear interpolation due to the absence of other more reliable information. Our method, on the other hand, can recover some of the detailed information from the high-resolution material feature maps to some extent. The situation in the other area of interest is similar to that in the orange area of scene 1, the difference being that the object in Frame 1 itself has rich geometric detail, while in this scene, it is caused by the object being too far from the camera.

Frame 3, a close-up scene with multiple figures, shows how the high-resolution G-buffers information helps the network recover geometric edge information. In particular, the first region of interest also shows how the additional information helps the network to correct for color bias errors. In the NSRR results, the figure's helmet is tinted with some cyan blue on nearby pixels and has jagged edges similar to the low-resolution sample. In our results, on the other hand, the original color and contour of the scene can be restored due to the original material information.

CONCLUSIONSAND FUTURE WORK

Although 4x superresolution can bring a good performance improvement, the results are not stable. Relatively good results can be obtained when the camera movement is relatively flat, but when the camera and scene objects move violently, the quality of the results drops dramatically. This is because violent motion causes a dramatic decrease in the reliability of historical frame information, as a large number of historical frames are reprojected outside the viewport of the current frame. and violent motion also causes the shading result on the surface of the object to change rapidly, especially high frequency information, specular reflection for example. In addition, the backward warping error caused by the excessive difference between the rendering resolution and the local resolution is also responsible for the unstable performance of the 4x superresolution.

Jittering Samples. In the TAA (Karis, 2014) algorithm, jittering sampling is used to obtain more informative samples. When the rendered content is still or does not change much, jittering sampling can capture more different samples within a pixel range, according to a low-discrepancy sequence (Christensen et al., 2018). Superresolution allows for more efficient sample acquisition using a similar approach. In addition, dithering on low-resolution samples would specifically dither to different high-resolution pixels, which might be combined with zero upsampling to provide the network information on which samples must be generated.

G-Buffers Feature Extraction. Even though high-resolution G-buffers provide rich high-resolution information at a low cost, they will create a performance bottleneck in each process prior to reconstruction. By using pooling to generate low-resolution compact feature maps from high-resolution G-buffers, we will be able to perform subsequent steps more efficiently.

Motion Vector. Occlusion may cause pixels to be projected onto other objects, resulting in objects from different objects being projected onto the same pixel, causing a misinterpretation

by the network. Additional processing of the motion vector with other information, such as spatial position and depth, produces a more efficient motion vector (Zeng et al., 2021), which allows the network to use more reliable warped information and thus focus on reconstructing high-resolution results.

ACKNOWLEDGMENT

This work is supported by the Zhejiang Science and Technology Plan Project of China (No. 2020C03091) and the Zhejiang Provincial Central Government Guided Local Science and Technology Development Project (No. 2020ZY1010).

REFERENCES

Akeley, K. (1993). Reality engine graphics. *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, 109–116. doi:10.1145/166117.166131

Chaitanya, C. R., Kaplanyan, A. S., Schied, C., Salvi, M., Lefohn, A., Nowrouzezahrai, D., & Aila, T. (2017). Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics*, *36*(4), 1–12. doi:10.1145/3072959.3073601

Christensen, P., Kensler, A., & Kilpatrick, C. (2018). Progressive multi-jittered sample sequences. *Computer Graphics Forum*, *37*(4), 21–33. doi:10.1111/cgf.13472

Dong, C., Loy, C. C., He, K., & Tang, X. (2015). Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *38*(2), 295–307. doi:10.1109/TPAMI.2015.2439281 PMID:26761735

Games, E. (2020). *Unreal engine*. Retrieved from https://www.unrealengine.com/

Kaplanyan, A. S., Sochenov, A., Leimkühler, T., Okunev, M., Goodall, T., & Rufo, G. (2019). DeepFovea: Neural reconstruction for foveated rendering and video compression using learned statistics of natural videos. *ACM Transactions on Graphics*, *38*(6), 1–13. doi:10.1145/3355089.3356557

Karis, B. (2014). *High Quality Temporal Supersampling*. Retrieved from https://advances.realtimerendering.com/s2014/

Kingma, D. P., & Ba, J. (2014). *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412.6980.

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., & Acosta, A. (2017). Photo-realistic single image super-resolution using a generative adversarial network. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4681–4690. doi:10.1109/CVPR.2017.19

Liu, E. (2020). DLSS 2.0 - Image Reconstruction for Real-time Rendering with Deep Learning. *GPU Technology Conference (GTC).* Retrieved from https://developer.nvidia.com/gtc/2020/video/s22698-vid

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention*, 234–241. doi:10.1007/978-3-319-24574-4_28

Salvi, M. (2017). *Deep Learning: The Future of Real-Time Rendering?* Retrieved from https://openproblems.realtimerendering.com/s2017/index.html

Thomas, M. M., Vaidyanathan, K., Liktor, G., & Forbes, A. G. (2020). A reduced-precision network for image reconstruction. *ACM Transactions on Graphics*, *39*(6), 1–12. doi:10.1145/3414685.3417786

Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, *13*(4), 600–612. doi:10.1109/TIP.2003.819861 PMID:15376593

Xiao, L., Nouri, S., Chapman, M., Fix, A., Lanman, D., & Kaplanyan, A. (2020). Neural supersampling for real-time rendering. *ACM Transactions on Graphics*, *39*(4), 142–1. doi:10.1145/3386569.3392376

Yang, L., Nehab, D., Sander, P. V., Sitthi-Amorn, P., Lawrence, J., & Hoppe, H. (2009). Amortized supersampling. *ACM Transactions on Graphics*, *28*(5), 1–12. doi:10.1145/1618452.1618481

Zeng, Z., Liu, S., Yang, J., Wang, L., & Yan, L.-Q. (2021). Temporally Reliable Motion Vectors for Real-time Ray Tracing. *Computer Graphics Forum*, *40*(2), 79–90. doi:10.1111/cgf.142616