IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 26, Issue 5, Ser. 2 (Sept. – Oct. 2024), PP 55-61
www.iosrjournals.org

Advancing Photorealism Enhancement Using Convolutional Neural Networks (CNNs)

Dr. (Mrs.) N.F. Shaikh, Dr. (Mrs.) S.K. Wagh, Devesh Sandeep Singh

Abstract
Photorealism has been the defining goal of computer graphics for half a century. A look at even the most sophisticated real-time games will quickly reveal that photorealism has not been achieved. An ineffable difference in the appearance of simulation and reality remains. In recent years, a complementary set of techniques has been developed in computer vision and machine learning. These techniques, based on deep learning, convolutional networks, and adversarial training, bypass physical modeling of geometric layout, material appearance, and light transport. Instead, images are synthesized by convolutional networks trained on large datasets.
These techniques have been used to synthesize representative images from a given domain to convert semantic label maps to photographic images and to attempt to bridge the appearance gap between synthetic and real images. The images are enhanced by a convolutional network that leverages intermediate representations produced by conventional rendering pipelines. The network is trained via a novel adversarial objective, which provides strong supervision at multiple perceptual levels.
Index Terms—*Photorealism enhancement, Photorealism, Image-to-Image translation, Style transfer*

Date of Submission: 18-09-2024 Date of Acceptance: 28-09-2024

I. Introduction

Photorealism has been the defining goal of computer graphics for half a century. In 1977, Newell and Blinn [1] surveyed a decade of work on this problem. In the ensuing four decades, substantial further progress has been made, due in part to the physical light transport [2], principled representation of material appearance [3], [4], and photogrammetric modeling [5]. These techniques and their approximations have been integrated into real-time rendering pipelines, stantially advancing the realism of computer games [6]. Nevertheless, a look at even the most sophisticated real-time games will quickly reveal that photorealism has not been achieved. An ineffable difference in the appearance of simulation and reality remains. In recent years, a complementary set of techniques has been developed in computer vision and machine learning. These techniques, based on deep learning, convolutional networks, and adversarial training, bypass physical modeling of geometric layout, material appearance, and light transport. Instead, images are synthesized by convolutional networks trained on large datasets. These techniques have been used to synthesize representative images from a given domain to convert semantic label maps to photographic images and to attempt to bridge the appearance gap between synthetic and real images. Images synthesized by these approaches capture aspects of photographic appearance that often elude even state- of-the-art computer games are largely disconnected from the rendering pipelines that drive computer games, can be hard to control, and often produce jarring artifacts that would be unacceptable in production quality media.

In this work, we take a step towards melding these two complementary routes to photorealism. We seek to build on the infrastructure developed in the production of modern games and enhance their photorealism via techniques developed in the deep learning community. Our starting point is a set of intermediate buffers (G-buffers) produced by game engines during the rendering process [6]. These buffers provide de- tailed information on geometry, materials, and lighting in the scene. We train convolutional networks with these auxiliary inputs to enhance the realism of images produced by the rendering pipeline. To integrate these buffers into the photorealism enhancement flow, we design new network components that modulate features from a rendered image according to information extracted from the buffers. We compare the presented approach against a broad array of strong baselines that represent diverse perspectives on photorealism enhancement. We also conduct a perceptual experiment to assess photorealism.

A. Motivation

Image-based rendering techniques can produce results that are indistinguishable from photographs by recycling real imagery of a scene for rendering novel views. However, they require capturing photos of the scene of interest beforehand and make it difficult to manipulate captured scenes afterward. Furthermore, novel

views need to be fairly close to the prerecorded camera trajectory to avoid artifacts.

II. Literature Survey

A. Literature Survey

Many approaches to image synthesis or translation have employed adversarial objectives, which involve a discriminator network that evaluates the realism of generated images. The discriminator is commonly trained alongside a network performing the image generation. One intention of this setup is for the discriminator to learn high-level semantic concepts to provide high-quality supervision to the generator network. However, with a simple binary classification objective, the discriminator may focus on low-level textures and patches instead, since these already provide discriminative features. To direct attention to high level semantic content, a number of modifications have been proposed. For example, the binary real vs. fake decision can be accompanied by a classification objective Additional semantic segmentation map can be con-catenated to the input image projected to a high-dimensional feature space [8] processed via a separate network stream, or guide an auxiliary classifier.

