Neural Supersampling for Real-time Rendering

LEI XIAO, SALAH NOURI, MATT CHAPMAN, ALEXANDER FIX, DOUGLAS LANMAN, and ANTON KAPLANYAN, Facebook Reality Labs

Fig. 1. Results of our real-time, learned 4×4 supersampling are shown for four sample scenes. From top to bottom: the rendered low-resolution color input, our reconstruction, and the rendered reference images. Our supersampling method takes the color, depth, and motion vectors of multiple low-resolution frames, and produces high-fidelity reconstructions by reducing aliasing and recovering scene details.

Due to higher resolutions and refresh rates, as well as more photorealistic effects, real-time rendering has become increasingly challenging for video games and emerging virtual reality headsets. To meet this demand, modern graphics hardware and game engines often reduce the computational cost by rendering at a lower resolution and then upsampling to the native resolution. Following the recent advances in image and video superresolution in computer vision, we propose a machine learning approach that is specifically tailored for high-quality upsampling of rendered content in real-time applications. The main insight of our work is that in rendered content, the image pixels are point-sampled, but precise temporal dynamics are available. Our method combines this specific information that is typically available in modern renderers (i.e., depth and dense motion vectors) with a novel temporal network design that takes into account such specifics and is aimed at maximizing video quality while delivering real-time performance. By training on a large synthetic dataset rendered from multiple 3D scenes with recorded camera motion, we demonstrate high fidelity and temporally stable results in real-time, even in the highly challenging 4×4 upsampling scenario, significantly outperforming existing superresolution and temporal antialiasing work.

CCS Concepts: • **Computing methodologies** → **Machine learning**; **Rendering**.

Additional Key Words and Phrases: deep learning, rendering, upsampling, superresolution, virtual reality

ACM Reference Format:
Lei Xiao, Salah Nouri, Matt Chapman, Alexander Fix, Douglas Lanman, and Anton Kaplanyan. 2020. Neural Supersampling for Real-time Rendering. *ACM Trans. Graph.* 39, 4, Article 142 (July 2020), 12 pages. https://doi.org/10.1145/3386569.3392376

1 INTRODUCTION

Real-time rendering for modern desktop, mobile, and virtual reality applications is challenging due to increasing display resolutions and demands for photorealistic visual quality. For example, virtual reality (VR) headsets such as the Valve Index require rendering 2880×1600 pixels at 90-144Hz and recent gaming monitors support 3840×2160 resolution at 144Hz, which, together with the recent advances in physically based shading and real-time ray tracing, sets a high demand on computational power for high-quality rendering.

A multitude of techniques have been introduced to address this problem in recent years. Oculus Quest applies fixed foveated rendering, for which peripheral regions are rendered at low resolution. Kaplanyan et al. [2019] employ gaze-contingent foveated reconstruction by rendering non-uniform sparse pixel samples followed by

Authors' address: Lei Xiao, lei.xiao@fb.com; Salah Nouri, snouri@fb.com; Matt Chapman, mchapman@fb.com; Alexander Fix, alexander.fix@fb.com; Douglas Lanman, douglas.lanman@fb.com; Anton Kaplanyan, kaplanyan@fb.com, Facebook Reality Labs.

0730-0301/2020/7-ART142
https://doi.org/10.1145/3386569.3392376

neural reconstruction. Unreal Engine [2020] introduces the temporal antialiasing upscaling (TAAU) method which utilizes pixel color statistics and temporal accumulation for supersampling. Variable rate shading [Microsoft 2019] has been introduced recently to accelerate rendering by reducing the shading complexity for foveated and high-resolution displays. Closest to our work, Nvidia has recently released deep-learned supersampling (DLSS) [Edelsten et al. 2019] that upsamples low-resolution rendered content with a neural network in real-time. However, these methods either introduce obvious visual artifacts into the upsampled images, especially at upsampling ratios higher than 2×2, or rely on proprietary technologies and/or hardware that may be unavailable on all platforms.

In this paper, we introduce a method that is easy to integrate with modern game engines, requires no special hardware (e.g., eye tracking) or software (e.g., proprietary drivers for DLSS), making it applicable to a wider variety of existing software platforms, acceleration hardware and displays. The method takes common inputs from modern game engines, i.e., color, depth and motion vectors at a lower resolution, and significantly upsamples the input imagery to the target high resolution using a temporal convolutional neural network. Different than most existing real-time supersampling methods, which typically aim for no more than 2×2 upsampling in practice, our method allows for compelling 4×4 upsampling from highly aliased input and produces high fidelity and temporally stable results in real-time.

While prominent advances have been demonstrated for photographic image and video upsampling with deep learning techniques, these methods typically do not apply to rendered content. The fundamental difference in image formation between rendered and photographic images is that each sample in the rendering is a point sample in both space and time, in contrast to a pixel area integral in photographic images. Therefore, the rendered content is typically highly aliased, especially at a low resolution. This makes upsampling for rendered content both an *antialiasing* and *interpolation* problem, rather than the deblurring problem as studied in existing superresolution work in computer vision community. On the other hand, pixel samples in real-time rendering are accurate, and, more importantly, motion vectors (i.e. geometric correspondences between pixels in sequential frames) are available nearly for free at subpixel precision. These inputs bring both new benefits and challenges into the superresolution problem for rendering, which motivates us to revisit the deep learning techniques for rendering.

Large datasets are necessary for training robust networks. To train for temporal stability, the datasets should also represent realistic camera motions (e.g., with large rotation and translation). We found that no existing datasets satisfy our requirements. Therefore, we build a large-scale dataset generation pipeline in Unity [2020], replay head motion captured from VR user studies, and render color, depth and motion vectors for thousands of frames for each of our representative dynamic scenes. This new dataset enables us to train and test neural networks on realistic use cases, including our proposed architecture and existing learned superresolution methods. With such comparisons, we demonstrate that our network significantly outperforms prior state-of-the-art learned superresolution and temporal antialiasing upscaling work.

We summarize our technical contributions as follows.

- We introduce a temporal neural network tailored for image supersampling of rendered content that employs rich rendering attributes (i.e., color, depth, and motion vectors) and that is optimized for real-time applications.
- We demonstrate the first learned supersampling method that achieves significant 4×4 supersampling with high spatial and temporal fidelity.
- Our method significantly outperforms prior work, including real-time temporal antialiasing upscaling and state-of-the-art image and video superresolution methods, both in terms of visual fidelity and quantitative metrics of image quality.

2 RELATED WORK

2.1 Real-time Rendering

A particularly important problem in real-time rendering is undersampling. Each sample in rendering is a point sample, while an antialiased image requires computing all samples within sensor's pixel area. For a highly detailed surface, undersampling can lead to jarring visual shimmering and flickering appearance when surface features become subpixel. Multiple methods exist to mitigate various sources of aliasing in the undersampled image. We classify these methods in two groups: spatial-only antialiasing methods use the information from a single rendered image, and temporal antialiasing methods use temporal history from multiple rendered frames.

2.1.1 Spatial Antialiasing. One classical method in real-time graphics is multisampling antialiasing (MSAA) [Akeley 1993], a form of supersampling, where the color of a polygon covered by a pixel is only calculated once, avoiding computing multiple subpixel samples for the same polygon. Another classical method is texture filtering [Williams 1983], where high-frequency details coming from surface textures are prefiltered using image pyramids. A proper prefiltered region is then selected in runtime based on the pixel's footprint projected to the textured surface. Similarly, aliasing from other shading components, such as specular highlights [Kaplanyan et al. 2016] or shadows [Reeves et al. 1987] can be addressed using specialized methods.

