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 Supplementary material

Yu-Xiao Guo Guojun Chen Yue Dong Xin Tong

Microsoft Research Asia

In this supplementary material, we include additional implementation details of our system. In Section 1, we include detailed descriptions of the network design of our classification and composition network. In Section 2, we introduce how we automatically generate the classification labels for the training. In Section 3, we provide additional details for how we process the auxiliary buffers.

1. Network structures

1.1. Classifier network

The classification network consists of an encoder and decoder, as shown in Figure 1.

The encoder will extract the features in 4 different resolutions. For each resolution, we used a bottleneck-like convolution structure which is composed of a 3×3 depth-wise convolution and two 1×1 channel-wise convolution before and after the depth-wise one. We only apply Batch Norm and ReLU after the two 1×1 convolution layers. The output features will be evenly split into two branches: one for the decoder and the other for the next resolution encoder. We use max-pooling to down-sample the resolution and double the filters while the resolution is halved. Thus, after a set of convolution and downsampling, we get a series of feature maps in 4 different scales.

A *DepthToSpace* (also known as Periodic Shuffle or Pixel Shuffle) [SCH*16] operation merges multiple low resolution feature maps into one high resolution feature map, so that all the resulting feature maps will have the same spatial resolution as the output of the first convolution blocks. In our design, the convolutional filters are $32, 64, 128$ and 256 in 4 different scales, which will contribute the output channels of the encoder to $16, 8, 4$ and 4, respectively. Thus, we will get a 32 channels feature as the input of the decoder.

Our decoder consists of 3 continuous pixel-wise MLP with BatchNorm and ReLU. Then, an extra pixel-wise MLP with BatchNorm and Softmax makes the output channels 6 (3 labels for ghosting and 3 labels for aliasing). For computation-cost consideration, the classifier works on the half resolution compared with the output resolution. To this end, we first apply *SpaceToDepth* [RF17] to half the resolution of inputs and quadruple the number of channels, then using bi-linear interpolation to double the output logit resolution to align with the supersampled image.

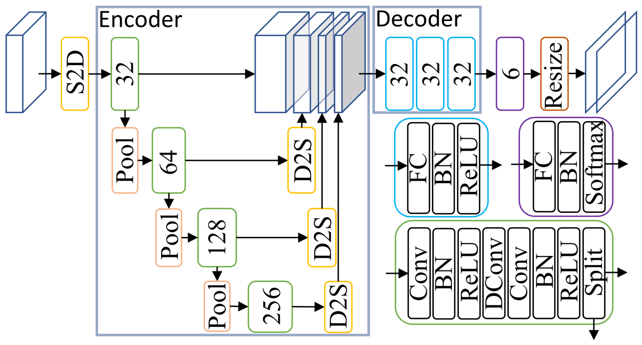

Figure 1: *The network structure of our classifier design.* ***S2D*** *is short for SpaceToDepth;* ***D2S*** *is short for DepthToSpace;* ***Conv*** *is short for channel-wise convolution;* ***DConv*** *is short for depth-wise convolution;* ***FC*** *is short for pixel-wise full connected layer; For all convolutional blocks, we zoom out its detail structure in the right-bottom and denote the number of output channels.*

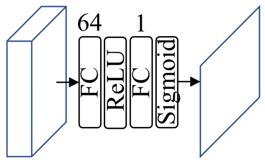

Figure 2: *The network structure of our refinement network design.* ***FC*** *is the pixel-wise full connected layer.*

1.2. Composition network

We use two continuous pixel-wise MLP as the backbone for the composition network, shown as Figure 2. The first 64-channels MLP is activated with ReLU. And the second single-channel MLP is using Sigmoid activation to output the single channel blending weight and ensure its value between $[0, 1]$.

2. Automatic training data labeling

A ghosting pixel is a pixel with its temporal samples no longer valid and may cause ghosting if the temporal same are incorrectly used. There are three sources that cause a ghosting pixel: visibility change, rapid shading change and scene cut. Where the scene cut is already labeled by the game engine.