III. Methodology

Overview

Fig. 1 provides an overview of our approach. Our method consists of an image enhancement network, which takes as input a rendered image and outputs an enhanced image. To facilitate the enhancement, we provide additional inputs to the network. Specifically, we extract intermediate rendering buffers (G-buffers) from the graphics pipeline. These G-buffers provide information on the geometry, materials, and lighting in the scene. They are processed by a G-buffer encoder network, which outputs G-buffer features at multiple scales. G- buffer features are then provided as input to the image enhancement network, where they modulate image features.

The image enhancement network is based on HRNetV2, which demonstrated strong performance on a variety of dense prediction tasks. The HRNet processes an image via multiple branches at different resolutions. Importantly, one feature stream is kept at relatively high resolution to preserve fine image structure. We modify the HRNet architecture as follows. First, we replace the initial stride convolutions by regular convolutions to have the network operate on the full resolution and preserve even finer detail. Second, within the residual blocks in each branch we replace the batch normalization layers by rendering- aware denormalization (RAD) modules. The modified blocks modulate the feature streams based on information extracted from the G-buffers.

Extraction of G-Buffers

The G-buffers we extract mix one-hot encodings for mate- rial information, dense continuous values for normal, depth, and color, and sparse continuous information for bloom and sky buffers. For some image regions the G-buffers are zero, depending on the rendered content. Few objects are transpar- ent, and sky regions, for example, contain neither geometry nor material information.

To account for the different types of data in the G-buffers, we process the G-buffers via a G-buffer encoder (Fig. 2). The G-buffer encoder consists of multiple network streams that process the same set of G-buffers. We fuse the streams based on masks derived from the semantic segmentation maps. This way, the streams can map information from the G-buffers differently for certain types of objects.

We base our work on the popular game Grand Theft Auto

To obtain G-buffers from the game, we follow recent approaches to extracting rendering resources from computer games. Specifically, we extract G-buffers that provide infor- mation about geometric structure (surface normals, depth), materials (shader IDs, albedo, specular intensity, glossiness, transparency), and lighting (approximate irradiance and emis- sion, sky, bloom), illustrated in Fig. 3. We further augment this set with two quantities we derive from the G-buffers. First, we reflect for each pixel the view vector at the surface normal to obtain a reflection vector. Second, we compute the dot product between the surface normal and the reflection vector.

Perceptual Discriminator

During training of the image enhancement network, a perceptual discriminator evaluates the realism of enhanced images. It consists of a robust semantic segmentation network, a perceptual feature extraction network, and multiple dis- criminator networks (Fig. 4). We employ MSeg for semantic segmentation and VGG-16 for perceptual feature extraction. Both networks are pretrained and are not optimized during training of the image enhancement network. We apply the segmentation network to real images from a target dataset and unmodified rendered images. This provides compatible semantic information for real and synthetic images. It also enables training on datasets without ground-truth annotations. In practice, we apply the segmentation network once to all images and cache results. Prior work trained a segmentation network on synthetic data and

applied it during training to ensure semantic consistency. We avoid this as networks trained on synthetic data can generalize poorly to real data. Thus, segmentations from such a network can differ substantially when applied to original and enhanced images. We also avoid backpropagating through the segmentation network as this may result in images that are easy to segment but not necessarily realistic.

The discriminator networks (Fig. 4) each consist of a stack of five Convolution-GroupNorm-LeakyReLU (CGL) layers, which produces a 256-dimensional feature tensor y, and a Convolution-LeakyReLU-Convolution (CLC) layer, which projects the feature tensor down to a single-channel map z. The feature tensor y is further fused with an embedding tensor e via an inner product. The embedding tensor contains a 256-dimensional embedding per pixel, learned from the label maps discussed above. The inner product of features and embeddings was used in prior work. Further details on the architecture of the discriminator networks are provided in the supplementary material.

