Enhancing Photorealism Enhancement

Stephan R. Richter, Hassan Abu Alhaija, and Vladlen Koltun

Abstract—We present an approach to enhancing the realism of synthetic 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. We analyze scene layout distributions in commonly used datasets and find that they differ in important ways. We hypothesize that this is one of the causes of strong artifacts that can be observed in the results of many prior methods. To address this we propose a new strategy for sampling image patches during training. We also introduce multiple architectural improvements in the deep network modules used for photorealism enhancement. We confirm the benefits of our contributions in controlled experiments and report substantial gains in stability and realism in comparison to recent image-to-image translation methods and a variety of other baselines.

Index Terms—Photorealism enhancement, photorealism, image-to-image translation, style transfer

1 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 physically based simulation of 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, substantially 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 [7], [8], [9], to convert semantic label maps to photographic images [10], [11], [12], [13], [14], [15], [16], and to attempt to bridge the appearance gap between synthetic and real images [17], [18], [19], [20], [21], [22], [23], [24], [25]. Images synthesized by these approaches capture aspects of photographic appearance that often elude even state-of-the-art computer games. On the flip side, these approaches 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], [26]. These buffers provide detailed 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 also seek to eliminate artifacts that can be seen in the results of prior deep-learning approaches, which often hallucinate objects. To this end, we analyze the datasets that are commonly used for photorealism enhancement. Our analysis reveals that their scene layouts differ in ways that can explain artifacts commonly seen in prior work. To better align the datasets and alleviate the artifacts, we propose a new strategy for sampling image patches during training. We further design a new adversarial training objective that facilitates enhancements that are geometrically and semantically consistent with the content of the input image.

Combining all of our contributions, our approach significantly enhances the photorealism of rendered images (Fig. 1). It can add gloss to cars (1st row), green parched hills (2nd row), and rebuild roads (3rd row). Training it with different real-world image collections (e.g., Cityscapes [28], KITTI [29], or Mapillary Vistas [30]) expresses the corresponding visual styles in the output (Fig. 2).

Our analysis further suggests that standard metrics confound differences in style and content. Motivated by this observation, we develop a new family of metrics that mitigate

- *Stephan R. Richter and Vladlen Koltun are with Intelligent Systems Lab, Intel Labs, Santa Clara, CA 95054 USA.*
- *Hassan Abu Alhaija is with PCH Innovations, 10435 Berlin, Germany.*

Manuscript received 5 June 2021; revised 14 Mar. 2022; accepted 20 Mar. 2022. Date of publication 12 Apr. 2022; date of current version 6 Jan. 2023.
(Corresponding author: Stephan R. Richter.)
Recommended for acceptance by R. Timofte.
Digital Object Identifier no. 10.1109/TPAMI.2022.3166687

Fig. 1. We train convolutional networks to enhance the photorealism of rendered images, using intermediate buffers produced by a conventional rendering engine. Left: frames from a modern computer game (GTA V) [27]. Right: same frames enhanced by our approach to mimic the style of Cityscapes [28]. Enhancements by our method are semantically consistent and non-trivial. For example, our method adds gloss to cars (1st row), reforests parched hills to mimic German climate (2nd row), and paves roads with smoother asphalt (3rd row). Insets magnify marked regions.

the effect of mismatched scene layouts and provide a finer-grained assessment of realism at multiple levels.

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. The results indicate that our approach consistently produces the most realistic results, by a wide margin. In all experiments, our approach outperforms all baselines and sets a new state of the art in photorealism enhancement.

2 Related Work

Photorealistic images can be synthesized by simulating all the physical processes involved in image formation. However, this simulation is computationally expensive, may not be feasible at interactive rates, and requires physically accurate models of scenes, objects, materials, and lighting. As a result, a variety of approximations have been developed that generate images that may not be physically correct, but nevertheless appear realistic to varying extents [6], [31].

Methods for conditional image synthesis aim to learn the complete image formation process from data [10], [11], [12], [14], [32], [33], [34], [35]. These works often focus on synthesizing images from semantic label maps. As such, the synthesis is severely underconstrained. Since geometric structure is only provided through the silhouettes of objects and their composition in the label map, substantial ambiguity remains, leading to visible artifacts and temporal inconsistency. Furthermore, the reliance on semantic label maps requires annotated real-world data, which is extremely laborious to create at large scale [28]. Instead of trying to synthesize images, our approach enhances already rendered images, integrates scene information to produce geometrically and semantically consistent images, and does not require any annotation of real data.

