Pacific Graphics 2022
N. Umetani, E. Vouga, and C. Wojtan
(Guest Editors)

Volume 40 (2021), Number 7

Classifier Guided Temporal Supersampling for Real-time Rendering

Yu-Xiao Guo Guojun Chen Yue Dong Xin Tong

Microsoft Research Asia

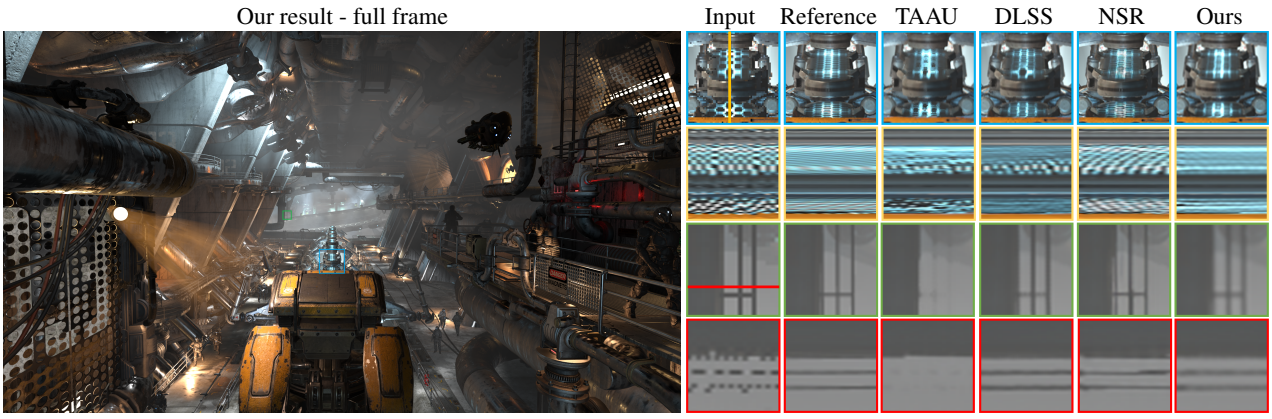

Figure 1: *Supersampling result of our method. With a low resolution aliased sample input* (1920×1080)*, our method can generate aliasing free high resolution output* (3840×2160) *while maintaining temporally stable between frames. Classical methods like TAAU [YLS20] fail to reconstruct fine details in their results (see zoom-in views in the right column, odd-rows), learning based solution [XNC*20] is slow and exhibits temporal jittering (see temporal profiles in the right column, even-rows). Our method generates results with similar quality to DLSS without requiring dedicated hardware and software.*

Abstract

We present a learning based temporal supersampling algorithm for real-time rendering. Different from existing learning-based approaches that adopt an end-to-end training of a 'black-box' neural network, we design a 'white-box' solution that first classifies the pixels into different categories and then generates the supersampling result based on classification. Our key observation is that the core problem in temporal supersampling for rendering is to distinguish the pixels that consist of occlusion, aliasing, or shading changes. Samples from these pixels exhibit similar temporal radiance change but require different composition strategies to produce the correct supersampling result. Based on this observation, our method first classifies the pixels into several classes. Based on the classification results, our method then blends the current frame with the warped last frame via a learned weight map to get the supersampling results. We design compact neural networks for each step and develop dedicated loss functions for pixels belonging to different classes. Compared to existing learning based methods, our classifier-based supersampling scheme takes less computational and memory cost for real-time supersampling and generates visually compelling temporal supersampling results with fewer flickering artifacts. We evaluate the performance and generality of our method on several rendered game sequences and our method can upsample the rendered frames from 1080P to 2160P in just 13.39*ms on a single Nvidia 3090GPU.*

Keywords: Real-time rendering, Supersampling

1. Introduction

Temporal antialiasing and supersampling [YLS20] have been widely used in real-time rendering for removing the temporal flickering and improving the image resolution. The basic idea behind this technique is to use the jittered temporal samples in previous frames to approximate the spatial samples in the current frame for antialiasing and supersampling. However, the shading variations and visibility changes between the frames make this task difficult.

Traditional methods use hand-crafted heuristics to determine the validity of history samples and composite history samples with current frame ones for rendering the final results. Although these methods can achieve a high frame rate, it is difficult to avoid artifacts such as flickering and ghosting [YLS20]. Xiao et al. [XNC*20] present a convolutional network for directly mapping a frame sequence to the current supersampled frame. It achieves good visual quality for each frame at the cost of temporal flickering. Also, the large computational cost of the network makes it difficult to be applied in real-time rendering. Thomas et al. [TVLF20] design a kernel-prediction network for temporal anti-aliasing. Without carefully designed loss functions, the black-box network is difficult to achieve a good balance between the temporal stableness and visual quality of each frame.

In this paper, we present a learning-based temporal supersampling scheme for real-time rendering. Our key observation is that the most critical task in temporal supersampling is to distinguish between pixels that have large shading variations or visibility changes and the pixels that cover high frequency content and are thus insufficiently sampled. Both kinds of pixels exhibit similar radiance changes between frames but need to be handled in a totally different way. For the first class of pixels, their historical values make little contribution to the current frame. While for the latter, their historical samples should be accumulated to resolve the aliasing in the current frame. Without carefully processing these pixels, the supersampling result will exhibit "ghosting" artifacts for the first class of pixels and temporal flickering for the ones of the second. We thus name the two classes of pixels as "ghosting" and "aliasing" respectively.

Based on this key observation, we decompose the temporal supersampling into two subtasks: pixel classification and image composition, each of which is performed by a neural network. In particular, the classification network identifies the probabilities of pixels in the current frame belonging to "ghosting" or "aliasing" classes. Based on the classification result, the image composition network blends the current frame and the warped super-sampled last frame with a predicted weight map to generate the final result. To train the two networks, we execute the underlying real-time rendering algorithm offline to render a sequence of high-resolution frames with multiple samples per pixel and obtain the ground truth class and color information of the pixels in each frame.