Applying the VGG feature extraction network to real and enhanced images provides perceptual features at several levels of abstraction. We extract feature tensors from the Relu layers of the VGG and train a discriminator network for each level. This way each network specializes on a different perceptual level.

IV. Experimental Work

Comparison to prior work

For the comparison to prior work, we select a number of baselines that represent multiple lines of work that can be applied to photorealism enhancement. For methods that require semantic segmentation labels as input, we provide maps for synthetic and real images predicted by MSeg, the same robust segmentation network we employ in the discriminator of our method.

Color transfer: We compare against classic work on color transfer. Namely, we evaluate the seminal work of Reinhard et al. [9] (Color Transfer) and the color distribution transfer

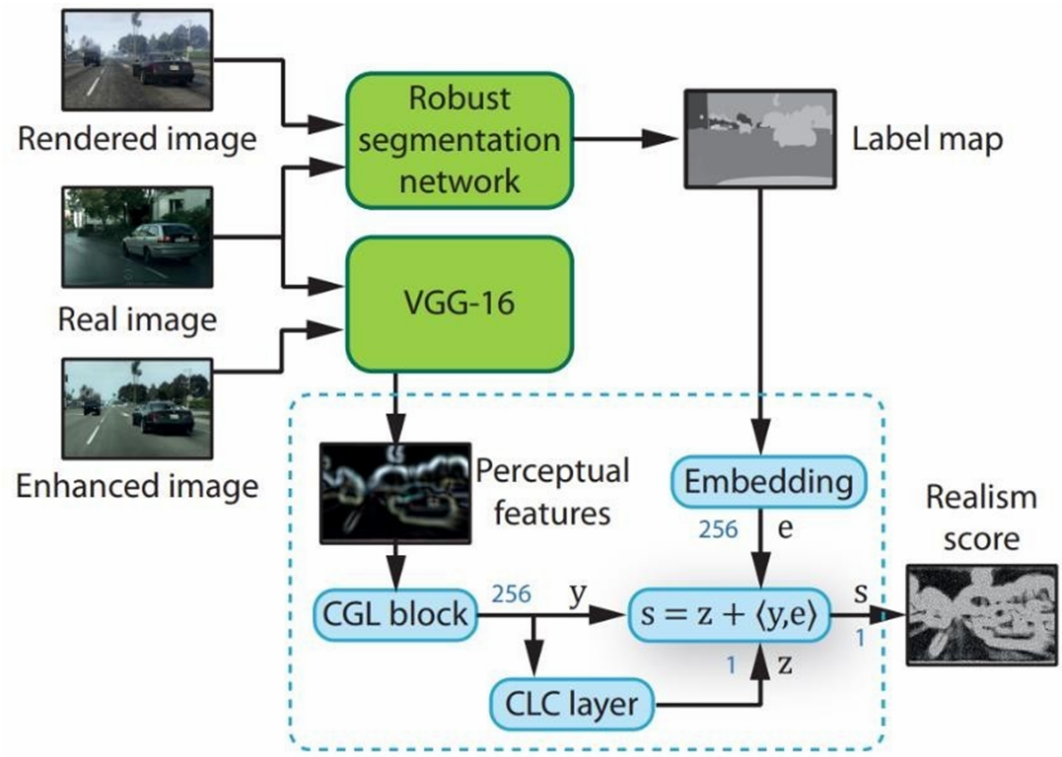

Fig. 1. 1: Methodology

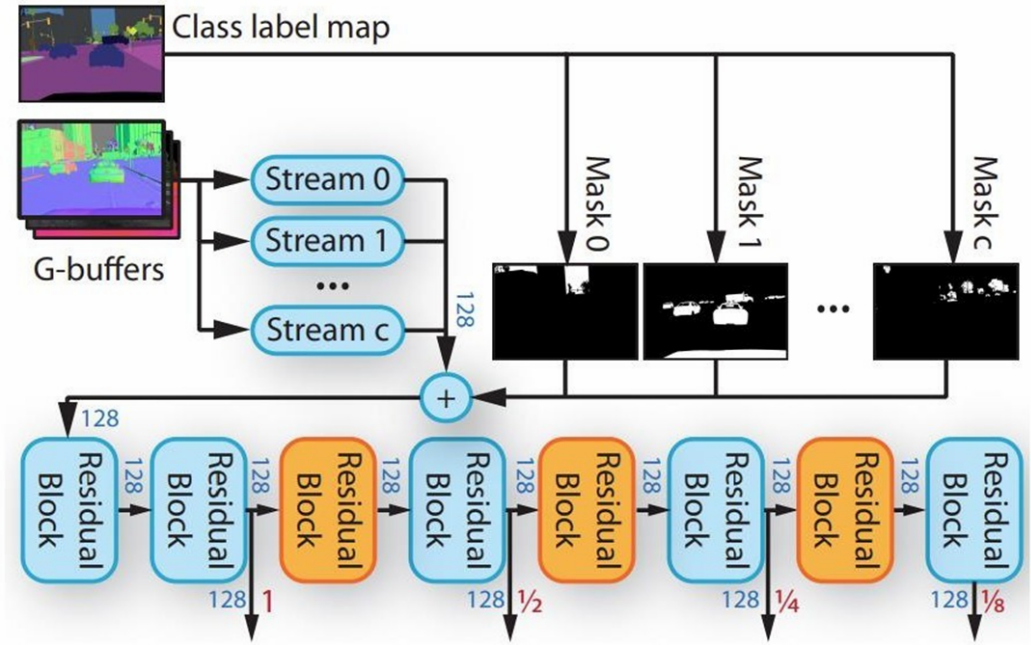

Fig. 2. 2: Extraction from G-Buffers

(CDT). Modifications by these methods are restricted to colors of individual pixels. While this prevents enhancements of textures, it also prevents the introduction of artifacts common to more aggressive learning-based approaches, thus keeping the resulting images fairly close to the original input. Con- sequently, the largest improvements can be observed in the metrics for low-level features (sKVD1 2 and sKVD2 2), where the gains match the level of more recent deep learning approaches.

Metrics

A number of metrics for evaluating the realism of generated images have been proposed. The most common are the Incep- tion Score (IS), the Frechet Inception Distance (FID), and the Kernel Inception Distance (KID). Among these, the KID has been shown to be superior, and we use it in our evaluation for this reason. However, our analysis on the structural shift across datasets implies that quality assessment using the KID may be misleading due to mismatched scene layouts. The KID compares features extracted from the pool3 layer of an inception network, which corresponds to high-level semantic concepts. Thus, roughly speaking, the KID measures distance between semantic structure, but not necessarily a difference in perceived realism. This is