Image-based rendering techniques can produce results that are indistinguishable from photographs by recycling real imagery of a scene for rendering novel views [36], [37], [38], [39], [40], [41], [42], [43], [44]. 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.

Another promising direction combines traditional rendering with data-driven approaches. Johnson *et al.* proposed to improve the realism of rendered images by transferring nearest neighbor patches from similarly structured photographs [45].

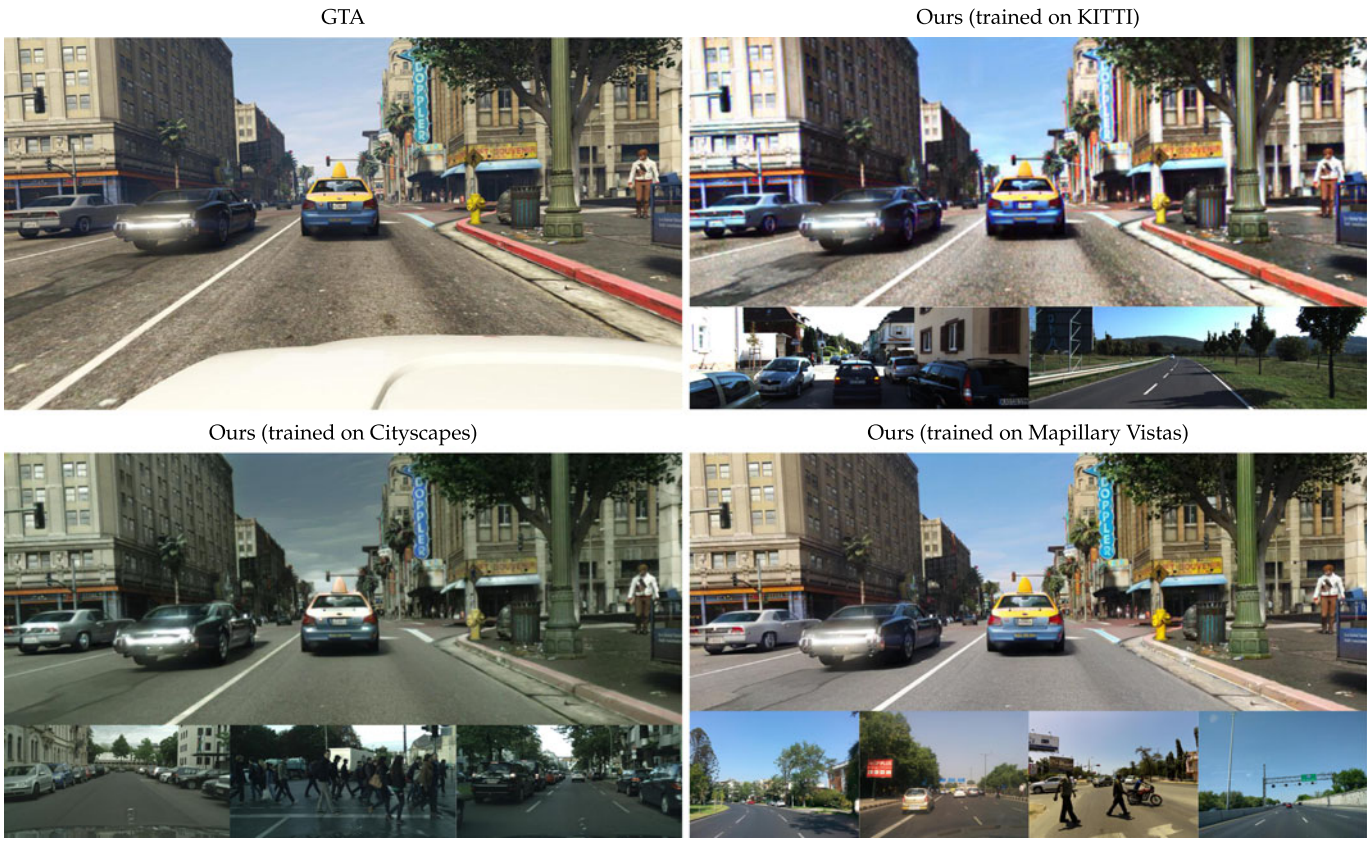

Fig. 2. Targeting different real-world datasets. We train our method to enhance images from GTA (top left) with KITTI, Cityscapes, and Vistas as target datasets. Our method is able to reproduce the characteristic appearance of these datasets (e.g., sensor noise in KITTI, saturation in Cityscapes, fine textures in Vistas) while keeping the structure of the original GTA images. Insets show sample images from the respective target datasets.