Most spatial antialiasing methods are based on image enhancement methods from image processing. The basic idea is to find discontinuities on the image and to blur them in clever ways, in order to smooth the jagged edges. Morphological antialiasing [Reshetov 2009] (MLAA) try to estimate the pixel coverage of the original geometry based on the color discontinuities found in the proximity of the pixels in the final image. Fast approximate antialiasing (FXAA) [Lottes 2009] approaches the undersampling problem by attenuating subpixel features, which enhances the perceived temporal stability. Subpixel morphological antialiasing (SMAA) [Jimenez et al. 2012] combines MLAA with MSAA. While these methods can provide a good quality for single still images, the temporal variation between frames antialiased with these methods still contains visible flicker and other types of false pixel motion.

2.1.2 Temporal Antialiasing and Reconstruction. These methods use temporal history, usually in a form of a temporally accumulated buffer, with some form of temporal rejection filtering. The key difference is that the high-quality temporal methods, such as amortized

supersampling [Yang et al. 2009], reproject the history from one frame to another in order to compensate for motion, similar to motion compensation used in video compression. Temporal antialiasing (TAA) [Karis 2014] uses an edge detection filter as a proxy to suppress flicker by heavier temporal accumulation. Recently, TAA has been also employed to perform temporal upsampling (TAAU) [Epic Games 2020] and we provide comparisons with our method.

Deep-learned supersampling (DLSS) [Edelsten et al. 2019] is the closest to our method, and uses temporal history and neural networks to enhance edges and perform upscaling. While being used for games, no reliable public information is available on the details, quality or performance of the method. In contrast, our work provides full details and evaluation of the method and can be employed without a need for proprietary hardware or software.

Another recent trend in reducing the rendering cost is to apply reconstruction methods to sparsely ray-traced [Schied et al. 2017] and foveated images [Patney et al. 2016]. There is a recent body of work on applying machine learning methods to real-time low-sample-count reconstruction [Chaitanya et al. 2017] and foveated reconstruction [Kaplanyan et al. 2019]. These methods train temporally stable U-Net architectures to achieve a stable reconstructed video out of very noisy and/or sparse input frames, which is related to our task of interpolation for upsampling. 与插值有关

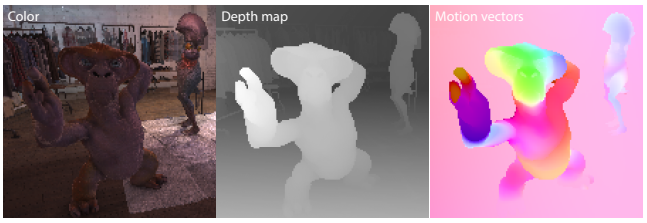

Fig. 2. Example color, depth, and motion vectors rendered by game engines.

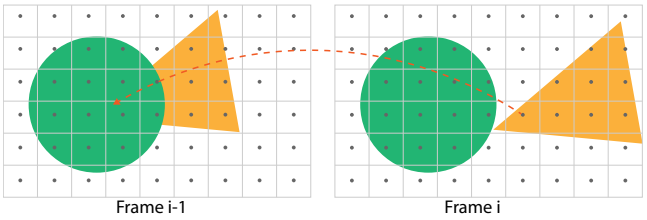

Fig. 3. Illustration of the subpixel backward motion vector rendered by game engines. The rendered point samples are represented by the black circle points at the center of each pixel. An example of the subpixel backward motion vector between frame i and $i-1$ is illustrated by the red dashed line. The motion vector defines where an infinitesimal 3D point that is visible at frame i would appear at its previous frame $i-1$, without its visibility or color information.

2.2 Image and Video Superresolution

2.2.1 Single Image Superresolution. Single image superresolution (SISR) is an ill-posed problem that aims to recover a high-resolution image from its low-resolution, typically degraded version. Since the pioneering work SRCNN [Dong et al. 2015] which uses a 3-layer convolutional neural network (CNN) for SISR, numerous deep neural network approaches have been proposed and achieved the state-of-the-art quality. We review the most relevant and advanced methods here and refer to [Yang et al. 2019] for more detailed discussions.

Instead of learning the direct mapping between the high-resolution target image and the low-resolution input image, VDSR [Kim et al. 2016] learns the residual between the two. SRResNet [Ledig et al. 2017] applies residual network architecture [He et al. 2016] to the superresolution problem, and EDSR [Lim et al. 2017] further improves the performance by utilizing more, modified residual blocks. ESPCN [Shi et al. 2016] introduces a subpixel CNN that operates at low resolution and achieves real-time performance. LapSRN [Lai et al. 2017] proposes a Laplacian pyramid network for progressively reconstructing the sub-band residuals of high-resolution images. RDN [Zhang et al. 2018b] incorporates residual and dense connections [Huang et al. 2017] for hierarchical feature extraction. RCAN [Zhang et al. 2018a] introduces a residual-in-residual structure to form a very deep network, i.e., over 400 convolutional layers, and a channel attention mechanism to adaptively rescale channel-wise features, achieving the state-of-the-art results in quality. Other work [Ge et al. 2018; Ledig et al. 2017] build on generative adversarial networks for optimizing perceptual quality, typically leading to limited reconstruction fidelity and temporal consistency.

2.2.2 Video Superresolution. Video superresolution methods typically exploit temporal coherence in adjacent frames to improve reconstruction upon SISR methods. A key component of most methods is motion estimation between frames. VESPCN [Caballero et al. 2017] introduces a multi-resolution spatial transformer module for joint motion compensation and video superresolution. SPMC-VSR [Tao et al. 2017] introduces a subpixel motion compensation layer to fuse multiple frames for revealing image details. EDVR [Wang et al. 2019] applies a pyramid, cascading and deformable alignment module and a temporal and spatial attention module. Differently, DUF [Jo et al. 2018] introduces a deep neural network that generates dynamic upsampling filters computed from the local spatio-temporal neighborhood of each pixel to avoid explicit motion compensation.

Another group of methods use recurrent neural networks (RNN) that naturally encourages temporally consistent results. FRVSR [Sajjadi et al. 2018] proposes a RNN that warps the previously estimated frame to facilitate the subsequent one. RBPN [Haris et al. 2019] develops a recurrent encoder-decoder architecture for incorporating features extracted from single-image and multi-frame modules. While RNN methods encourages temporally coherent results, their results may lack spatial details due to the averaging nature of simple norm loss functions at training. TecoGAN [Chu et al. 2018] attempts to address this problem by improving the training loss and introducing a temporally self-supervised adversarial learning algorithm.

3 METHOD

In this section, we describe our learned supersampling algorithm that is tailored for real-time rendering. We start with discussions on the challenges of rendered content upsampling and briefly overview a few key techniques of our method in Section 3.1, and then describe our network architecture in Section 3.2, training loss function in Section 3.3 and dataset collection in Section 3.4.