Compared to the end-to-end networks used in previous methods, our two-step solution provides a set of advantages. First, the simplified task in each step allows us to design a compact network for each task and thus reduces the computational and memory I/O cost of the whole solution. Second, it enables us to design different loss functions for pixels in different classes, which provides a good balance between temporal stableness and per-frame image quality. Finally, instead of taking all jittered previous frames as network input, the classification network can efficiently extract statistic information of previous frames as the network input and thus supports long jittering sequences used in current rendering algorithms. To this end, we carefully design the compact network for each task, and a set of loss functions for pixels in different classes, as well as the historical information gathered from previous frames used for the network input. We also develop an efficient training scheme for two networks used in our solution.

We evaluate our method with the frame sequences of different scenes rendered by Unreal Engine and validate its efficiency and generality. Compared to existing learning based temporal super-resolution or antialiasing solutions, our method achieves comparable per-frame image quality and better temporal stableness with much less computational and memory I/O cost.

2. Related Work

Temporal antialiasing and supersampling is an important research topic in 3D rendering. A large set of methods and commercial solutions have been developed for this task. In this section, we discuss works that are directly related to our method.

Spatial antialiasing exploits multiple samples per pixel to remove the aliasing artifacts in a single frame. To reduce the rendering cost of multiple samples, Multisampling antialiasing (MSAA) [Ake93] uses multiple samples for visibility only to reduce the rendering cost caused by multiple samples. For deferred shading based rendering, morphological antialiasing (MLAA) [Res09] and subpixel morphological antialiasing (SMAA) [JESG12] have been developed for spatial antialiasing. All these methods aim to improve the image quality of a single frame at the cost of approximating multiple samples per pixel, without considering temporal stableness.

Heuristic based temporal supersampling samples the historical frames with the help of the motion vectors for supersampling and antialiasing of the current frame. [YNS*09]. To avoid the artifacts caused by the visibility and shading changes between frames, a set of neighborhood clamping strategies [Kar14, Sal16] have been developed to correct the historical samples of a pixel based on its neighborhood values for antialiasing and supersampling. We refer the reader to a comprehensive survey [YLS20] of the heuristic based methods. Concurrent to our work, FSR 2.0 [TC22] design a set of heuristics to determine visibility induced ghosting and maintain temporal stableness on thin features. Although the speed of these methods is high, the heuristics used in these solutions are prone to error and thus lead to ghosting and other artifacts in the rendered results.

Learning based temporal supersampling learns a deep neural network for temporal supersampling or antialiasing. Chaitanya et al. [CKS*17] present a learning scheme for denoising offline ray tracing results at an interactive rate. Xiao et al. [XNC*20] design an encoder-decoder network for temporal supersampling, which takes multiple consecutive frames (4 in their implementation) as input and directly outputs the high resolution image. Although their method can well reconstruct high resolution images, the computational cost is high. Without carefully designed loss functions, their method generates temporal flickering on geometry and texture boundaries. Thomas et al. [TVLF20] introduce a kernel-prediction network for temporal anti-aliasing, where the current frame and warped last one are filtered by a set of kernels predicted by the network to generate the anti-aliased results. Although this method greatly improves computational efficiency via the quantized network, the results still suffer from temporal flickering. Nvidia re-

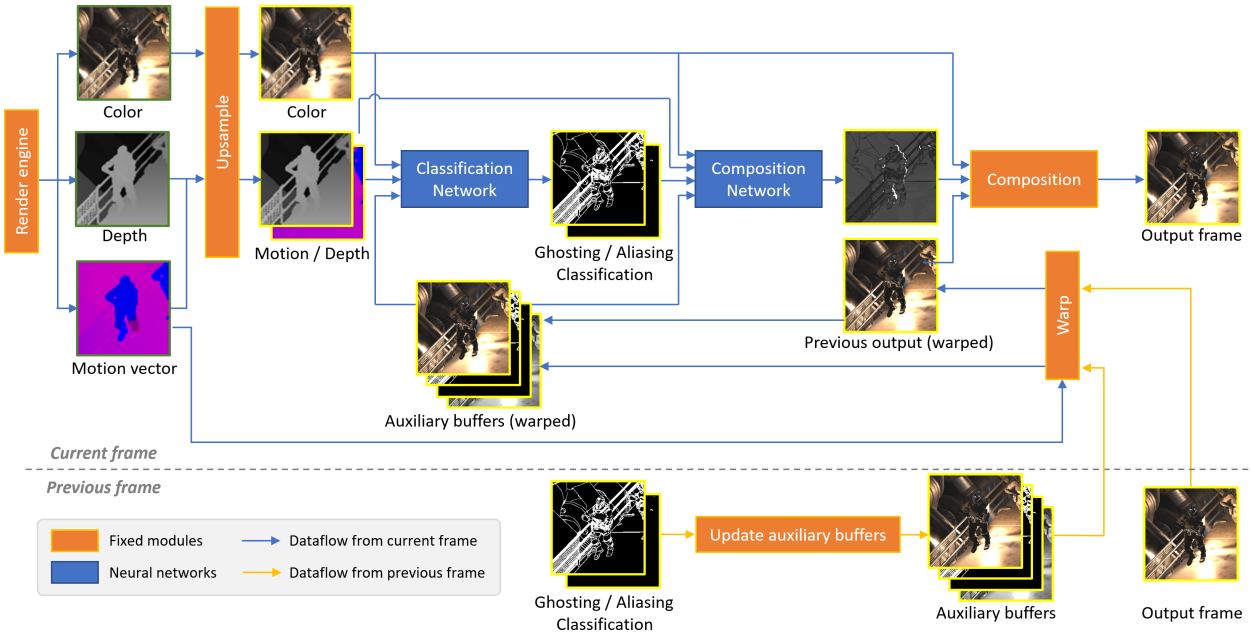

Figure 2: *Our method first classifies the pixels according to the ghosting and aliasing properties. Based on the classification results, the image composition network determines a blending weight map. The bi-cubic upsampled current frame will be blended into the warped last supersampled frame weighted by the blending weight map. The classification results will also help generate auxiliary buffers as input for the next frame. Please refer to the supplementary materials for implementation details of the 'upsample' and 'update auxiliary buffers' module.*

leased Deep Learning Super Sampling (DLSS) [Liu20] for real time supersampling and can be easily integrated into existing game engines. Although the method offers real-time performance and good super-sampling results, it requires proprietary GPU hardware (TensorCore) and the technical details of the method are unknown.