Visibility induced ghosting Ideally, visibility induced ghosting means the history samples are sampled from a different object compared to the current foreground object. To compensate the motion of the object, we calculate the original position of the current frame f pixel x,y in the previous frame $f-1$, and determine the visibility induced ghosting label R_G based on the position difference. However, to improve the robustness we also compare the velocity difference since different objects tend to have different view space velocity:

$$\frac{\left(D_{max}(\mathcal{W}_{f\to f-1}(x,y),f-1)-D_{max}(x,y,f-1)\right)}{D_{max}(\mathcal{W}_{f\to f-1}(x,y),f-1)} > T_D^G \quad (1)$$

$$||V(x,y,f)-V(\mathcal{W}_{f\to f-1}(x,y),f-1)||_2^2 > T_V^G \quad (2)$$

Note that throughout this paper, the depth D is actually the reversed-depth commonly used in modern game engines such as UE4 [Epi20], so that a larger reversed-depth value indicates the object is closer to the camera.

The criteria above became unstable in the object boundary where the depth within a pixel changes in a large range. As a result, we dilate the visibility induced ghosting label by one pixel to neighboring pixels (x',y') follows both the two criteria:

$$\frac{\left(D_{max}(x,y,f)-D_{max}(x',y',f)\right)}{D_{max}(x,y,f)} > T_D^G \quad (3)$$

$$||V(x,y,f)-V(x',y',f)||_2^2 > T_V^G \quad (4)$$

where $T_D^G = 0.01$ is a pre-defined threshold of depth, $T_V^G = 1.0$ is a pre-defined threshold of velocity, $\mathcal{W}_{f\to f-1}(x,y)$ indicates the frame space position of the f-th frame pixel x,y in the $(f-1)$-th frame.

Shading ghosting A straightforward way to detect rapid shading change is comparing the pixel color. If the pixel color in the current f-th frame is different from is history buffer value by a margin larger than the pre-determined threshold T_I^G, we regard it as a shading ghosting pixel R_S:

$$\left(I_{max}(x,y,f)-I_{max}(\mathcal{W}_{f\to f-1}(x,y),f-1)\right) > T_I^G \quad (5)$$

The criterion above will be affected by high frequency textures and normal maps, where a single value $I_{max}(x,y)$ cannot represent all the samples within this pixel. As a result, we introduced another criterion to further filter out pixel unstable color changes caused by spatially high frequency textures.

$$IoU\left(\mathbf{I}(x,y,f),\mathbf{I}(\mathcal{W}_{f\to f-1}(x,y),f-1)\right) < T_{IoU}^G \quad (6)$$

Here, we compare the intersection over union (IoU) of the color box of the current pixel and its corresponding history pixel. If the pixel has spatially smooth shading, it has a smaller color box, yet a smaller IoU, while pixels with large spatial variance have a large color box and lager IoU, which will be discarded.

To remove small regions and thin features marked as shading ghosting, which are usually prone to flickering. We apply morphological opening (erosion followed by dilation) twice to the shading ghosting label map.

Table 1: *List of symbols commonly used in this paper.*

Symbol	Meaning
$\mathbf{D}$	Depth[†] range of all samples in a pixel.
$\mathbf{I}$	Color range of all samples in a pixel.
$\mathbb{N}$	Set of all normal samples in a pixel.
D_{min}, D_{max}	Minimal and maximal depth in range $\mathbf{D}$.
I_{min}, I_{max}	Minimal and maximal color value in $\mathbf{I}$.
C	Number of accumulated valid frames.
$\mathcal{W}_{i\to j}(x,y)$	Warp the pixel x,y in the i-th to j-th frame.
A	Aliasing map
A_G	Geometry aliasing map
A_T	Texture aliasing map
R	Ghosting map
R_G	Visibility induced ghosting map
R_S	Shading ghosting map

† Reversed-depth are used instead of conventional depth.

Aliasing pixels cover regions with high frequency spatial variations, caused by geometry boundaries, high frequency normals and textures. A high number of valid samples are needed to correctly reconstruct the correct pixel value of an aliasing pixel.

We follow antialiased deferred rendering [NVI14] to determine the aliasing pixel by computing the depth, normal and color variation within one pixel and mark normal aliasing pixel with the thresholds $T_D^A = 0.01$, $T_N^A = 0.5$ and $T_I^A = 0.5$.