Liao *et al*. retrieved nearest neighbor patches in a learned feature space [46]. Reinhard *et al*. matched color distributions of a source image to a target image [47] and showed that this can enhance the realism of computer graphics images. Subsequent work introduced increasingly sophisticated techniques for photographic style transfer between images [18], [48], [49], [50], [51], [52], [53], [54]. While early CNN-based methods required costly optimization per image [55], improvements in feature normalization reduced computation [56], enabled transfer to new styles [57], and increased semantic consistency [14], [34], [35]. Recent work has focused on encoder/decoder structures, mixing content and style of source and target images [53], [58]. Common to all style-transfer and example-based approaches [24] is the dependence on a favorable reference image. Differences in content or layout between rendered and reference images hurt the quality of enhanced images. This is especially problematic for enhancing dynamic content. Our approach learns the style from images through adversarial training and does not require reference images for enhancement.

Our approach is aligned with work on unpaired image-to-image translation, which learns a mapping from one image collection to another [19], [20], [21], [59], [60], [61]. By training on large datasets, these methods do not depend on favorable reference images, and often learn to capture the unique styles of datasets very well. However, as there are no direct correspondences across image collections (e.g., through paired images), the methods need to learn suitable correspondences implicitly. This is challenging and often produces images that appear realistic at first glance, but contain artifacts that are inconsistent with the input images. Improving the consistency between input and output has received a lot of attention and led to additional constraints on this ill-defined problem. Notable improvements came from cycle-consistency [59], [62], custom attention modules [63], temporal regularization [64], modeling sensor noise [65], geometric constraints in the image plane [66], constraints derived from depth maps [67], and contrastive losses [60].

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 [10], [59]. 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 versus fake decision can be accompanied by a classification objective [68], [69]. Additional semantic segmentation maps can be concatenated to the input image [12], [70], projected to a high-dimensional feature space [8], [15], [71], processed via a separate network stream [72], or guide an auxiliary classifier [73]. These works use ground-truth annotations for guiding the discriminator.

While this provides ground-truth semantic information, it restricts the image collections available for training to densely annotated datasets compatible with the synthetic label maps at hand. In our work, we leverage a robust semantic segmentation network [74] to provide label maps that are approximately consistent for synthetic and real images. Previously, Hoffman *et al.* used a segmentation network for synthetic-to-real translation [19]. In contrast to prior work [19], [23], we neither train the segmentation network on any of our datasets nor apply it to enhanced images. This is for two reasons. First, training a segmentation network on synthetic data is prone to overfitting. Thus a network trained in this way may not provide semantically consistent guidance for the transition from synthetic to real images during the training of the generator. Second, incorporating a segmentation network during training can result in generated images that are easy for the segmentation network to parse but are not necessarily more realistic.

Providing the discriminator with additional semantic information supports adaptive processing for different semantic categories. Incorporating the information into the learning objective is an even stronger type of supervision, as the distinction between different objects is explicitly learned. However, datasets that are commonly used in synthetic-to-real translation are limited in their diversity, especially in the context of urban driving [27], [28], [75]. Thus high-capacity networks may overfit to spurious artifacts in the data, leading to suboptimal performance of the generator [76], [77]. Prior research addressed this overfitting via regularization [78], [79] or augmentation [76]. We take inspiration from the neural patch discriminator of Li and Wand [80] and train our discriminator on feature maps from a pretrained VGG network. In contrast to the single-image style transfer approach of Li and Wand, we use multiple feature maps extracted from different layers of the VGG network, and train a discriminator on each.

Prior work hypothesized that differences in scene layout may be a key contributing factor in suboptimal performance in image translation [19] or in training semantic segmentation models [81], [82], and addressed this challenge via regularization [78], semantic consistency losses [19], [83], and matching source and target images [81]. Akin to the work of Li *et al.* on semantic segmentation [81], we address the layout mismatch via semantics-aware sampling. However, Li *et al.* train a dedicated network for matching samples across datasets at image level, which significantly reduces the samples available for training, and hence diversity. We devise a simpler sampling strategy that requires no training and operates at the patch level, leaving sufficiently many diverse samples for training.