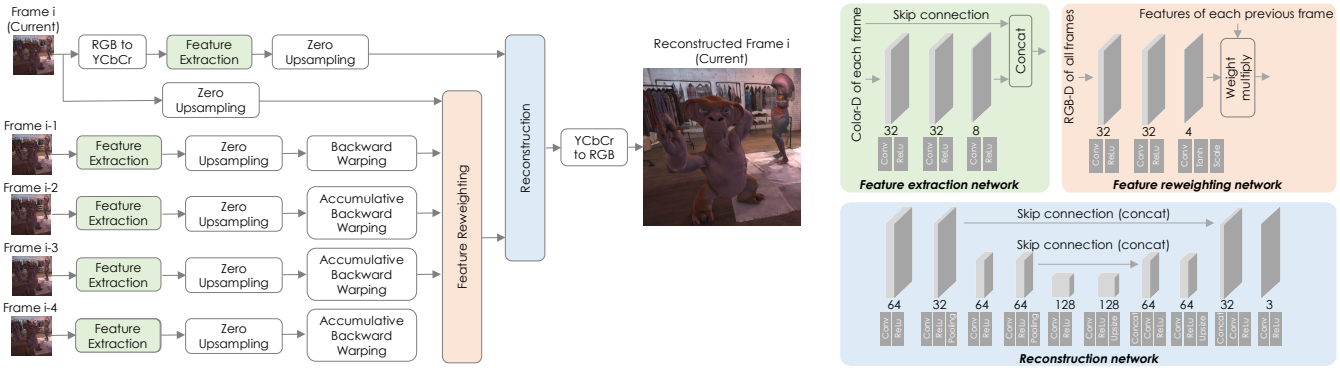

Fig. 4. Network architecture of our method. Left shows an outline of the method and right shows the architecture of each sub-network, including the feature extraction, feature reweighting, and reconstruction networks. The numbers under each network layer represent the output channels at corresponding layers. The filter size is 3×3 at all layers. The tanh layer in the feature reweighting network is followed by a scaling operation to map the values from (-1,1) to (0,10).

3.1 Problem Setting

In real-time rendering, each pixel is a point sample of the underlying high-resolution image when multi-sample antialiasing is disabled, as illustrated in Figure 3, where the dot at the center of each pixel represents the sample to render. Post-upsampling from such input is an antialiasing and interpolation problem, in contrast to a deblurring problem as in the camera image superresolution applications.

The challenge of high-fidelity upsampling for rendered content is that the input images with point-sampled pixels are extremely aliased at shading and geometry edges, and the information at the target to-be-interpolated pixels is completely missing. To address this problem, we are motivated by the fact that in rendering, we know a more detailed information about the current and past frames and the way they are sampled. To effectively leverage this information across multiple frames, we utilize the inputs commonly available in today's games engines, such as pixel color, depth map, and motion vectors, as visualized in Figure 2.

In rendering, a motion vector points at an analytically computed screen-space location where a 3D point that is visible at the current frame would appear in the previous frame, with a subpixel precision, as shown in Figure 3. While the rendered motion vector provides candidate matching between pixels for low cost, it presents a few limitations that prevent its direct use in multi-frame upsampling. *First*, because it maps pixels backwards as illustrated in Figure 3, pixels at a previous frame cannot be directly projected to the current frame. *Second*, it does not consider dynamic disocclusion between the current and previous frame, i.e., 3D points visible at current frame might be occluded in the previous frames due to viewpoint change or object movement. An example case is illustrated in Figure 3. *Third*, it provides only a partial optical flow, i.e., only the motion of surface points and camera, and does not consider the change in lighting, shadows, view-dependent reflectance, etc. Consequently, the reconstruction method needs to robustly handle these limitations while taking advantage of the rendered motion vectors.

While the method details are given in Section 3.2, we briefly overview a few key insights of our method here. Similar to several existing video superresolution work (Section 2.2.2), our method first warps previous frames to align with the current frame, in order to reduce the required receptive field and complexity of the reconstruction network. In contrast to existing work, however, to better exploit the specifics of rendered data, i.e., point-sampled colors and subpixel-precise motion vectors, our method applies the frame warping at the target (high) resolution space rather than at the input (low) resolution. Specifically, the method projects the input pixels to the high resolution space, prior to the warping, by zero-upsampling. The details of the zero-upsampling and warping are given in Section 3.2.2.

As the rendered motion vectors do not reflect disocclusion or shading changes between frames, the warped previous frames would contain invalid pixels mismatching with the current frame, which would mislead the post-reconstruction. To address this problem, we include a reweighting mechanism before the reconstruction network to de-select those invalid pixels. The reweighting mechanism is related to the confidence map approaches used for multi-frame blending in various applications [Hasinoff et al. 2016; Karis 2014; Mildenhall et al. 2019; Wronski et al. 2019]. In contrast to these approaches, however, our method utilizes a neural network to learn the reweighting weights. The details of our reweighting module are given in Section 3.2.3.

Lastly, the preprocessed previous frames (after zero-upsampling, warping and reweighting) are stacked together with the current frame (after zero-upsampling), and fed into a reconstruction network for generating the desired high-resolution image, as details given in Section 3.2.4.

3.2 Network Architecture

In this section, we describe our method in details with Figure 4.

3.2.1 Feature Extraction. The feature extraction module contains a 3-layer convolutional neural network. This subnetwork processes each input frame individually, and shares weights across all frames except for the current frame. For each frame, the subnetwork takes color and depth as input, and generates 8-channel learned features, which are then concatenated with the input color and depth, resulting in 12-channel features in total. The network details are given in Figure 4 in green.

3.2.2 Temporal Reprojection. To reduce the required receptive field and thus complexity of the reconstruction network, we apply temporal reprojection to project pixel samples and learned features of each previous frame to the current, by using the rendered motion vectors. In order to fully exploit the subpixel backward motion vectors, we conduct the temporal reprojection at the target (high) resolution space. First, we project the pixel samples from input (low) resolution space to the high resolution space, by zero upsampling, i.e. assigning each input pixel to its corresponding pixel at high resolution and leaving all the missing pixels around it as zeros. The location of each input pixel falls equally in between s pixels in the high resolution, where s is the upsampling ratio. Zero upsampling is chosen for its efficiency and because it provides the network information on which samples are valid or invalid.

Then, we resize the rendered low resolution map of motion vectors to high resolution simply by bilinear upsampling, taking advantage of the fact that the motion vectors are piece-wise smooth. While such simple upsampling introduces errors to the upsampled map at discontinuous regions, it well recovers the majority of regions compared to ground truth. Next, we apply backward warping of the zero-upsampled previous frames using the upsampled motion vectors, while bilinear interpolation is adopted during warping.

In Figure 5, we show an example of the zero-upsampled and warped frames. Note that the motion vectors are only defined for an adjacent pair of frames. To warp across multiple previous frames, we apply the described warping process iteratively until each previous frame is warped to the current one. We use up to 4 previous frames in our experiments.

Performing warping at the zero-upsampled target resolution space reduces the effect of low-pass interpolation during warping and thus protects the high-frequency information contained in the rendered point samples. This makes our network distinct from existing superresolution work that typically warps frames at the input low resolution space.

3.2.3 Feature Reweighting. As discussed in Section 3.1, the rendered motion vectors do not reflect dynamic disocclusions or shading changes between frames. Thus the warped frames would contain artifacts such as ghosting at disocclusion regions and mismatched pixels at inconsistent shading regions.

To address this problem, we introduce a feature reweighting module to be able to mask out these mismatched samples. The feature reweighting module is a 3-layer convolutional neural network, which takes the RGB-D of the zero-upsampled current frame as well as the zero-upsampled, warped previous frames as input, and generates a pixel-wise weighting map for each previous frame, with values between 0 and 10, where 10 is a hyperparameter. The hyperparameter is set to allow the learned map to not just attenuate, but also amplify the features per pixel, and empirically we found the dynamic range of 10 was enough.