problematic, since in enhancing the photorealism of synthetic images we aim to retain the scene structure and semantic content of the source image, rather than shift them towards scene structures that may be more common in the real-world dataset. Put another way, we can trivially minimize the KID by reproducing a real image from the target dataset and ignoring rendered images altogether. Thus, preserving semantic content poses a lower bound on the KID, a level below which the KID should not be driven. Overall, the KID objective is misaligned with the broader photorealism enhancement objective and is a

problematic metric for this reason. We propose a different set of metrics that alleviate this problem. Our metrics build on the KID, but incorporate some key changes. In order to better assess the difference between images at several perceptual levels, we replace the features from the inception network with features extracted at different layers of VGG, since this architecture has been widely used for assessing perceptual image quality. To address the problem of mismatched layouts in the source and target datasets, we align the distribution of patches for which we extract features. Specifically, we extract quadratic patches of 1/8 of the image size from the semantic label maps of source and target datasets. We down sample these patches to a resolution of 16 16 to obtain a 256-dimensional vector. The vectors obtained in this way correspond to ground-truth semantic descriptions of the patches. For each such vector from the synthetic dataset, we find the nearest neighbor in the set of vectors from the real dataset. We retain pairs of vectors with more than 50% matching entries. This way, we obtain a set of semantically corresponding patches from the two datasets. Following the construction of KID, we define our metric as the squared maximum mean discrepancy (MMD) between features from a VGG-16, computed on corresponding patches.1 The different feature representations of the VGG give rise to a family of metrics, characterized by the layer at which feature maps are extracted. We extract features at relu1-2, relu2-2, relu3- 3, relu4-3, relu5-3, and term the corresponding metrics sKVD for semantically aligned Kernel VGG Distance with a subscript indicating the corresponding VGG relu layer.