Our work is inspired by hybrid approaches that integrate the conventional rendering pipeline in a more structured way, such that information generated during rendering can be subsequently exploited by a learning-based approach. Nalbach *et al.* demonstrated that a CNN can learn to shade G-buffers rendered by a conventional rendering pipeline [84]. AlHaija *et al.* combined this approach with an adversarial loss to improve the realism of rendered images [85]. We introduce a different network structure that better integrates the G-buffers, as well as a new discriminator architecture and training objective, leading to significantly better results.

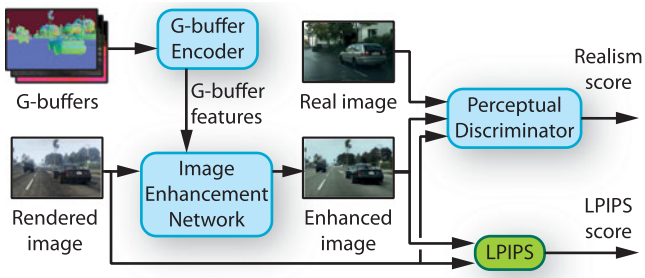

Fig. 3. Our approach is built around an image enhancement network that transforms a rendered image. In addition to the image, the network ingests G-buffer feature tensors at multiple scales. The tensors represent rendering information from a conventional graphics pipeline, encoded by a G-buffer encoder network. We train both networks jointly via an LPIPS loss (to retain the structure of the rendered image) and a perceptual discriminator (to maximize the realism of the enhanced image).

Bi *et al.* developed a pipeline that enhanced the realism of low-quality renderings of indoor scenes [86]. Their approach relies on the availability of paired low-quality and high-quality renderings of the same scenes, involves a number of synthesis stages with different losses, and produces results of limited fidelity. Our work is different in several ways. We assume higher-quality input, as produced by modern computer games. We do not rely on the availability of rendered images at different levels of quality, but do utilize auxiliary buffers produced by rendering engines. Our approach also differs in the network architecture, training objectives, and the application domain. We demonstrate significantly better results, with temporal consistency and measurably higher realism than high-production-value commercial games.

3 Method

3.1 Overview

Fig. 3 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. This is described in Section 3.2. The 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 [88] (see Fig. 4). The HRNet processes an image via multiple branches at different resolutions. Importantly, one feature stream is kept at relatively high resolution ($\frac{1}{4}$ of the input resolution) to preserve fine image structure. We modify the HRNet architecture as follows. First, we replace the initial strided 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, described in Section 3.2. The modified blocks modulate the feature streams based on information extracted from the G-buffers.

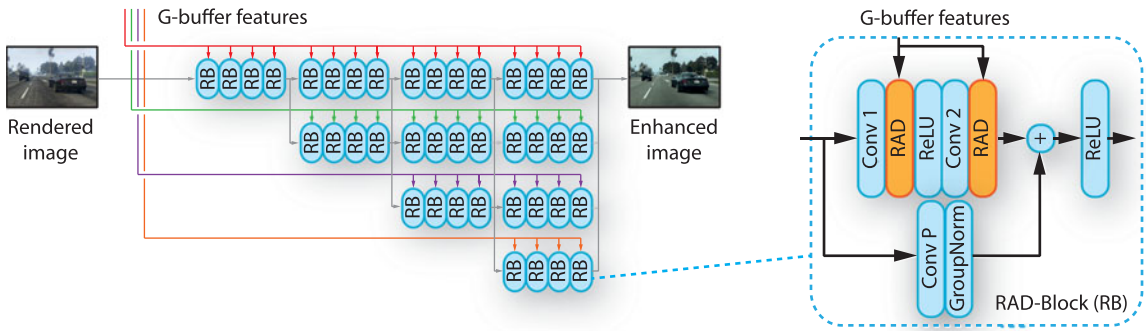

Fig. 4. Image enhancement network. We replace the batch normalization layers within HRNet by rendering-aware denormalization (RAD), forming RAD blocks (RB). Each branch of the HRNet receives a G-buffer feature tensor at a matching scale (different scales are coded by color). Original feature streams are shown in gray. We omit initial stem convolutions as well as transition and fusion layers for clarity.