Learning based image and video superresolution have been extensively studied in the past several years [WCH20, AKB20]. Although our method shares the similar goal with the video superresolution, the two tasks are different in several ways. First, the input video superresolution is always aliasing free and low-pass filtered, while our input is point-sampled and aliased. Second, compared to video super-resolution that can take full GPU resources, our task needs to share the GPU with the real time rendering algorithm and has much less computational and memory I/O budget. Finally, different from captured videos that lack the accurate underlying scene and motion information, the rendered frames have accurate motion vectors between frames and underlying depth information of the scene, which could be used in our task.

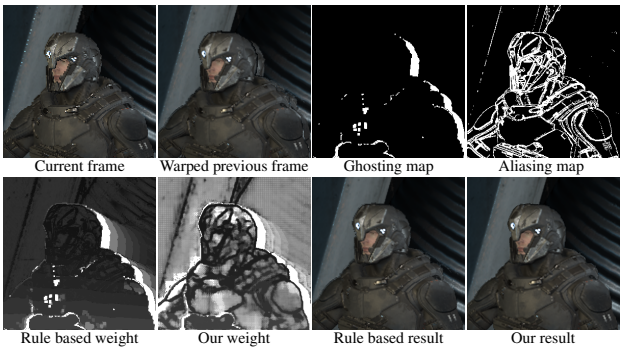

Figure 3: *Composition example. With the correct classification labels, a simple rule based composition already produces reasonable results. Our classifier guided design estimates a classification and determines the blending weight using neural networks and produces good supersampling results.*

3. Classifier guided temporal supersampling

In this section, we first give an overview of our system (Sec. 3.1), then discuss the design of our classification network (Sec. 3.2 and image composition network (Sec. 3.3. Finally, we describe the technical details of the network training (Sec. 3.4).

3.1. System Overview

As shown in Figure 2, our system takes the color buffer and several associated G-buffers from the rendering engine as the input for each frame. Specifically, the system input includes a color frame rendered with jittered point-samples, the corresponding motion vectors and the depth buffer. All of them are rendered in the low-resolution. The output of our system is an upsampled and antialiased color image for the current frame. In our implementation, we set the input resolution as 1920×1080 and the output one as 3840×2160 with two times scale up. The input of our system is the same as the commercial temporal supersampling scheme [Liu20] and thus can be easily integrated with existing real time rendering engines.

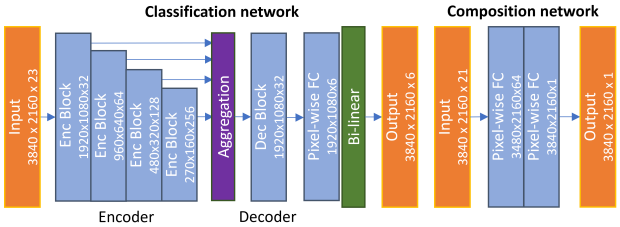

Figure 4: *The network structure of our classification and composition network. For detailed network configurations, please refer to the supplementary material.*

Given the input buffers obtained from the render engine, our system applies two networks to get the upsampled frame. Both networks are recurrent networks that take both the rendered buffers from the current frame and the warped output from the last frame as input. However, we find that the upsampled last frame is not sufficient to store historical information of multiple previous frames for pixel classification and image upsampling. Therefore, based on the classification results, we also keep a set of auxiliary buffers to store the historical information of current and previous frames. These auxiliary buffers are not only fed into the composition network for generating the supersampling output but also warped to the next frame as the input of the classification network.

To this end, the pixel classification network takes input buffers of the current frame as well as the supersampled result and auxiliary buffers warped from the last frame as the input, and outputs the pixel classification results. The high resolution auxiliary buffer is also updated based on the classification results. After that, the composition network takes the current frame buffers, warped last frame, classification results and auxiliary buffers as the input to produce a high resolution blending weight map. To generate the supersampled image result for the current frame, we first upsample the current low resolution color frame to high resolution via a bicubic interpolation and then blend it with the warped last supersampled frame according to the weight maps predicted by the composition network. The design and workflow of our system are summarized in Figure 2.

To train the two networks, we need to have the input frame, G-buffer sequences, the corresponding aliasing-free high resolution frames and the ground-truth classification labels of the high resolution frames. To generate such training data, we modify the real-time rendering engine to render a sequence of high resolution frames with multiple samples per pixel; both the color frame and G-buffers are computed. With a sufficient number of samples generated for each pixel, together with additional buffers from the rendering engine, we design a scheme to automatically generate the ground truth classification of the high resolution frames. Since the training data generation mainly consists of existing technical components and is not the technical contribution of our work, we put all technical details in the supplementary material.

3.2. Classification network

Pixel class definition The goal for the classification network is to perform two classification tasks: one for ghosting (i.e. radiance variations between two frames) and the other for aliasing. Since the radiance variation or high frequency contents in the pixel may come from different sources and has different behavior, we define three classes for each classification task.

In particular, the ghosting classifier categorizes pixels into three classes: visibility induced radiance change; shading induced radiance change and no radiance change. The visibility induced radiance change means the motion vector is pointing to a different object due to occlusion, where both the structure and the value of the history sample are untrustworthy. The shading induced radiance change indicates that the historical values of a pixel are useful but their contributions need to be computed based on the shading variation.

The aliasing classification also consists of three classes: pixels with geometry boundaries (named geometry aliasing), pixels with high frequency textures (named texture aliasing), and pixels that have no high frequency contents (named no aliasing). Each group requires different strategies for accumulating historical samples for supersampling.

Note that the classification of aliasing and ghosting are two independent tasks performed by one classification network. As a result, each pixel has both aliasing and ghosting classification results and the two classification results are not in conflict with each other.

Network input and output Both the ghosting and aliasing probability is estimated with the same network and the classification results are encoded as two 3-channel vectors indicating 3 categories for each ghosting and aliasing condition.