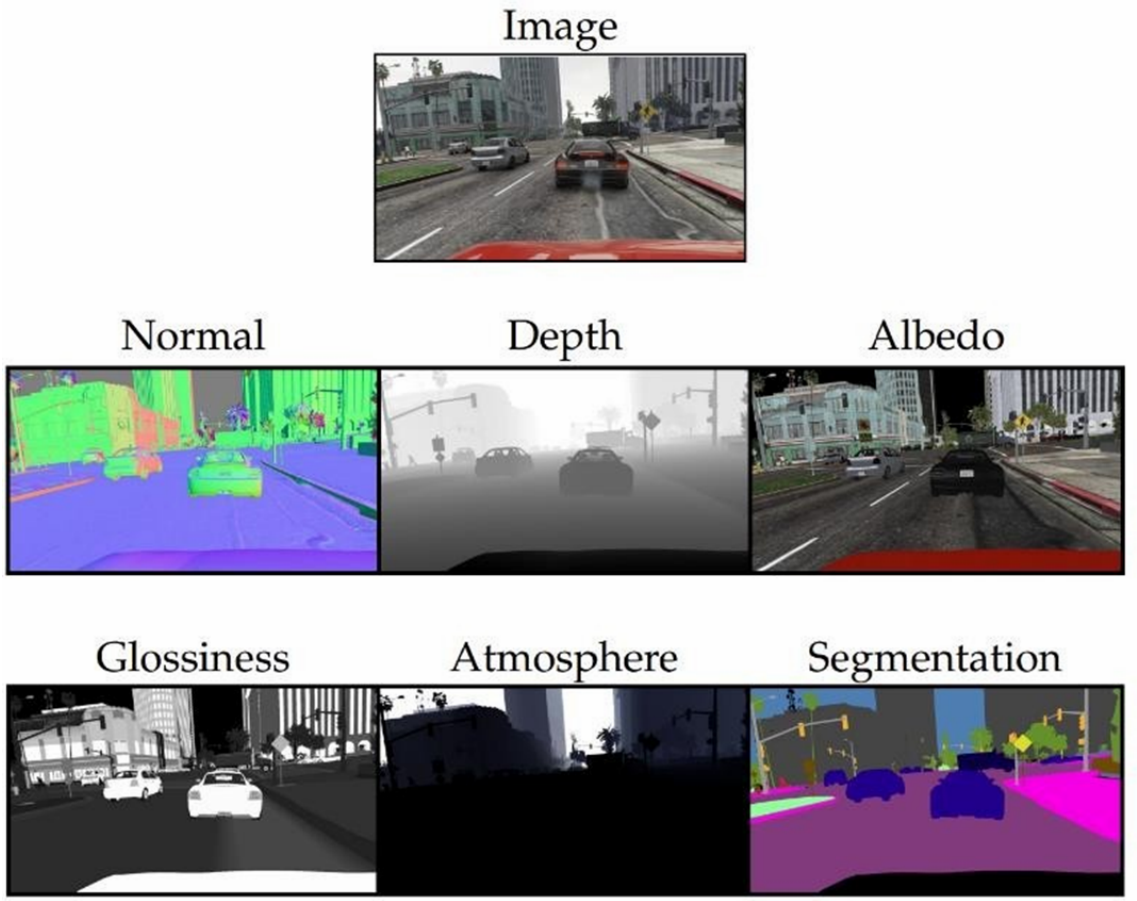

Fig. 3. 3: Features extracted from G-Buffers

Controlled Experiment

To assess the effect of specific ideas in our approach, we conduct a set of controlled experiments. We evaluate our sampling strategy, the importance of G-buffers, architectures for ingesting G-buffers, and different setups for the adversarial loss. The results are shown in Tab. 2. The first column identifies the experimental question and the second row lists the baseline condition. The first row provides reference metrics for unmodified GTA images. The last row provides the metrics for our full approach. How to sample? To assess how the patch sampling strategy affects photorealism enhancement, we compare uniform sampling with different patch sizes (196, 256, 400) to the sampling of matching patch pairs (Ours). The results are consistent with our hypothesis that sampling at smaller patch sizes reduces the mismatch between source and target datasets. We observe stronger hallucination artifacts for larger patch sizes (Fig. 16, columns 2 & 3). Sampling at smaller patch sizes reduces the sKVD considerably. Sampling matching patches further reduces sKVD at medium to high levels of abstraction, while slightly increasing sKVD at the lowest level. This may be explained by the benefits of diversity when uniformly sampling patches, offset by the distribution mismatch for higher levels.

Do G-buffers help? The set of available G-buffers depends on the rendering method used by the game and the method for capturing G-buffers. For example, recording at video rates and lack of deep integration with the game engine limited the set of G-buffers recorded for the VIPER dataset. (The set of G- buffers available for VIPER includes normal, reflection, depth, normal · view, and semantic segmentation.) To investigate

Experiment	Method	KID	$sKVD_{1-2}$	$sKVD_{2-2}$	$sKVD_{3-3}$	$sKVD_{4-3}$	$sKVD_{5-3}$