Fig. 5. For a rendered image (left), G-buffers represent information on geometry (e.g., normal, depth), materials (e.g., albedo, glossiness), and lighting (e.g., atmosphere). A semantic segmentation, which can be derived from the G-buffers [27], [87], provides further high-level information on the scene.

The image enhancement network is trained with two objectives. First, an LPIPS loss [89] penalizes large structural differences between the input and output images. Second, a perceptual discriminator evaluates the realism of output images. The discriminator, described in Section 3.3, is trained to distinguish images enhanced by our network and real photographs from a target dataset. During training, a specific sampling strategy for selecting real and synthetic image patches significantly reduces artifacts that are commonly observed in prior work. This is described in Section 3.4.

3.2 Leveraging Conventional Rendering Pipelines

Many real-time rendering methods factor the rendering process into multiple passes. A particularly popular approach is *deferred shading* or *deferred lighting*, which decouples visibility and shading computations by caching intermediate rendering results in image-sized G-buffers [6], [26], [90], [91]. Although G-buffers generally contain no explicit semantic information, they are consistent with semantic entities [27], [92]. Since they capture geometry and material properties, using them as auxiliary input allows a network to condition the synthetic-to-real translation on the geometry, materials, and illumination of a scene.

Extraction of G-buffers. We base our work on the popular game Grand Theft Auto V. To obtain G-buffers from the game, we follow recent approaches to extracting rendering resources from computer games [27], [87], [93]. Specifically, we extract G-buffers that provide information about geometric structure (surface normals, depth), materials (shader IDs, albedo, specular intensity, glossiness, transparency), and lighting (approximate irradiance and emission, sky, bloom), illustrated in Fig. 5. 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.

While we extract an extensive set of G-buffers for providing comprehensive information about the scene to the generator network for best results, our experiments in Section 4.4 confirm that substantial enhancements can be achieved even if only a subset of the buffers is available.

G-buffer encoder. The G-buffers we extract mix one-hot encodings for material information, dense continuous values for normals, 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 transparent. 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. 6). 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.

Each stream comprises two residual blocks, shown in Fig. 7.

Let f_c denote a feature tensor from a stream targeting object class c, and let m_c denote a mask for objects of that class. We then fuse the tensors via $\sum_c m_c \cdot f_c$.

The fused feature tensors are further processed via residual blocks. We extract a feature tensor before each downsampling residual block to obtain tensors at multiple scales. The feature tensors are ingested by the image enhancement network via RAD modules.

Rendering-Aware Denormalization (RAD). Our rendering-aware denormalization modules take inspiration from recent work on modulating feature tensors based on external information [9], [14], [34], [35]. In contrast to prior work, which conditions on semantic classes [14], style features [34], [35], optical flow [16], or noise [9], our modules learn weights from a comprehensive scene representation. Specifically, our modules transform a feature tensor received from the G-buffer encoder network via 2 residual blocks (Fig. 7). The transformed features condition elementwise scale and shift weights γ and β. The weights represent the

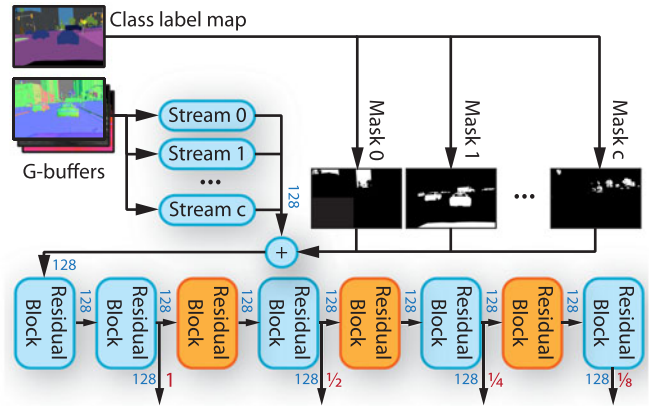

Fig. 6. G-buffer encoder. The G-buffer encoder accounts for different data types and varying spatial density of the G-buffers. It processes them via multiple streams (0–c), which are fused into a joint representation in accordance with one-hot-encoded object IDs. The features are further transformed via residual blocks (see Fig. 7), of which the orange blocks downscale the tensors. The output scale of the feature tensors is red, the channel width is blue. Scales match with branches in the image enhancement network.