Similar to previous learning based supersampling methods [Liu20, XNC*20] we take the current rendered frame, *previous supersampled frame* as well as the *depth buffer* and *motion vector* as input. Specifically, we directly use the reversed-depth provided by the game engine as the depth input.

All the input frames of the classification network are high resolution frames. The low resolution rendered frame, motion vector and depth buffer are upsampled first to align with the input resolution. Similar to other temporal supersampling methods like TAAU, the motion vector is dilated by one pixel based on depth [YLS20].

Besides the input buffers rendered from the current frame, a set of auxiliary buffers that store the statistical information of historical samples warped to each pixel in the current frame are also computed and fed into the classification network, which includes

- *counter buffer* that counts the number of accumulated valid frames, in order to determine the blending weight between the current sample and accumulated samples. If the current pixel is marked as visibility induced ghosting, its counter buffer resets to 0. Otherwise, its counter is increased by 1. When feeding to the network as input, the reciprocal of the counter will be used.
- *history color range buffer* that accumulates the color variation of one surface point over time. The minimal and maximal color values are stored for each color channel. Such multiple frame statistics are crucial for separating aliasing and shading changes. For a new frame, the history color bounding box is first warped to the current frame; the bounding box is then expanded to include

the current frame sample. If the pixel is marked as visibility induced ghosting, the color range buffer will also be reset, with the bounding box set as the current pixel color value.
- *depth difference buffer* that indicates the visibility changes by comparing the depth of the current pixel and the depth of the warped previous pixel in the same time frame. To compute the depth difference, we first get the reprojected depth by following the method in [TC22] and then compute the difference between the current frame depth and the reprojected depth.
- *classification logit buffer* that stores the logits of the last frame derived by the classification network before thresholding to get the classification labels.

The impact of each auxiliary buffer on the classification is discussed in Sec. 4.2. Please refer to the supplementary material for more details about the auxiliary buffer computation.

Network structure The classification network consists of an encoder and decoder, as shown in Figure 4. The input buffers are first downsampled to half resolution by SpaceToDepth [RF17]; then, the encoder will extract the features in four different resolutions with bottleneck-liked encoder blocks. The feature maps from multiple resolutions will be fed to a single decoder block and output the classification logit at half of the output resolution. The logits are then bi-linear upsampled to high resolution as the final output.

3.3. Composition network

The composition network takes the same set of input as the classification network, except for the *depth difference buffer*. Unlike the classification network which takes the *classification logit* from the previous frame, the composition network directly uses the *classification logit* estimated for the current frame. The output of the composition network is a high resolution blending weight map. The bi-cubic interpolated current color frame and the warped last supersampled frame are composed into the final supersampling result based on the blending weight.

Since the *classification logit* and other auxiliary buffers already encode sufficient information for determining the blending weight the composition network can be very light weight. In practice, as shown in Figure 4, we use a pixel-wise MLP with two full-connected layers for the composition network, and output a $[0, 1]$ blending weight map using the Sigmoid activation.

3.4. Network Training

Training Loss The classification network is trained with the regular cross-entropy loss applied on every pixel. Since there are much fewer ghosting and aliasing pixels, we re-balance them by weighting the cross-entropy loss based on the ratio of the number of labeled pixels among all labels among all training data.

We also have one special design for the training loss to emphasize temporal stability for one special case. When the false positive ghosting happens in the aliasing region, the wrongly classified ghosting label will cause significant flickering for those aliasing pixels. In practice, we amplify the loss for this case by using the same weights as ghosting pixels, which is much larger due to the small number of ghosting pixels.

Based on different pixel classifications, the training loss for the composition network is separated into three different parts:

- For geometry aliasing pixels that are correctly classified by the classification network, we apply a $L1$-loss over the blending weight directly. The blending weight should equal the reciprocal of the sample counter to get a temporally stable result.
- For aliasing pixels that cannot be correctly classified, or false positive aliasing pixels, we regard them as unstable and do not apply any loss to them.
- The blending weights for the remaining pixels are trained with a regular $L1$ image reconstruction loss. The training loss is averaged on each part separately and summed up as the final loss.

Training process Given the training data, the two networks are trained separately. The classification network is trained first. We then train the composition network with the classification network fixed.

Since the supersampled output frame is used as an input for the next frame, as a result, both networks are trained as recurrent networks [SVB18]. The output frame and auxiliary buffers of one recurrent instance are warped to the next recurrent instance as the input. The training loss will be applied to outputs of all recurrent instances. In practice, we train eight frames recurrently at the same time.

The input of the classification network also depends on the composition network. To train the classification network without a composition network, we use a set of rules to determine the blending weights and generate the supersampled frame. Specifically, we use a 1.0 blending weight for the ghosting regions and the reciprocal of the counter for the remaining regions for this simple rule based blending.

Since the classification network is trained with rule based blending, when training the composition network, we will perform image composition twice. The rule based composition result is used for the classification network. The composition network uses superspampled results composed by the learning based blending weight.

Ideally, the classification network and composition network should be end-to-end fine-tuned. In practice, we find that the separately trained classification and composition networks already produce good quality results. Thus, we leave end-to-end training with the two networks as a future work.

4. Experiment Results

We generate the training and test dataset using a modified version of the Unreal Engine 4 [Epi20]. Cinematic sequence demos were selected in which the exact rendering sequence can be accurately repeated, thus we dump different samples by changing the viewport offset and generate rendering results with controllable sub-pixel jitter patterns.