Then each weighting map is multiplied to *all features* of the corresponding previous frame. The reason we feed only RGB-D, instead of the whole 12-channel features, into the reweighting network is to further reduce the network complexity. The network details are given in Figure 4 in orange, and an example of a learned reweighting map is given in Figure 5.

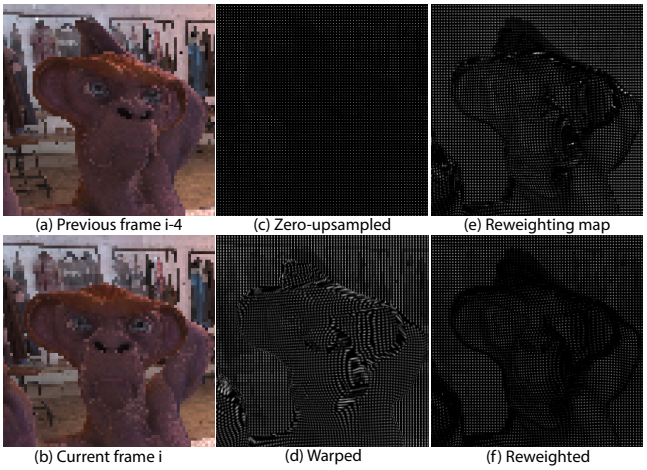

Fig. 5. Example intermediate outputs of our method. In subfigures (c)-(f) we visualize a single channel of the feature map in greyscale. The features of previous frame $i-4$ are zero-upsampled (fig. c) and warped to align with the current frame i, using the subpixel backward motion vectors (fig. d). The warped features go through the feature reweighting subnetwork (fig. e), and the re-weighted features (fig. f) are then fed into the reconstruction subnetwork for final results. Best viewed zoomed on a monitor.

3.2.4 Reconstruction. Finally, the features of the current frame and the reweighted features of previous frames are concatenated and fed into a reconstruction network, which outputs the recovered high resolution image of the current frame. We adopt a 3-scale, 10-layer U-Net with skip connections for the reconstruction subnetwork. The network details are given in Figure 4 in blue.

3.2.5 Color Space. The method optionally converts the input RGB image of current frame to YCbCr color space, before feeding it to the neural network. The direct output of the network and the training loss stay in YCbCr space, before the result is converted back to RGB space for viewing. While optional, we experimentally find the colorspace conversion slightly improves reconstruction quality, i.e. 0.1dB improvement in peak signal-to-noise ratio (PSNR).

3.3 Losses

The training loss of our method, as given in (1), is a weighted combination of the perceptual loss computed from a pretrained VGG-16 network as introduced in Johnson et al. [2016], and the structural similarity index (SSIM) [Wang et al. 2004].

$$loss(\mathbf{x}, \bar{\mathbf{x}}) = 1 - \mathrm{SSIM}(\mathbf{x}, \bar{\mathbf{x}}) + w \cdot \sum_{i=1}^{5} ||\mathrm{conv}_i(\mathbf{x}) - \mathrm{conv}_i(\bar{\mathbf{x}})||_2^2 \quad (1)$$

where $\mathbf{x}$ and $\bar{\mathbf{x}}$ are the network output and reference high-resolution image respectively, and the relative weight is $w = 0.1$. We refer to Johnson et al. [2016] for the full details of selected VGG-16 layers.

3.4 Datasets and Implementation

We train a separate network for each 3D scene unless specified in the experiments. Large datasets are necessary for training robust networks. We collected several representative, dynamic scenes in Unity [2020] and built a large-scale dataset generation program to

render the training and test data. The program replays head motions that were captured from user studies in a VR headset, and renders color, depth and motion vectors of every frame.

Specifically, we render 100 videos from each scene, and each video contains 60 frames. Each video's camera starts from a random position in the scene and moves as defined in a pre-captured head motion path that is randomly selected for each video from a large candidate pool. For reference images, we first render the images at 4800×2700 with 8× MSAA and then downscale the images to 1600×900 with 3×3 box filters to further reduce aliasing. For low-resolution input images, we turn off MSAA and adjust mip level bias for texture sampling to match the selected mip level with the full resolution images. The mip level bias approach is applied to reduce prefiltering in the rendered low-resolution images and is similarly done in existing supersampling algorithms such as TAAU in Unreal [2020].

During training, 80 videos are used to generate training batches, 10 for validation batches, and the remaining 10 are for testing. For training and validation, we divide the images into overlapped patches with resolution 256×256 pixels, while for testing we run the network on the full frames with 1600×900 pixels. Our network is fully convolutional so it is able to take any resolution as input.

We train our networks with TensorFlow [Abadi et al. 2015]. The network weights are initialized following Glorot et al. [2010]. The ADAM method [Kingma and Ba 2014] with default hyperparameters is used for training optimization, with learning rate 1e-4, batch size 8, and 100 epochs of the data. Each network takes around 1.5 days to train on a Titan V GPU.

4 RESULTS

In this section, we evaluate the performance of our method. We report its runtime in Section 4.1, compare its quality in Section 4.2, analyze the algorithm with various ablation experiments in Section 4.3, and describe its limitations and future work in Section 4.4.

4.1 Runtime Performance

After training, the network models are optimized with Nvidia TensorRT [2018] at 16-bit precision and tested on a Titan V GPU. In Table 1, we report the total runtime of our method for 4×4 supersampling at varying target resolutions, including 720p (1280×720), Oculus Rift (1080×1200) and 1080p (1920×1080). In Table 2, we report the runtime breakdown of our method with 4×4 supersampling at 1080p. The runtime is reported in unit of milliseconds (ms).

To study the trade-off between network complexity and reconstruction quality, in Table 1, 2 and 3, we report two flavors of our method, i.e., the primary network, namely "Ours", and a lighter version, namely "Ours-Fast". The hyperparameters of the primary network are given in Figure 4, and the only difference in the lighter network is that the output channels of each layer except for the last one in the reconstruction U-Net are reduced by 50%. In Table 3, we compare the reconstruction quality of the two networks. The lighter network has minor decreased quality compared to the primary, however, both networks outperform existing methods by a large margin as compared in the following Section 4.2. In the remainder of the section, we report the results using the primary network.

Table 1. Runtime (ms) of our 4×4 upsampling for varying target resolutions, including 720p (1280×720), Oculus Rift (1080×1200) and 1080p (1920×1080).

	1280×720	1080×1200	1920×1080
Ours	11.96	15.99	24.42
Ours-Fast	8.84	11.87	18.25

Table 2. Runtime breakdown for 4×4 upsampling for 1080p resolution. Corresponding time cost of the lighter network "Ours-Fast" is included in parentheses (the two networks are identical through Feature Reweighting).

Module	Time (ms)
Feature extraction	0.97
Motion vector upsampling	0.25
Feature zero-upsampling	0.28
Warping	0.90
Feature reweighting	4.73
Reconstruction	Ours 17.2 (Ours-Fast 11.1)
Total	Ours 24.4 (Ours-Fast 18.3)

Table 3. Quality comparisons of variants for our network. "Ours" and "Ours-Fast" represent the primary and lighter networks respectively, trained on each scene separately (Section 4.1). "Ours-AllScenes" and "Ours-AllButOne" represent the primary networks trained on all scenes together, and on all scenes but the one tested, respectively (Section 4.3.2).