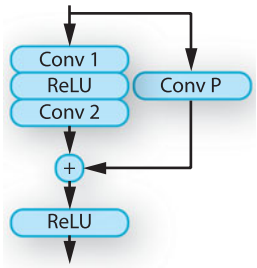

Fig. 7. Residual blocks. In the G-buffer encoder and RAD modules we employ residual blocks. They consist of convolutional layers (kernel size 3) with spectral normalization [94] and ReLUs. Changes in channel width or downscaling are performed in `Conv 1` and `Conv P`. If channel width and resolution are constant, the projection via `Conv P` is omitted.

parameters of an affine transformation of normalized image features (Fig. 8).

Controlled experiments reported in Section 4.4 confirm that RAD modules deliver better results than SPADE [14] when applied to ingesting G-buffer features into the image enhancement network.

3.3 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 discriminator networks (Fig. 9). We employ MSeg [74] for semantic segmentation and VGG-16 [95] 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 [19]. We avoid this as networks trained on synthetic

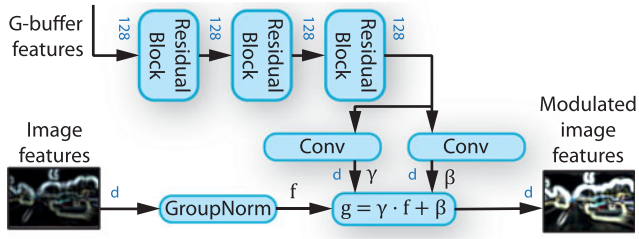

Fig. 8. Rendering-aware denormalization (RAD) modulates image feature tensors via encoded geometry, material, lighting, and semantic information from a conventional rendering pipeline. The image features are normalized via group normalization, then scaled and shifted via per-element weights γ, β. The weights are learned and adapt to G-buffer features received from the G-buffer encoder (Fig. 6). To better adapt the weights, we transform the G-buffer features via three residual blocks within each RAD module.

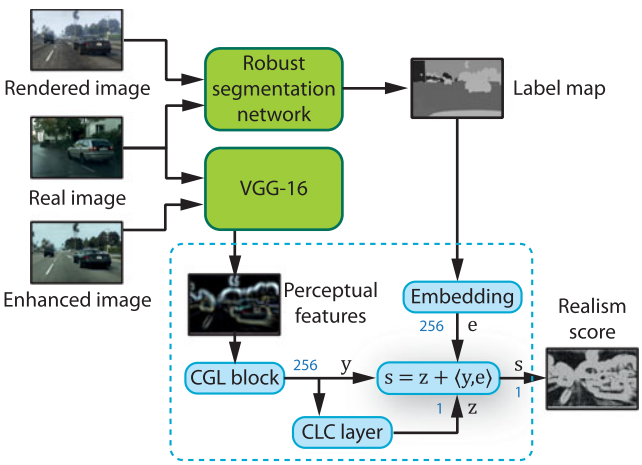

Fig. 9. Perceptual discriminator. A perceptual discriminator evaluates the realism of enhanced images. It consists of pretrained robust segmentation (MSeg [74]) and perceptual (VGG [95]) networks (green). These provide high-level semantic information via label maps and perceptual feature tensors. The maps and tensors are ingested by discriminator networks, which produce a realism score map. For clarity, we only show a single discriminator network, indicated by the dashed rectangle.

data tend to generalize poorly to real data. Applying such a network to original and enhanced images can yield substantially different segmentations. We also avoid backpropagating through the segmentation network as this may result in images that are easy to segment but not necessarily realistic.

Applying the VGG 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.

The discriminator networks (Fig. 9) 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 [8], [15], [71].

Further details on the architecture of the discriminator networks are provided in the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2022.3166687.

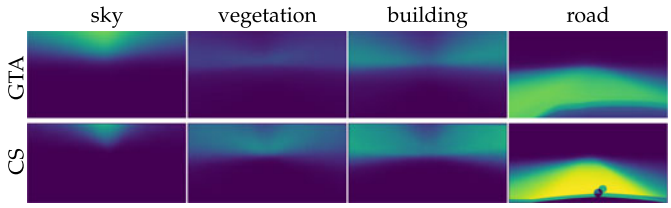

Fig. 10. Scene layouts are different in GTA (top) and Cityscapes (bottom). This affects the probability of seeing certain classes at specific positions. For example, GTA is more likely to have sky at the top of the image, while Cityscapes is more likely to have trees there and a star logo on the hood of the car. Fig. 11 illustrates the impact of this structural mismatch on photorealism enhancement.