In practice, we selected four demo games from the Unreal Engine marketplace: INFILTRATOR, SEQUENCER, ELEMENTAL and SHOWDOWN; each represents a different rendering style. We create our training dataset from the INFILTRATOR and ELEMENTAL game, 6000 and 4000 training frames are selected from each game,

	TAAU	FSR	FSR2	NSR	DLSS	Ours
ELE.	31.82	29.71	31.75	37.50	34.43	33.53
INF.	30.77	28.45	29.72	36.01	32.77	31.46
SEQ.	33.22	30.80	32.43	40.17	35.59	33.78
SHD.	36.01	31.03	35.96	41.15	36.45	37.60
Avg.	32.95	30.00	32.46	38.71	34.81	34.09

Table 1: *We compare the spatial supersampling quality by listing the PSNR among TAAU [Sal16], FSR [AMD21], FSR2 [TC22], NSR [TVLF20], DLSS [Liu20] and ours on 4 test datasets.*

	NSR	QW-Net	Ours
Computation cost (GFLOPS)	2874.904	309.966	71.435
Memory throughput (GB)	27.513	6.860	4.064

Table 2: *The network computation cost (measured by GFLOPS) and memory throughput (measured by Gigabytes) comparison among NSR [XNC*20], QW-Net [TVLF20] and our method. The metric of ours and NSR is measured for the 1080P to 2160P supersampling task, the metric for QW-Net is measured for 1080P to 1080P image reconstruction task. All numerics are based on FP32.*

respectively. For each game, we select one or two 120-frames clips for testing, the test frame is not overlapped with the training frames.

We render the ground truth images and G-buffers for calculating the classification labels using a 64-spp supersampling with a 3840×2160 resolution. The input sequence issue captured at a 1920×1080 resolution with a 128 Halton jitter sequence. The rendered linear-response HDR frames are tone mapped with the Perceptual Quantizer EOTF [ST214] before sending as input to our system; after the supersampling, the output image is reverted mapped to the linearized HDR space again.

For training, we divide the full frame images into overlapped patches with 256×256 output resolution and each patch has 8 continuous frames corresponding to the recurrent training configuration. Without proper selection, most of the patches contain few aliasing or ghosting pixels, making them inefficient for training. As a result, after randomly selecting $100K$ patches, we sort the patches according to the ratio of ghosting and aliasing pixels it contains and only leaves the first 25% patches used for our training.

We train our networks with TensorFlow [AAB*15] on a server with four Nvidia V100 GPUs. We use the Range optimizer [Wri19] with default parameters and a learning rate of 0.0005 to train both networks. The classification network is trained with 180 epochs and the composition network is trained with 120 epochs, both networks are trained with mini-batches of 24 patches.

The inference of the neural networks, image composition and auxiliary buffer computation is implemented by HLSL compute shaders, in our implementation all the shaders use FP32 precision for computation and memory storage. We test the real-time supersampling performance on a workstation with a single 3090 GPU. For each frame, our full pipeline takes 13.39ms, including 3.46ms for input preparation and 9.93ms for network inference. Even without further optimization like adopting reduced precision computation and memory access or leveraging dedicated machine learning computation hardware, our network already exceeds real-time performance and is capable of challenging tasks like supersampling for a 4K frame.

4.1. Comparisons

To validate the supersampling quality and efficiency of our system, we compare our method with state-of-the-art learning based supersampling (NSR) [XNC*20], DLSS [Liu20] and antialiasing (QW-Net) [TVLF20]; we also compare with methods without relying on deep learning, like TAAU [Kar14], FSR 1.0 [AMD21] and 2.0 [TC22].

For a fair comparison, both NSR and QW-Net are re-trained using the same training dataset as ours. Since NSR does not release their official implementation and datasets, we follow their paper and implement their method. NSR takes five consecutive frames as input, we reorganize the training and testing data to accommodate this need.

For QW-Net, we train and test their method using their official released code. QW-Net [TVLF20] is designed for the antialiasing task, which outputs the same resolution frame as the input. To keep the same input resolution, we train their network for an antiliasing task at the 1920×1080 resolution. We downsample our result to the same resolution for the comparison.

NVidia's DLSS [Liu20] supports learning based real-time supersampling, however, no public information is available for the technical details and training data for their system. It is also unclear about its computation and memory cost since the method leverages proprietary hardware like TensorCore [NVI17]. As a result, to provide a reference, we adopt the DLSS 2.3 plug-in for UE4 and generate supersampling results using the same input as our testing set.

For FSR 1.0 [AMD21] and 2.0 [TC22], the official released code is used. We provide low resolution TAA frame input for FSR 1.0 as required.

Quality We first compare the visual quality and temporal stability of the supersampled results. Figure 1 and 5 show one full resolution frame generated by our method, with cropped regions showing the comparison with existing methods. For each group of the comparison, the top row compares the reconstructed spatial details; the bottom row shows the temporal profile illustrating the temporal stability. The temporal profile is generated by warping 20 consecutive frames to the current frame with the motion vector. The vertical axis is one line of the resulting frame; the horizontal axis is the temporal dimension. Since the content is properly warped, a stable temporal profile should be composed of smooth horizontal lines. Aliasing in the temporal profile reflects visible temporal flickering. The numerical comparison on all test datasets is also provided in Table 1.

As shown in the results, supersampled frames produced by TAAU miss a lot of spatial details. Learning based methods produce good spatial reconstruction results. This is also reflected in the PSNR numerics, with TAAU having the lowest PSNR (32.95);