		Ours-Fast	Ours	Ours-AllScenes	Ours-AllButOne
PSNR (dB)	Robots	35.68	36.08	36.01	35.39
	Village	30.36	30.70	30.75	30.18
	DanceStudio	33.56	34.07	33.68	33.21
	Spaceship	36.09	36.69	36.64	35.60
SSIM	Robots	0.9657	0.9696	0.9692	0.9626
	Village	0.8892	0.9019	0.9002	0.8753
	DanceStudio	0.9176	0.9224	0.9201	0.9104
	Spaceship	0.9674	0.9712	0.9696	0.9647

Table 4. Quality comparisons with existing methods on all scenes. Results for each method averaged across 10 test videos in each scene. Our method outperforms all others by a large margin on every quality metric. Note that different than PSNR and SSIM, lower values in STRRED mean higher quality results.

		ESPCN	VESPCN	DUF	EDSR	RCAN	Ours
PSNR (dB)	Robots	31.62	31.72	32.30	33.72	33.40	36.08
	Village	27.26	27.39	27.62	27.74	27.77	30.70
	DanceStudio	30.24	30.41	30.96	31.64	31.62	34.07
	Spaceship	32.73	32.80	33.65	34.29	34.39	36.69
SSIM	Robots	0.9134	0.9142	0.9335	0.9476	0.9440	0.9696
	Village	0.7908	0.7950	0.8270	0.8296	0.8294	0.9019
	DanceStudio	0.8375	0.8418	0.8640	0.8794	0.8777	0.9224
	Spaceship	0.9119	0.9123	0.9286	0.9427	0.9418	0.9712
STRRED	Robots	109.7	103.5	73.2	56.5	63.6	19.3
	Village	192.4	186.6	131.8	169.8	168.6	42.5
	DanceStudio	213.0	194.8	118.8	117.8	121.6	40.6
	Spaceship	98.8	96.6	66.6	58.1	58.4	22.1

4.2 Quality Evaluation

We compare our method to several state-of-the-art superresolution work, including single image superresolution methods ESPCN [Shi et al. 2016], EDSR [Lim et al. 2017] and RCAN [Zhang et al. 2018a], and video superresolution methods VESPCN [Caballero et al. 2017]

and DUF [Jo et al. 2018]. We re-implemented and trained all the methods on the same datasets as in our method with the same training procedure. For the video superresolution methods, we adjusted their networks to take only current and previous frames as input, avoiding any future frames. The number of input previous frames used in video superresolution methods is also increased to 4 to match our method.

We evaluate the results with three quality metrics: peak signal-to-noise ratio (PSNR), structural similarity index (SSIM) [Wang et al. 2004], and spatio-temporal entropic difference (STRRED) [Soundararajan and Bovik 2012]. PSNR and SSIM are well-known for single image assessment, the higher the better. STRRED is widely used for video quality assessment that includes temporal stability, the lower the better. We evaluate the results on four representative scenes, namely Robots, Village, DanceStudio and Spaceship.

In Table 4, we compare the above quality metrics, averaged over 10 test videos from our dataset described in Section 3.4. In Figure 7 and 8, we compare result images both visually and quantitatively. Our method outperforms all other methods on all scenes by a large margin.

In addition, we compare to the temporal antialiasing upscaling (TAAU) method from Unreal Engine [2020]. We take the Robots scene as an example, and convert it to Unreal to collect the TAAU results. As the scene in Unity and Unreal cannot be matched exactly, we first provide visual comparisons in Figure 6, showing that our method produces significantly better visual quality. Furthermore, we evaluate the PSNR and SSIM of ours and TAAU with respect to each own's reference image rendered at Unity and Unreal respectively. Our result (PSNR = 31.74dB, SSIM = 0.9430) significantly outperforms TAAU (PSNR = 30.06dB, SSIM = 0.9070).

For all results in this section, we have included the full-frame result images in the supplementary material, and the results on video sequences in the supplemental videos. Our method produces significantly more temporally stable video results than existing approaches, both visually, and quantitatively according to the video quality metric STRRED.

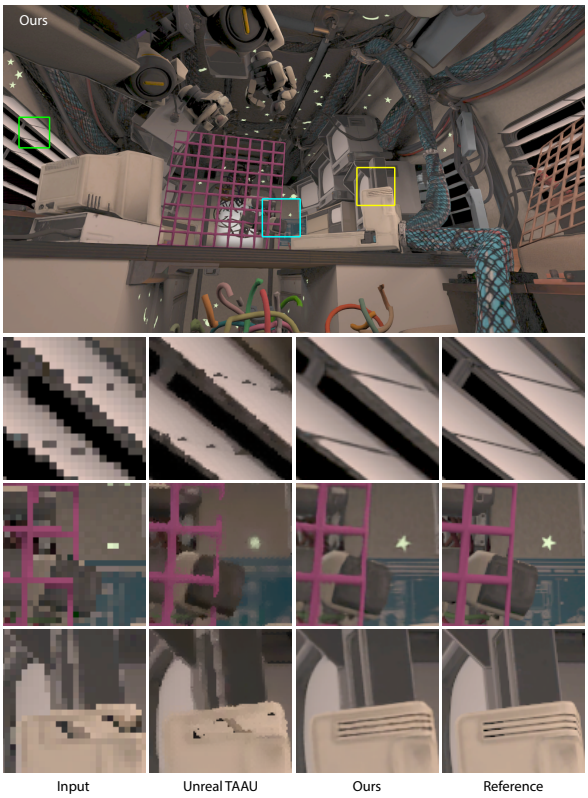

Fig. 6. Comparison with Unreal TAAU with 4×4 supersampling. The PSNR and SSIM of our and TAAU results are computed with respect to each's own reference image rendered at Unity and Unreal respectively. Our resulting PSNR is 31.74dB, SSIM is 0.9430, outperforming TAAU whose resulting PSNR is 30.06dB, SSIM is 0.9070.

4.3 Analysis

4.3.1 Rendering Efficiency. We take the Spaceship scene as a representative scenario to demonstrate how the end-to-end rendering efficiency can be improved by applying our method. We render on an Nvidia Titan RTX GPU using the expensive and high-quality ray-traced global illumination effect available in Unity. The render pass for a full resolution image takes 140.6ms at 1600×900. On the other hand, rendering the image at 400×225 takes 26.40ms, followed by our method, which takes 17.68ms (the primary network) to upsample the image to the target 1600×900 resolution, totaling to 44.08ms. This leads to an over 3× rendering performance improvement, while providing high-fidelity results.

4.3.2 Generalization. While we choose to train a network for each scene to maximize its quality, an open question we would like to answer is how it generalizes across scenes. In Table 3, we report the quality of our primary network trained jointly on all four scenes ("Ours-AllScenes") and trained on all scenes but the one tested ("Ours-AllButOne"), respectively, and compare them to the primary network trained on each scene separately ("Ours"). The test quality reduces slightly with Ours-AllScenes (0.05 − 0.4dB in PSNR) and more with Ours-AllButOne (0.5−1dB in PSNR). However both networks still noticeably outperform all comparison methods that are trained on each scene separately. This indicates that the network can generalize across scenes with different appearance although including the test scenes into training datasets seems to always improve the quality. However a full evaluation of network generalization will require collecting more scenes.

4.3.3 Previous Frames. In Table 5, we report the reconstruction quality by using a varying number of previous frames. The quality increases as more previous frames are used, however, the network runtime likewise increases. Of note is that runtime is dominated by the reconstruction sub-network (Table 2). Only the first layer of this part is affected by the number of frames, so adding more previous frames only slightly increases runtime. Thus, applications can vary this parameter to get to a sweet spot in quality/runtime trade-off. In Figure 9 we show a visual comparison to the network variant using a single (current) frame as input. The experiment demonstrates the quality gained from the use of previous frames.