3.4 Layout Differences Cause Artifacts

In an adversarial setup as we employ it, a discriminator is trained to classify images, assigning a *real* or *fake* label to each image or pixel. During training, the discriminator will pick up any feature that allows it to easily discriminate real and fake images. For example, if sensor noise is present in real images, but not in synthetic ones, the discriminator will quickly learn to correctly label a noisy patch as *real*. Back-propagating the gradient from the discriminator to the generator encourages the generator to add noise to synthetic images, making them appear more realistic.

A problem arises when fake and real images can be distinguished by relying on spurious features. For example, the probability of seeing sky at the top of an image from GTA V is much higher than for Cityscapes, as can be seen in the probability density maps in Fig. 10. Conversely, at the same position it is much more likely to find trees in Cityscapes than in GTA. Thus, a classifier trained on uniformly sampled images from both datasets may easily identify (real) images from Cityscapes by checking the top of the image. If the top contains some texture resembling trees, it is more likely to be real. Putting this discriminator to work in an adversarial training setup will push the generator to place trees in the sky.

Confirming evidence for this hypothesis comes from recent methods that employ GANs to translate GTA V to Cityscapes – they indeed hallucinate trees at the top or star logos at the bottom of images, as shown in Fig. 11.

Sampling matching patches. Our analysis suggests that randomly sampled images from GTA and Cityscapes differ in their layout in expectation, although they may contain the same type and quantity of objects [87]. This suggests that the standard strategy of comparing image patches that are as large as possible to maximize context [19], [20], [60], [78] is suboptimal for enhancing realism. We propose a different sampling strategy.

Fig. 12. Matching patches across datasets. We show patches sampled from GTA (top) and corresponding patches from Cityscapes (bottom).

First, we shrink the crop size to an area of only about 7% of the full image. This contrasts with prior work which ingests 30–50% of an image as a single patch. Experiments in Section 4.4 confirm that this simple modification already improves results significantly.

Second, we match sampled patches across datasets to balance the distribution of objects presented to the discriminator. Specifically, we process patches from synthetic images with a VGG network [95], pretrained on ImageNet [96], and extract feature tensors at the last `relu` layer. We crop patches at a width of 196 pixels, which corresponds to the receptive field of VGG at this layer. We thus obtain a $1 \times 1 \times 512$ dimensional feature vector per patch. Let $\phi(p_i)$ denote the feature vector computed from patch p_i. We consider two patches as matching if they have a cosine similarity above 0.5:

$$\mathcal{P}_{match}(p_i) = \left\{ p_j \in \mathcal{P}_{real} \middle| \frac{\phi(p_i) \cdot \phi(p_j)}{||\phi(p_i)||||\phi(p_j)||} > 0.5 \right\}, \quad (1)$$

where $\mathcal{P}_{real}$ is the set of patches extracted from real images.

For efficiency, we unit-normalize $\phi(p_i)$ and compute the L_2 distance via FAISS [97]. Per image of a dataset, we sample 75 patches. For each patch from the synthetic dataset, we keep the 10 nearest patches of the target dataset, resulting in 13 milion pairs. During training, we sample from these pairs such that all pixels from the synthetic dataset appear at the same frequency. Examples of matching patches across GTA and Cityscapes are shown in Fig. 12.

3.5 Implementation & Training Details

We train the generator and discriminator with an L_2 loss. The generator is further regularized with an LPIPS loss ($\lambda = 5$) [89]. We train all networks with Adam [98] ($\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay 0.0001). The learning rate is set to 0.0001 and halved every 100K iterations. We clip gradient magnitudes greater than 1000 for all networks. All discriminators are regularized with a gradient penalty on real data

Fig. 11. When transforming images from GTA (left) to match the style of Cityscapes, GANs commonly hallucinate objects such as trees in the sky or star logos at the bottom (remaining columns, highlighted).

($\gamma = 0.06$) [79]. We train with batch size 1 for 1M iterations (controlled experiments 600K).