Specifically, we mark pixels as aliasing when they meet one of the following three criteria:

$$\frac{\left(D_{max}(x,y,f)-D_{min}(x,y,f)\right)}{D_{max}(x,y,f)} > T_D^A \quad (7)$$

$$\max_{i\in\mathbb{N}(x,y)} ||N_0(x,y,f)-N_i(x,y,f)||_1 > T_N^A \quad (8)$$

$$\max_{r,g,b}\left|\frac{\left(I_{max}(x,y,f)-I_{min}(x,y,f)\right)}{I_{max}(x,y,f)}\right| > T_I^A \quad (9)$$

Where $\mathbb{N}(x,y)$ is the set containing all normal samples within the pixel (x,y), N_0 is the first sample in the set. Ideally, the normal variation should be compared to the total combination between any pair of samples. However, in practice, we find that only calculating the normal variation against the first sample works as a good approximation while saving a lot of computation.

To separately label geometry aliasing and texture aliasing, we first compute the geometry aliasing A_G following equation 7 and 8 based on rendered depth and normal map without using normal map. Then the complete aliasing map A is computed following equation 7, 8 and 9, with rendered depth, normal and color buffer, with normal map enabled. Finally, the texture aliasing A_T map is determined by subtracting geometry aliasing A_G from the complete aliasing map A.

Shading ghosting label correction The shading ghosting label is computed based on the motion vector and is supposed to compare values for one surface point. When there is a visibility change, the motion vector is no longer valid, thus a visibility induced ghosting should not be marked as shading ghosting.

Shading ghosting labels are also affected by the aliasing, when the content has strong aliasing, the shading ghosting label also becomes unstable. As a result, we will remove the shading ghosting label when the same pixel is marked as visibility induced ghosting or aliasing.

3. Computing the auxiliary buffers

All the auxiliary buffers have the same resolution as the output image, if the original buffer is low resolution, it needs to be upsampled to the output resolution first. For each auxiliary buffer, we provided a detailed description as follows:

Current frame color buffer Similar to TAA [YLS20], we perform unjitter for the current frame rendered color buffer, and bicubic interpolation into the output resolution.

Current frame depth and motion vector Since the depth and motion vector cannot be interpolated, they are upsampled to the output resolution using nearest neighbor sampling. Similar to TAA [YLS20], we dilate the foreground objects by 1 pixel when sampling motion vectors.

Counter buffer The counter buffer is computed in the output resolution based on the high resolution classification label. The counter buffer is initialized as 0 for all the pixels and the counter increases by 1 for every frame; unless the pixel is classified as visibility induced ghosting, then the counter is reset to 0. The counter from the previous frame is warped to the current frame as input using nearest sampling.

History color range buffer For each color channel in RGB, a minimal and a maximal value is stored as the history color range, resulting in a 6 channels buffer. The history color range buffer computed for the previous frame is warped to the current frame using the nearest sampling.

Depth difference buffer The depth difference buffer measures the normalized difference between the current depth D_{cur} and the reprojected depth D_{rp} [TC22] follows:

$$\frac{D_{rp}(\mathcal{W}_{f\to f-1}(x,y)) - D_{cur}(x,y)}{\texttt{max}(D_{rp}(\mathcal{W}_{f\to f-1}(x,y)), D_{cur}(x,y))} \tag{10}$$

where the depth is warped with nearest neighbor sampling.

Classification logit The classification logit buffer has 6 channels, standing for: visibility induced ghosting, shading ghosting, no-ghosting, geometry aliasing, texture aliasing and no-aliasing. The high resolution classification logit is warped from the previous frame to the current frame using bi-linear interpolation.

References

[Epi20] EPIC GAMES: The unreal engine 4. https://www.unrealengine.com/, 2020. 2

[NVI14] NVIDIA GAMEWORKS: Antialiased deferred rendering. https://docs.nvidia.com/gameworks/content/gameworkslibrary/graphicssamples/d3d_samples/antialiaseddeferredrendering.htm, 2014. 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. 1

[SCH*16] SHI W., CABALLERO J., HUSZÁR F., TOTZ J., AITKEN A. P., BISHOP R., RUECKERT D., WANG Z.: 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* (2016), pp. 1874–1883. 1

[TC22] THOMAS A., COLIN R.: Fidelityfx super resolution 2.0. In *Game Developer's Conference (GDC)* (2022). 3

[YLS20] YANG L., LIU S., SALVI M.: A survey of temporal antialiasing techniques. *Computer Graphics Forum 39*, 2 (2020), 607–621. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14018, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14018, doi:https://doi.org/10.1111/cgf.14018. 3