	GTA	41.44 ± 1.5	330.28 ± 6.5	699.08 ±32.5	917.34 ±53.5	56.89 ± 2.5	11.01 ± 0.4
How to sample?	Ours (Uniform, 400)	**5.34** ± 0.6	43.80 ± 4.0	72.95 ±10.6	61.67 ±11.9	9.50 ± 1.0	1.94 ± 0.2
	Ours (Uniform, 256)	10.65 ± 0.9	10.50 ± 1.6	52.60 ± 7.9	70.98 ±13.3	9.46 ± 1.0	2.48 ± 0.3
	Ours (Uniform, 196)	10.09 ± 0.8	2.63 ± 0.6	12.60 ± 1.8	18.08 ± 4.5	4.30 ± 0.5	1.33 ± 0.1
Do G-buffers help?	Ours (No G-buffer)	12.53 ± 1.0	5.84 ± 0.8	23.08 ± 3.1	34.76 ± 4.5	6.34 ± 0.5	1.96 ± 0.2
	Ours (VIPER)	8.96 ± 0.7	**1.59** ± 0.3	15.70 ± 1.2	20.74 ± 2.7	5.82 ± 0.5	1.60 ± 0.2
How to ingest G-buffers?	Ours (Concat)	11.57 ± 0.7	9.80 ± 1.5	47.07 ± 7.0	90.28 ±13.0	9.26 ± 0.9	2.34 ± 0.2
	Ours (SPADE)	19.14 ± 1.4	32.52 ± 3.2	192.21 ±22.3	296.10 ±35.1	23.54 ± 1.8	3.98 ± 0.3
Which discriminator?	Ours (PatchGAN)		45.46 ± 4.0	42.28 ± 6.4	43.98 ± 7.8	7.90 ± 0.6	2.40 ± 0.2
	Ours (No projection)	14.68 ± 1.0	3.27 ± 0.4	**9.89** ± 0.9	19.08 ± 3.4	4.94 ± 0.5	1.58 ± 0.1
	Ours (No adapt. bp)	10.08 ± 0.7	8.09 ± 1.0	20.83 ± 1.9	20.07 ± 2.1	4.08 ± 0.3	**1.07** ± 0.1
	Ours	10.95 ± 0.8	6.13 ± 1.1	11.12 ± 2.4	**15.45** ± 4.0	**3.61** ± 0.5	1.27 ± 0.1

Chart 1.

The effect of the available set of G-buffers on photorealism enhancement, we compare three conditions: not using G- buffers at all (No G-buffer), the limited set of buffers available for VIPER (VIPER), and our full set. We find that without any G-buffers, enhanced images are significantly less realistic at all but the lowest level. Without any auxiliary information provided by the G-buffers, the network seems to focus much more on low-level features. Adding the buffers from VIPER improves realism at all levels, with the strongest effects observed at low and medium levels. With our full set, the sKVD1 2 is higher, which suggests that the network allocates more capacity to enhancing mid- and high-level features, for which sKVD is reduced.