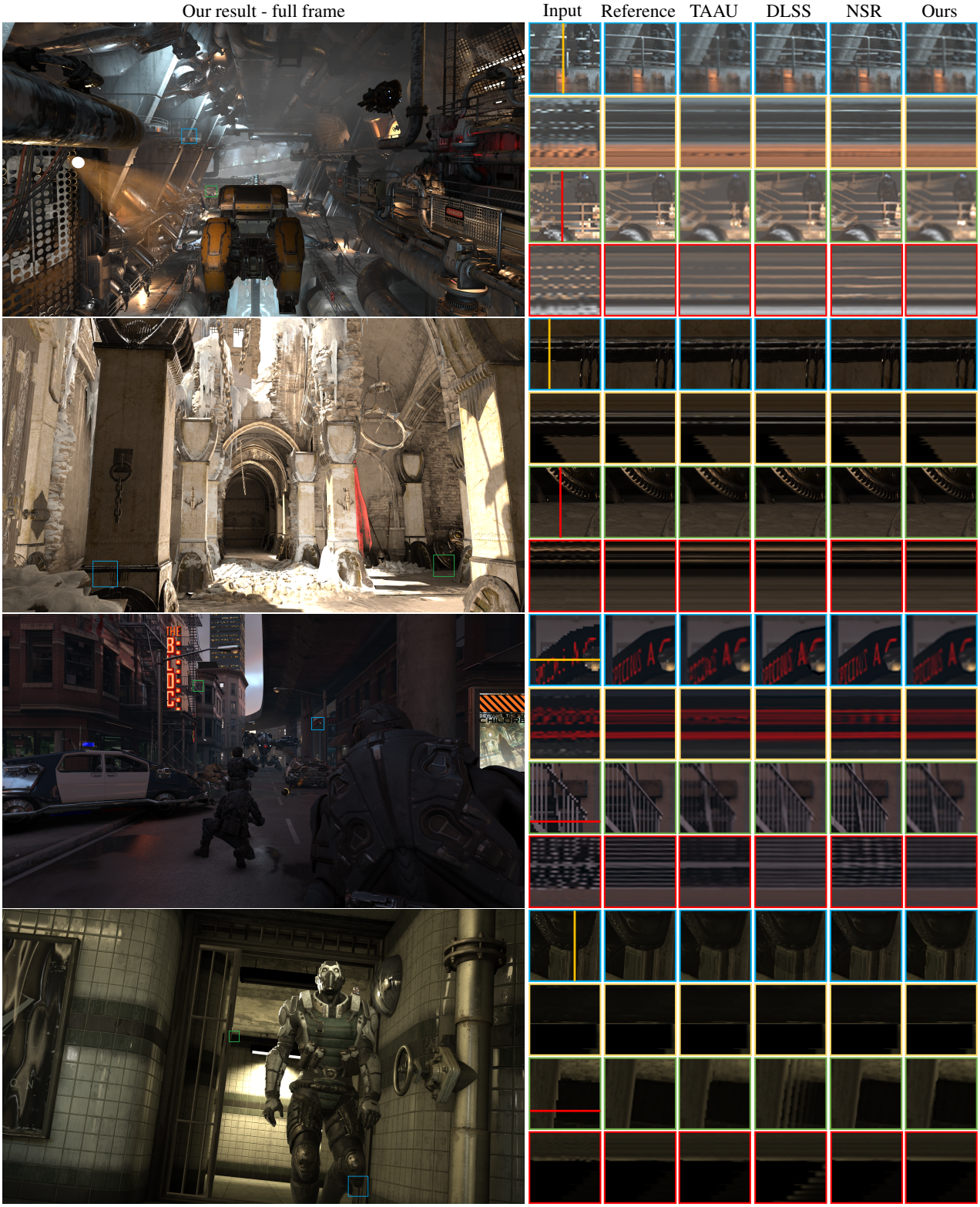

Figure 5: *Visual comparison among TAAU [Sal16], DLSS [Liu20], NSR [TVLF20] and ours on 4 testing datasets. Full frame supersampled result from our method is shown to the left and a detailed comparison is shown to the right. For each comparison group, zoom-in views are shown at the top and the temporal profiles are shown at the bottom. The vertical direction of the temporal profile represents one line in the cropped region (marked as the colored bar). The horizontal direction represents temporal change. The temporal profile is already properly warped, thus an ideal temporal stable profile should be a straight line without any aliasing as the reference dose.*

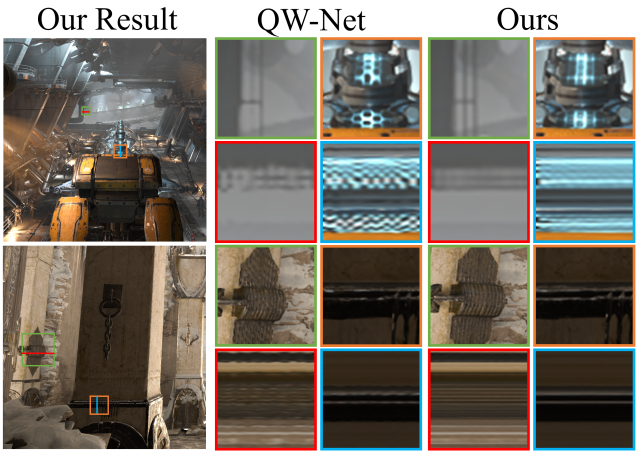

Figure 6: *Visual comparison between QW-Net [TVLF20] and ours on the* INFILTRATOR *and* ELEMENTAL. *Our result is downsampled to the same resolution as the output of the QW-net.*

Figure 7: *Visual comparison among FSR [AMD21], FSR2 [TC22] and ours on the* INFILTRATOR *and* SHOWDOWN *datasets.*

our result (34.09) is comparable to DLSS (34.81); NSR achieves the best result (38.71) with the cost of requiring a much higher computation and memory throughput. Also, notice that the result frames produced by NSR surfer from strong temporal flickering as their temporal profile shows significant aliasing artifacts. Results generated by DLSS 2.3 are temporally stable except for a highly challenging case in the INFILTRATOR test sequence shown in Figure 1. In addition, we find some ghosting artifacts in their results for the SEQUENCER sequence. Since it is difficult to judge the temporal stability by paper figure, please refer to the supplementary video to visually inspect the temporal stability.

Figure 6 compares antialiasing results by QW-Net [TVLF20] and our supersampled result downsampled to the same resolution. Our result contains more details and is also temporally stable. Results from QW-net still contain aliasing artifacts and suffer from temporal flickering.

Figure 7 compares superresolution results by FSR 1.0 [AMD21], 2.0 [TC22] and our method. Without leveraging temporal information for supersampling, the results of FSR 1.0 miss a lot of fine details. The manually designed heuristics of FSR 2.0 fail to detect some ghosting and thin feature cases, resulting in strong ghosting or aliasing artifacts. On the contrary, our method can correctly handle those cases and produces high quality results.

Performance Although achieving accurate and temporally stable supersampling results, our method also takes much less computation and memory throughput. Table 2 lists the computation cost and memory throughput for processing a single frame. Compared to our network, NSR's network [XNC*20] takes 40.2 times more computation and 6.7 times more memory throughput, making it significantly slower. The QW-Net [TVLF20] takes 6.7 times computation and 1.6 times memory. Even considering all the benefits from reduced precision computation, their network structure still takes more computation than our network. Also note that QW-net only outputs half-sized frames compared to ours.