4.3.4 Supersampling Ratios. In Table 6, we report the reconstruction quality of our method with varying supersampling ratios from 2×2 to 6×6. In this experiment, we keep the target resolution the

Fig. 7. Visual results on the Robots (top) and Village (bottom) scene.

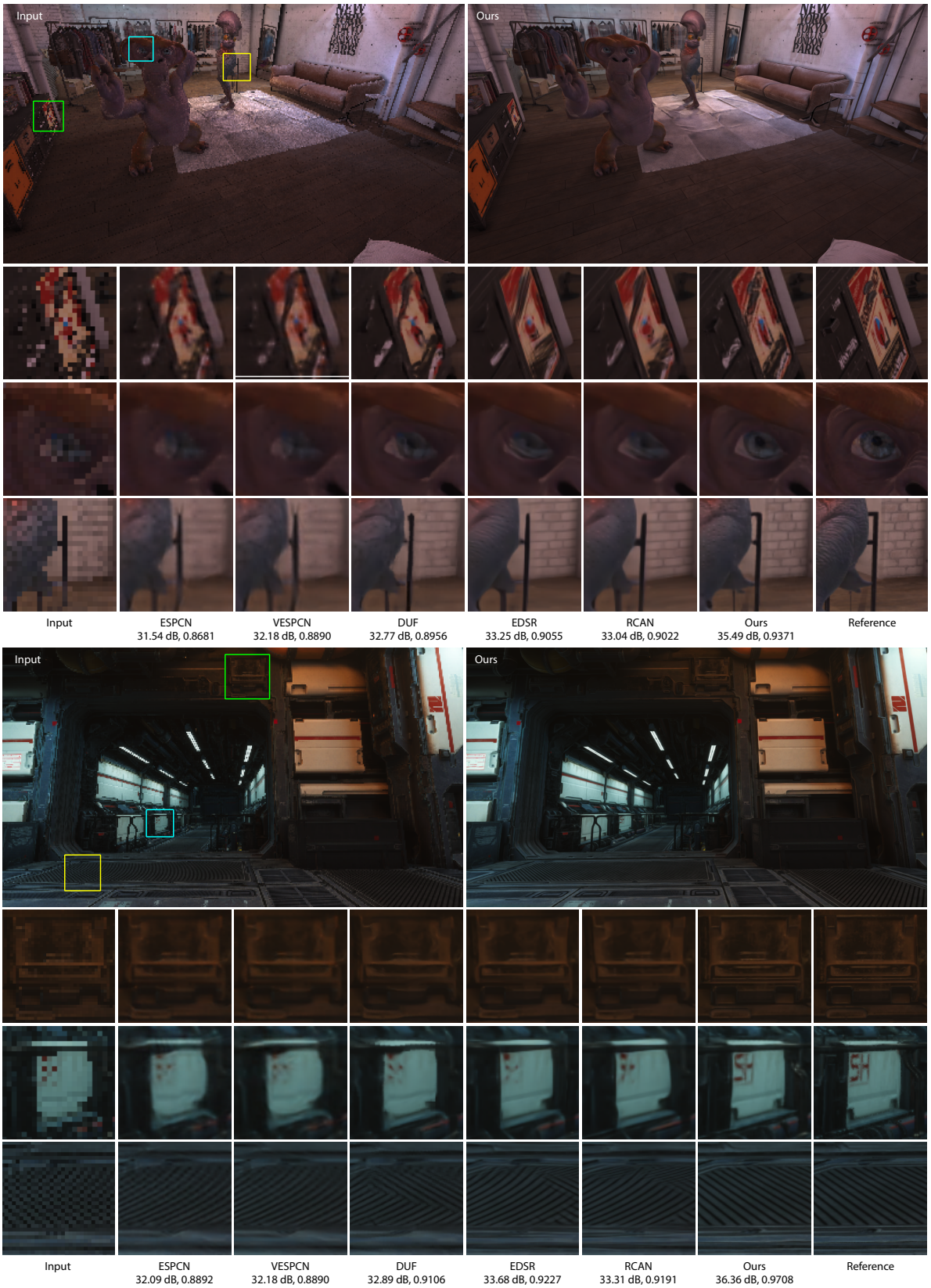

Fig. 8. Visual results on the DanceStudio (top) and Spaceship (bottom) scene. Asset credits to Ruggero Corridori and Unity Technologies respectively.

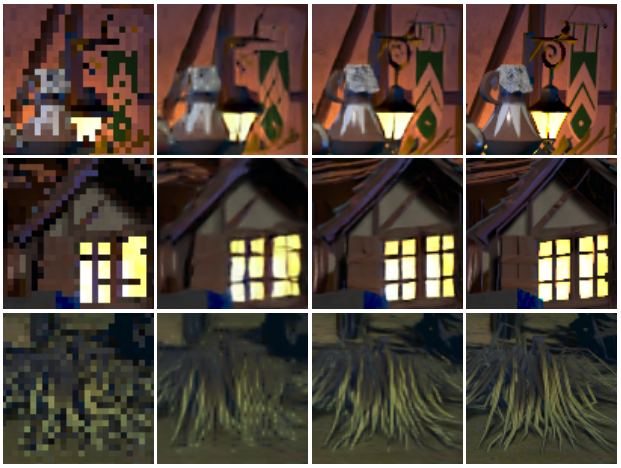

Fig. 9. Ablation experiment for performance gain from using previous frames. "Ours-SingleFrame" represents the trained network that takes only a single frame as input.

Table 5. Quality and runtime (for 1080p target resolution) versus number of previous frames used as inputs on the Robots scene.

# Previous frames	0	1	2	3	4
PSNR (dB)	32.66	34.00	34.92	35.61	36.08
SSIM	0.9340	0.9505	0.9596	0.9653	0.9696
Runtime (ms)	16.46	21.36	23.19	23.71	24.42

Table 6. Reconstruction quality versus supersampling ratio on Robots scene.

Supersampling ratio	6×6	5×5	4×4	3×3	2×2
PSNR (dB)	32.97	34.36	36.08	38.17	41.19
SSIM	0.9392	0.9538	0.9696	0.9820	0.9928

Table 7. Quantitative comparisons for 2×2 upsampling on Robots scene.

	ESPCN	VESPCN	DUF	EDSR	RCAN	Ours
PSNR (dB)	36.21	36.39	37.12	38.10	37.99	41.19
SSIM	0.9692	0.9694	0.9772	0.9881	0.9807	0.9928

same and vary the input image resolution according to the supersampling ratio. As expected, the reconstruction quality gracefully improves as the supersampling ratio reduces.

Additionally, to verify the performance advantage of our method at varying supersampling ratios, we train all existing methods with 2×2 upsampling and report the results in Table 7. Our method significantly outperforms the existing work.

4.3.5 Quality Gain from Additional Inputs. While our method outperforms all compared methods by a large margin, we would like to understand the quality gain from its additional depth and motion vector inputs. We revise the VESPCN method to take the same depth and motion vector input as ours, namely "VESPCN+", where the motion vectors replace the optical flow estimation module in the original VESPCN and the depth is fed as an additional channel together with the RGB color input. As reported in Table 8, with the additional inputs, VESPCN+ improves moderately (1.1-1.3dB in PSNR) upon VESPCN, however it is still noticeably worse (2.2-3.1dB in PSNR) than our method. This indicates that both the additional inputs and the specifically tailored network design of our method play important roles in our performance achievement.