How to ingest G-buffers? We investigate several strate- gies for ingesting the G-buffers into the image enhancement network. The first is to simply append them to the rendered image (Concat). This corresponds to the strategy employed by AlHaija et al. [10]. As this variant does not treat G-buffers in any special way, RAD modules are not required, and we use instance normalization instead. This condition uses a standard HRNet architecture for image enhancement (no RAD modules or RAD blocks). In the second condition, we replace our RAD modules by SPADE modules (SPADE). The third condition is our full approach, which uses our RAD modules.

V. Conclusion

Our approach significantly enhances the realism of rendered images. This is confirmed by a comprehensive evaluation of our method against strong baselines. Intuitively, our method achieves the strongest and most consistent results for objects and scenes that have clear correspondences in the real dataset; our method excels at road textures, cars, and vegetation. Objects and scenes that are less common in the real images (e.g., close-up pedestrians) are modified less convincingly. Overall, our approach produces high-quality enhancements that are geometrically and semantically consistent with the input images while matching the style of the respective dataset. Our method integrates learning-based approaches with con- ventional real-time rendering pipelines. We expect our method to continue to benefit future graphics pipelines and to be compatible with real-time ray tracing. Inference with our approach in its current unoptimized implementation takes half a second on a Geforce RTX 3090 GPU. Since G-buffers that are used as input are produced natively on the GPU, our method could be integrated more deeply into game engines, increasing efficiency and possibly further advancing the level of realism.

Additional results are shown in the supplementary video: h ttps://youtu.be/P1IcaBn3ej0

Images produced by our method are structurally consistent with the input scenes, which can facilitate the use of ground- truth annotations that may be available for synthetic data.

In addition to the Cityscapes dataset, we demonstrate en- hancing images of GTA to mimic the appearance of Mapillary Vistas [12] and KITTI [11].

References

[1] M. E. Newell And J. F. Blinn, "The Progression Of Realism In Computer Generated Images," Acm Annu. Conf, Pp. 444–448, 1977.

[2] M. Pharr, W. Jakob, And G. Humphreys, 2016.

[3] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, And T. Limperis, 1977.

[4] T. Weyrich, J. Lawrence, H. P. A. Lensch, S. Rusinkiewicz, And T. E. Zickler, "Principles Of Appearance Acquisition And Representation," Found. Trends Comput. Graph. Vis, Vol. 4, No. 2, Pp. 75–191, 2009.

[5] A. Knapitsch, J. Park, Q. Zhou, And V. Koltun, "Tanks And Temples: Benchmarking Large-Scale Scene Reconstruction," Acm Trans. Graph, Vol. 36, No. 4, Pp. 1–78, 2017.

[6] T. Akenine-Moller, E. Haines, N. Hoffman, A. Pesce, M. Iwanicki, And S. Hillaire, 2018.

[7] T. Karras, T. Aila, S. Laine, And J. Lehtinen, "Progressive Growing Of Gans For Improved Quality, Stability, And Variation," Proc. Int. Conf. Learn. Representations (Iclr), 2018.

[8] A. Brock, J. Donahue, And K. Simonyan, "Large Scale Gan Training For High Fidelity Natural Image Synthesis," Proc. Int. Conf. Learn. Representations (Iclr), 2019.

[9] E. Reinhard, M. Ashikhmin, B. Gooch, And P. Shirley, "Color Transfer Between Images," Ieee Comput. Graph. Appl, Vol. 21, No. 5, Pp. 34–41, 2001.
[10] H. A. Alhaija, S. K. Mustikovela, A. Geiger, And C. Rother, "Geometric Image Synthesis," Proc. Asian Conf. Comput. Vis. (Accv), Pp. 85–100, 2018.
[11] A. Geiger, P. Lenz, And R. Urtasun, "Are We Ready For Autonomous Driving? The Kitti Vision Benchmark Suite," Proc. Ieee Conf. Comput. Vis. Pattern Recognit. (Cvpr), Pp. 3354–3361, 2012.
[12] G. Neuhold, T. Ollmann, S. R. Bulo, And P. Kontschieder.