4.2. Ablation studies

Here we validate the effect of our two-step design, class-aware training loss and auxiliary buffers by a set of ablation experiments. All the ablation experiments are trained and tested on the INFILTRATOR dataset.

Direct regression Classifying the ghosting and aliasing region is the key to our network design and training scheme. To validate the efficiency of such a design, we adapt our basic network structures into a direct regression network that produces the composition blending weight directly, without outputting a classification label. As shown in Figure 8, results generated by the Direct Weight Regression network (DWR) network exhibit a discontinuous temporal profile, indicating temporal flickering.

Training loss To validate the effect of classification guided training loss, we train the composition network with only the L1 loss. Although the same input and classification network are used, the results show that the composition network trained using L1 loss only (L1-Only) will produce temporally unstable results, as shown in Figure 8.

On the contrary, our result (baseline) produces temporal stable results with the temporal profile similar to the ground truth (Reference). Please also refer to the supplementary video to see how our two-step design and training loss improves temporal stability.

Auxiliary input Our network takes several auxiliary buffers to provide additional information for the supersampling. We test the effect of those auxiliary inputs by removing each of them and test the performance. Typical artifacts when removing each auxiliary input are shown in Figure 9.

The *counter buffer* helps the refinement network correctly apply the proper blending weights, especially around aliasing regions. Without the counter buffer (No-Ctr) the result shows strong artifacts around the object boundaries where aliasing occurs.

The *depth difference buffer* helps distinguish between geometry

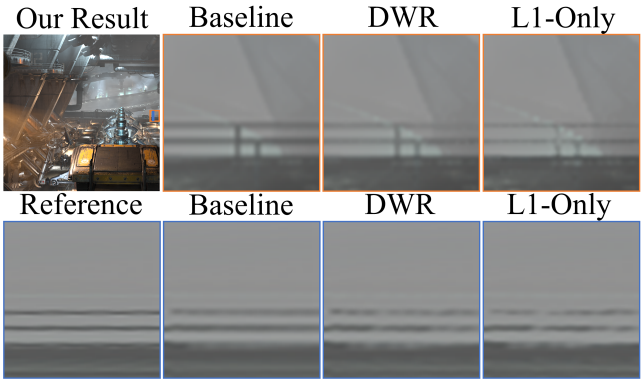

Figure 8: *Visual comparison of results generated by our method (baseline), direct weight regression network (DWR) and composition network trained with only L1 loss (L1-only). A 64SPP reference is also included. The top row shows the spatial details with zoom-in view, the bottom row shows temporal profile illustrating temporal stability.*

Trained on	ELE.	INF.	SEQ.	SHD.	Avg.
INF.&ELE.	33.53	31.46	33.78	37.60	34.09
All 4 datasets	33.65	31.55	33.89	37.75	34.21

Table 3: *We validate the generality of our method by training on two datasets and tested on four datasets. The networks trained on two datasets produce similar PSNR results as the networks trained on all four datasets.*

aliasing and geometry aliasing. Without the depth difference buffer (No-DD) the result produces artifacts around thin features where geometry ghosting and aliasing happen in nearby pixels.

The *history color range buffer* helps distinguish between shading changes and aliasing regions. Without the history color range (No-HCR) the network failed to detect rapid shading change regions, causing incorrect shading results.

The *classification logits buffer* helps identify geometry aliasing away from geometry ghosting. Without the classification logits buffer (No-Logit), the result will mislabel geometry ghosting in aliasing regions, leading to aliasing or flickering.

The *motion vector buffer* contains a clear boundary between foreground and background, which helps determine the ghosting. Without the motion vector buffer (No-MV), the result will lag and blur behind the moving foreground objects.

Generality To validate the generality of our method, we train our model on the INFILTRATOR and ELEMENTAL datasets and test its generality on the other two datasets captured from SHOWDOWN and SEQUENCER. For reference, we also train our networks with training data from all four of them. Table 3 shows the numerical results, the networks trained on all four datasets only introduce marginal quality improvements, while the networks trained with only two datasets generalized well on rendering frames from different games.

Figure 9: *Ablation experiments by removing each auxiliary buffer from the input. We remove the depth difference buffer (No-DD), counter buffer (No-CTR), history color range buffer (No-HCR), classification logit buffer (No-CL) and motion vector (No-MV) from the input and retrain the networks. Here, we show supersampling results from those variants and compare with results produced by our method using all auxiliary buffers (baseline). The results show typical artifacts when removing each of the auxiliary buffers, while our method produces reasonable results similar to the reference.*

5. Conclusion

We introduce a learning based temporal supersampling system guided by the classification of pixel contents. The temporal supersampling task is decomposed into the pixel classification and image composition task, with each task performed by a neural network. Our method significantly improves the temporal stability of learning based supersampling methods, and achieves comparable image reconstruction quality to existing methods using much fewer computation and memory costs.

Limitation and future works Our automatic data labeling scheme cannot generate correct ghosting classification labels for video textures and transparent scenes, and affects the classification network training. Figure 10 shows one result with video texture, some ghosting artifacts occur due to the classification network failing to detect this ghosting case.

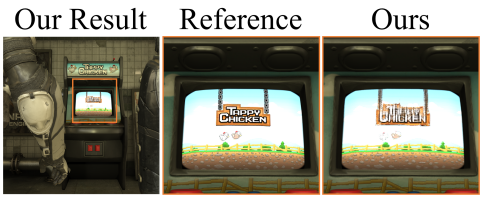

Figure 10: *Our data labeling scheme cannot correctly label the ghosting pixels introduced by video textures, the trained supersampling network produces ghosting artifacts.*

The supersampling output of our method is a per-pixel blending between the previous frame and the bi-cubic upsampled current frame. Such low-pass filtering of the upsampling introduced a slight blur in the final result. In the future, we will explore improving the results' sharpness by replacing the bi-cubic upsampling with a learned upsampling kernel.