Table 8. Ablation experiment for quality gain from the additional depth and motion vector inputs. VESPCN+, which is modified from VESPCN, takes the same depth and motion vector inputs as ours. PSNR and SSIM are reported.

	VESPCN	VESPCN+	Ours
Robots	31.72dB / 0.9142	33.03dB / 0.9250	36.08dB / 0.9696
Village	27.39dB / 0.7950	28.55dB / 0.8222	30.70dB / 0.9019
DanceStudio	30.41dB / 0.8418	31.48dB / 0.8584	34.07dB / 0.9224
Spaceship	32.80dB / 0.9123	33.91dB / 0.9251	36.69dB / 0.9712

Table 9. Ablation experiment for temporal reprojection (Section 3.2.2). The network is trained with each of the following upsampling and warping settings, and results on the Robots scene are reported.

	PSNR (dB)	SSIM
Warp at low-res, then bilinear upsampling	34.97	0.9570
Warp at low-res, then zero-upsampling	35.08	0.9570
Bilinear upsampling, then warp at high-res	35.82	0.9658
Zero-upsampling, then warp at high-res (Ours)	36.08	0.9696

Table 10. Ablation experiment for the feature extraction and feature reweighting modules. The network is trained with each (and both) of these submodules removed, and results on the Robots scene are reported.

Feature Extraction	Feature Reweighting	PSNR (dB)	SSIM
×	×	35.63	0.9652
×	✓	35.76	0.9670
✓	×	35.90	0.9678
✓	✓	36.08	0.9696

4.3.6 Zero-Upsampling and Warping. As described in Section 3.2.2 (temporal reprojection), our method projects input pixels to the target (high) resolution space by zero-upsampling, and then warps the upsampled previous frames to the current frame for post-processing. To understand its impact on the reconstruction quality, we experiment with alternative ways for temporal reprojection, i.e., replacing zero-upsampling with bilinear upsampling and/or warping at the input (low) resolution space instead, and the results are reported in Table 9. We observe about 1dB improvement in PSNR by warping at the target resolution compared to at the input resolution, and about 0.3dB additional improvement by using zero-upsampling compared to bilinear upsampling. This indicates the benefit of our approach tailored for effectively leveraging the rendering-specific inputs, i.e., point-sampled color and subpixel-precise motion vectors.

4.3.7 Network Modules. In Table 10, we report the ablation experiments for analyzing the quality improvements from the feature extraction (Section 3.2.1) and feature reweighting (Section 3.2.3) modules. Average results are reported on the 10 test videos of the Robots scene. While the numeric results show only minor improvements from the reweighting module, the results are averaged over large amounts of data, and the regions affected by disocclusion and mismatched pixels (the parts of images most impacted by this module) only make up a relatively small part of the images. In Figure 10 we provide a visual comparison to demonstrate the contribution of the feature reweighting module. When the network is trained

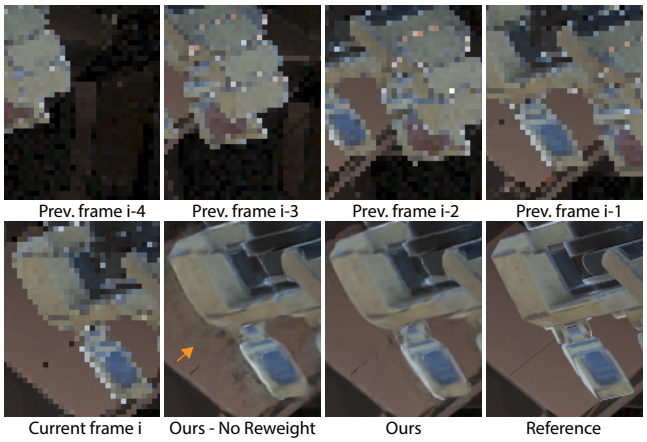

Fig. 10. Ablation experiment for the feature reweighting module. When the network is trained without the reweighting module ("Ours-NoReweight"), ghosting artifacts appear at the disoccluded regions as pointed by the orange arrow.

without the reweighting module, ghosting artifacts appear at the disocclusion regions around the robot's fingers.

4.3.8 Discussion with DLSS. While Nvidia's DLSS [Edelsten et al. 2019] also aims for learned supersampling of rendered content, no public information is available on the details of its algorithm, performance or training datasets, which makes direct comparisons impossible. Instead, we provide a preliminary ballpark analysis of its quality performance with respect to our method, however, on different types of scenes. Specifically, we took the AAA game "Islands of Nyne" supporting DLSS as an example, and captured two pairs of representative screenshots, where each pair of screenshots include the DLSS-upsampled image and the full-resolution image with no upsampling, both at 4K resolution. The content is chosen to be similar to our Spaceship and Robots scene in terms of geometric and materials complexity, with metallic (glossy) boxes and walls and some thin structures (railings, geometric floor tiles). For copyright reasons, we cannot include the exact images in the paper. Instead, we computed the PSNR and SSIM of the upsampled images after masking out mismatched pixels between the upsampled and the full-resolution images due to dynamic objects, plot the numerical quality as a *distribution*, and add our results quality to the same chart. Our results were computed on the test dataset from our Unity scenes (600 test frames per scene), reported as a box and whisker chart in Figure 11. While it is not a direct comparison (and generally it is impossible to compare the methods on the same scene), we believe this experiment can suggest that the quality ballparks of our method and DLSS are comparable.

4.4 Limitations and Future Work

Our method learns a mapping between the low-resolution multi-frame inputs and the high-resolution output image. As a regression system, the method tends to gracefully degrade in quality outside of the trained domain, similarly to existing neural network approaches. If the multi-frame inputs contain too little information of a scene

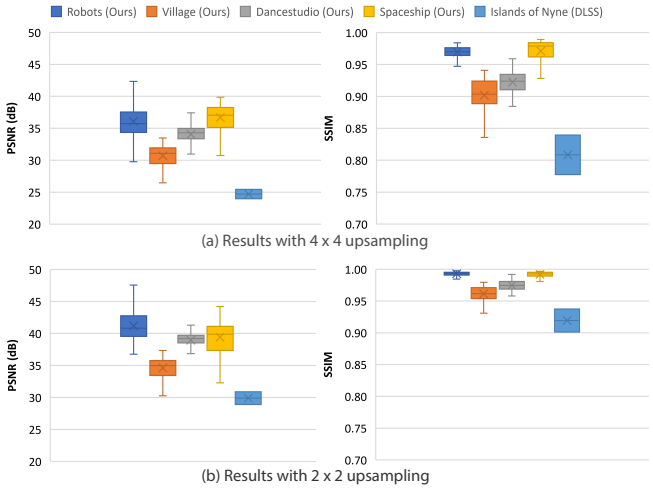

Fig. 11. Discussion figure for DLSS quality. The content for DLSS and for our method is different, and we were able to capture only a few pixel accurate image pairs, so a direct comparison is not possible. This plot is meant to illustrate that the quality ballparks for both methods are comparable. Box and whisker chart of the quality for (a) 4×4 upsampling and (b) 2×2 upsampling, by our method and DLSS on different scenes.

Fig. 12. Failure case. When the multi-frame inputs do not contain enough information of the underlying scene, i.e., the text on the sticky note, the method fails to recover the full details of the text.

region, e.g., due to extreme aliasing, the method would fail to recover the missing scene details, as the example shown in Figure 12.