Although our network achieves real-time performance on challenging supersampling conditions that outputs a 4K image. Our shader implementation is still far from fully optimized. Our network is still based on FP32 memory storage and computation. Leveraging reduced precision network design [TVLF20] can further improve the speed. Acceleration with dedicated neural compute units like TensorCore or XMX hardware is also an interesting future work.

The auxiliary buffers act an important role to encode critical history information for supersampling. Currently, our auxiliary buffer is manually designed. In the future, the auxiliary buffer can be automatically learned, which will be more effective and also efficient.

Our network structure is manually designed for achieving real-time performance on the current GPU hardware. Different hardware or different supersampling configurations might need manually adjusting the network structures. Combining the classifier guided network design with neural network architecture search is also one promising direction that leads to more efficient networks for various supersampling tasks on future hardware.

Acknowledgments

We would like to thank the reviewers for their constructive feedback; Chong Zeng, Jilong Xue, Yuqing Xia, Wei Cui and the NNFusion [MXY*20] team's support for optimizing the HLSL shader code; Youkang Kong's help implementing the NSR method [XNC*20]. We would also like to thank people from the Microsoft Xbox and xCloud team for valuable discussions, including Daniel Kennett, Hoi Vo, Matt Bronder and Andrew Goossen.

References

[AAB*15] ABADI M., AGARWAL A., BARHAM P., BREVDO E., CHEN Z., CITRO C., CORRADO G., DAVIS A., DEAN J., DEVIN M., GHEMAWAT S., GOODFELLOW I., HARP A., IRVING G., ISARD M., JIA Y., JOZEFOWICZ R., KAISER L., KUDLUR M., LEVENBERG J., MANÉ D., MONGA R., MOORE S., MURRAY D., OLAH C., SCHUSTER M., SHLENS J., STEINER B., SUTSKEVER I., TALWAR K., TUCKER P., VANHOUCKE V., VASUDEVAN V., VIÉGAS F., VINYALS O., WARDEN P., WATTENBERG M., WICKE M., YU Y., ZHENG X.: Tensorflow: Large-scale machine learning on heterogeneous distributed systems, 2015. URL: http://download.tensorflow.org/paper/whitepaper2015.pdf. 6

[AKB20] ANWAR S., KHAN S., BARNES N.: A deep journey into super-resolution: A survey. *ACM Computing Surveys (CSUR) 53*, 3 (2020). 3

[Ake93] AKELEY K.: Reality engine graphics. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (1993), pp. 109–116. 2

[AMD21] AMD: Fidelityfx super resolution. In *GPU Open* (2021). 6, 8

[CKS*17] CHAITANYA C. R. A., KAPLANYAN A. S., SCHIED C., SALVI M., LEFOHN A., NOWROUZEZAHRAI D., AILA T.: Interactive reconstruction of monte carlo image sequences using a recurrent denoising autoencoder. *ACM Trans. Graph. 36*, 4 (jul 2017). 2

[Epi20] EPIC GAMES: The unreal engine 4. https://www.unrealengine.com/, 2020. 5

[JESG12] JIMENEZ J., ECHEVARRIA J. I., SOUSA T., GUTIERREZ D.: Smaa: Enhanced subpixel morphological antialiasing. In *Computer Graphics Forum* (2012), vol. 31, Wiley Online Library, pp. 355–364. 2

[Kar14] KARIS B.: High quality temporal supersampling. In *SIGGRAPH 2014 Advances in Real-Time Rendering in Games course.* (2014). 2, 6

[Liu20] LIU E.: Image reconstruction for real-time rendering with deep learning. In *GPU Technology Conference (GTC)* (2020). 3, 4, 6, 7

[MXY*20] MA L., XIE Z., YANG Z., XUE J., MIAO Y., CUI W., HU W., YANG F., ZHANG L., ZHOU L.: Rammer: Enabling holistic deep learning compiler optimizations with rtasks. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)* (Nov. 2020), USENIX Association, pp. 881–897. 10

[NVI17] NVIDIA: Discover how tensor cores accelerate your mixed precision models. https://developer.nvidia.com/tensor-cores, 2017. 6

[Res09] RESHETOV A.: Morphological antialiasing. In *Proceedings of the Conference on High Performance Graphics 2009* (2009). 2

[RF17] REDMON J., FARHADI A.: Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 7263–7271. 5

[Sal16] SALVI M.: An excursion in temporal supersampling. In *Game Developer's Conference (GDC)* (2016). 2, 6, 7

[ST214] St 2084:2014 - smpte standard - high dynamic range electro-optical transfer function of mastering reference displays. *ST 2084:2014* (2014), 1–14. doi:10.5594/SMPTE.ST2084.2014. 6

[SVB18] SAJJADI M. S., VEMULAPALLI R., BROWN M.: Frame-recurrent video super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 6626–6634. 5

[TC22] THOMAS A., COLIN R.: Fidelityfx super resolution 2.0. In *Game Developer's Conference (GDC)* (2022). 2, 5, 6, 8

[TVLF20] THOMAS M. M., VAIDYANATHAN K., LIKTOR G., FORBES A. G.: A reduced-precision network for image reconstruction. *ACM Trans. Graph. 39*, 6 (nov 2020). 2, 6, 7, 8, 10

[WCH20] WANG Z., CHEN J., HOI S. C.: Deep learning for image super-resolution: A survey. *IEEE transactions on pattern analysis and machine intelligence 43*, 10 (2020), 3365–3387. 3

[Wri19] WRIGHT L.: Ranger - a synergistic optimizer. https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer, 2019. 6

[XNC*20] XIAO L., NOURI S., CHAPMAN M., FIX A., LANMAN D., KAPLANYAN A.: Neural supersampling for real-time rendering. *ACM Trans. Graph. 39*, 4 (jul 2020). 1, 2, 4, 6, 8, 10

[YLS20] YANG L., LIU S., SALVI M.: A survey of temporal antialiasing techniques. *Computer Graphics Forum 39*, 2 (2020), 607–621. 1, 2, 4

[YNS*09] YANG L., NEHAB D., SANDER P. V., SITTHI-AMORN P., LAWRENCE J., HOPPE H.: Amortized supersampling. *ACM Transactions on Graphics (TOG) 28*, 5 (2009), 1–12. 2