While our method provides a significant improvement for neural supersampling, the current implementation is still expensive for high-resolution display applications. However, we believe the method can be significantly faster with further network optimization, hardware acceleration and professional grade engineering.

As a future research direction, we believe it would be interesting to investigate a more powerful spatio-temporal loss function. Our method, while providing a significant improvement in temporal stability compared to previous work, uses an image-to-image loss trying to match each frame as closely as possible. However, there might be a better trade-off in spatio-temporal reconstruction quality when considering temporal consistency. While promising, this is an active area of research and the best perceptual trade-off between spatial and temporal artifacts still remains an open question.

5 CONCLUSION

We have presented a new method for neural upsampling of rendered video content. Our method achieves a new state of the art in super-resolving undersampled videos with extreme aliasing by using a new temporal upsampling design. Our method also compares

numerically favorably to the existing state-of-the-art methods in superresolution, as well as in temporal supersampling used in game engines. We have also demonstrated the real-time performance of our method, which enables it to be used in real-time rendering applications in the future. We believe that the open design and the high-quality results reported in our method will also pave the road to a new body of work for neural supersampling in graphics.

REFERENCES

Martín Abadi et al. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/

Kurt Akeley. 1993. Reality engine graphics. In *Proceeding of Computer Graphics and Interactive Techniques (SIGGRAPH)*. 109—-116.

Jose Caballero, Christian Ledig, Andrew Aitken, Alejandro Acosta, Johannes Totz, Zehan Wang, and Wenzhe Shi. 2017. Real-time video super-resolution with spatio-temporal networks and motion compensation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4778–4787.

Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. 2017. Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 36, 4 (2017), 98:1–98:12.

Mengyu Chu, You Xie, Laura Leal-Taixé, and Nils Thuerey. 2018. Temporally coherent gans for video super-resolution (tecogan). *arXiv preprint arXiv:1811.09393* (2018).

Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. 2015. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38, 2 (2015), 295–307.

Andrew Edelsten, Paula Jukarainen, and Anjul Patney. 2019. Truly next-gen: Adding deep learning to games and graphics. In *NVIDIA Sponsored Sessions (Game Developers Conference)*.

Epic Games. 2020. Unreal engine. https://www.unrealengine.com

Weifeng Ge, Bingchen Gong, and Yizhou Yu. 2018. Image super-resolution via deterministic-stochastic synthesis and local statistical rectification. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 260.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of Artificial Intelligence and Statistics*. 249–256.

Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. 2019. Recurrent back-projection network for video super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3897–3906.

Samuel W Hasinoff, Dillon Sharlet, Ryan Geiss, Andrew Adams, Jonathan T Barron, Florian Kainz, Jiawen Chen, and Marc Levoy. 2016. Burst photography for high dynamic range and low-light imaging on mobile cameras. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 1–12.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4700–4708.

Jorge Jimenez, Jose I. Echevarria, Tiago Sousa, and Diego Gutierrez. 2012. SMAA: Enhanced subpixel morphological antialiasing. *Comput. Graph. Forum* 31, 2pt1 (2012), 355—-364.

Younghyun Jo, Seoung Wug Oh, Jaeyeon Kang, and Seon Joo Kim. 2018. Deep video super-resolution network using dynamic upsampling filters without explicit motion compensation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3224–3232.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*. 694–711.

A. S. Kaplanyan, S. Hill, A. Patney, and A. Lefohn. 2016. Filtering distributions of normals for shading antialiasing. In *Proceedings of High Performance Graphics (HPG)*. 151–162.

Anton S. Kaplanyan, Anton Sochenov, Thomas Leimkuehler, Mikhail Okunev, Todd Goodall, and Rufo Gizem. 2019. DeepFovea: Neural reconstruction for foveated rendering and video compression using learned statistics of natural videos. *ACM Trans. Graph. (Proceedings of SIGGRAPH Asia)* 38, 4 (2019), 212:1–212:13.

Brian Karis. 2014. High quality temporal anti-aliasing. In *ACM Trans. Graph. (Advances in Real-Time Rendering)*. Article 4.

Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. 2016. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1646–1654.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. 2017. Deep Laplacian pyramid networks for fast and accurate super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 624–632.

Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. 2017. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4681–4690.

Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. 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.

T. Lottes. 2009. *FXAA*. Technical Report. NVIDIA Corp. 3 pages.

Microsoft. 2019. Directx variable rate shading. https://microsoft.github.io/DirectX-Specs/d3d/VariableRateShading.html.

Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. 2019. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–14.

Nvidia Corporation. 2017–2018. TensorRT. https://developer.nvidia.com/tensorrt.

Anjul Patney, Marco Salvi, Joohwan Kim, Anton Kaplanyan, Chris Wyman, Nir Benty, David Luebke, and Aaron Lefohn. 2016. Towards foveated rendering for gaze-tracked virtual reality. *ACM Trans. Graph.* 35, 6 (2016), 179:1–179:12.

William T. Reeves, David Salesin, and Robert L. Cook. 1987. Rendering antialiased shadows with depth maps. In *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*.

Alexander Reshetov. 2009. Morphological antialiasing. In *Proceedings of High Performance Graphics (HPG)*. 109—-116.

Mehdi SM Sajjadi, Raviteja Vemulapalli, and Matthew Brown. 2018. Frame-recurrent video super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6626–6634.

Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R. Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn, and Marco Salvi. 2017. Spatiotemporal variance-guided filtering: Real-time reconstruction for path-traced global illumination. In *Proc. High Performance Graphics (HPG)*. Article 2, 2:1–2:12 pages.

Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. 2016. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1874–1883.

Rajiv Soundararajan and Alan C Bovik. 2012. Video quality assessment by reduced reference spatio-temporal entropic differencing. *IEEE Transactions on Circuits and Systems for Video Technology* 23, 4 (2012), 684–694.

Xin Tao, Hongyun Gao, Renjie Liao, Jue Wang, and Jiaya Jia. 2017. Detail-revealing deep video super-resolution. In *Proceedings of the IEEE International Conference on Computer Vision*. 4472–4480.

Unity Technologies. 2005–2020. Unity engine. http://unity3d.com.

Xintao Wang, Kelvin CK Chan, Ke Yu, Chao Dong, and Chen Change Loy. 2019. EDVR: Video restoration with enhanced deformable convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* 13, 4 (2004), 600–612.

Lance Williams. 1983. Pyramidal parametrics. In *Proceedings of the 10th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*. 1—-11.

Bartlomiej Wronski, Ignacio Garcia-Dorado, Manfred Ernst, Damien Kelly, Michael Krainin, Chia-Kai Liang, Marc Levoy, and Peyman Milanfar. 2019. Handheld multi-frame super-resolution. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–18.

Lei Yang, Diego Nehab, Pedro V. Sander, Pitchaya Sitthi-amorn, Jason Lawrence, and Hugues Hoppe. 2009. Amortized supersampling. *ACM Trans. Graph.* 28, 5 (2009), 1—-12.

Wenming Yang, Xuechen Zhang, Yapeng Tian, Wei Wang, Jing-Hao Xue, and Qingmin Liao. 2019. Deep learning for single image super-resolution: A brief review. *IEEE Transactions on Multimedia* 21, 12 (2019), 3106–3121.

Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. 2018a. Image super-resolution using very deep residual channel attention networks. In *Proceedings of the European Conference on Computer Vision*. 286–301.

Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. 2018b. Residual dense network for image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2472–2481.