Adversarial training can become unstable if discriminators become too strong and cease to provide meaningful gradients to the generator [76], [99]. In addition, we have encountered training difficulties that stem from the use of multiple discriminators. For example, we have observed that discriminators that operate on low-level VGG features tend to learn faster, and are thus prone to dominating the training and overfitting. To stabilize training and prevent overfitting of individual discriminator networks, we throttle the training speed of each discriminator by randomly skipping the backward pass for this discriminator with a certain probability. The probability of skipping the backward pass is determined by the current performance of the discriminator. Intuitively, the better a discriminator becomes at judging realism, the closer it is to overfitting, and the less frequently it will be updated by our training protocol. This strategy keeps all discriminators roughly at the same level of performance as training progresses. A similar mechanism was used in ABC-GAN [100] to balance generator and discriminator training. We use this idea to balance multiple discriminators. Our computation of the probability of skipping the backward pass for a given discriminator is inspired by the augmentation probability of Karras *et al.* [76]. (Different from Karras *et al.*, we take into account both real and fake samples in computing the update heuristic.)

4 Evaluation

4.1 Metrics

A number of metrics for evaluating the realism of generated images have been proposed. The most common are the Inception Score (IS) [101], the Fréchet Inception Distance (FID) [102], and the Kernel Inception Distance (KID) [103]. Among these, the KID has been shown to be superior [103], and we use it in our evaluation for this reason.

However, our analysis in Section 3.4 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 [104], 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 [11], [89]. 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 $\frac{1}{8}$ of the image size from the semantic label maps of source and target datasets. We downsample 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 (Fig. 13).

Fig. 13. Semantically corresponding patches sampled for calculating our sKVD metrics.

More formally, let σ denote the vector encoding of a patch and $[\cdot]$ the Iverson bracket. We sample the nearest neighbor patch for p_i by

$$NN(p_i) = \arg\max_{p_j} \sum_k \big[\sigma_k(p_i) = \sigma_k(p_j)\big], \tag{2}$$

where k iterates over elements in the vector encoding.

Following the construction of KID [103], 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.

4.2 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 [74], the same robust segmentation network we employ in the discriminator of our method (see the supplement, available online for examples). We report overall results in Table 1 and show examples in Fig. 14 and in the supplement, available online.

Color transfer. We compare against classic work on color transfer. Namely, we evaluate the seminal work of Reinhard *et al.* [47] (Color Transfer) and the color distribution transfer

1. To compute the MMD, we use the same polynomial kernel as Binkowski *et al.* [103].

TABLE 1
Comparison to Prior Work. All Methods Were Trained on the Cityscapes Dataset. Performance Reported as Kernel Inception Distance $\times 1000$ (KID) and Semantically Aligned Kernel VGG Distance $\times 1000$ (sKVD). Lower is Better

	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
Color transfer	Color transfer [47]	41.31 ± 1.6	36.64 ± 3.0	301.02 ±16.7	210.28 ±13.2	21.74 ± 0.8	4.86 ± 0.2
	CDT [49]	40.92 ± 1.6	56.33 ± 3.9	459.29 ±24.7	329.91 ±20.0	27.16 ± 1.1	5.06 ± 0.2
Photo style transfer	PhotoWCT [53]	56.67 ± 1.9	**4.51** ± 0.5	46.65 ± 3.3	94.36 ± 9.2	13.83 ± 1.1	3.66 ± 0.3
	WCT2 [54]	29.67 ± 1.4	33.33 ± 3.6	160.07 ±15.2	172.70 ±19.2	16.15 ± 1.0	3.30 ± 0.2
Conditional image synthesis	SPADE [14]	77.39 ± 1.9	11.56 ± 0.8	153.10 ± 4.3	244.28 ±12.4	44.97 ± 1.9	11.78 ± 0.4
Image-to-image translation	TSIT [35]	27.97 ± 1.5	38.80 ± 5.2	291.10 ±37.1	330.12 ±40.6	24.88 ± 1.7	4.66 ± 0.3
	CUT [60]	21.18 ± 1.4	19.44 ± 2.4	149.25 ±14.6	214.55 ±24.2	21.31 ± 1.8	5.08 ± 0.3
	MUNIT [20]	15.72 ± 0.9	42.41 ± 3.3	172.75 ±15.4	207.27 ±21.4	18.40 ± 1.1	2.75 ± 0.2
	CyCADA [19]	12.30 ± 0.9	28.39 ± 1.8	50.70 ± 4.4	57.74 ± 5.2	6.38 ± 0.4	2.22 ± 0.2
	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

Gray values indicate standard deviation.

(CDT) of Pitié *et al.* [49]. 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 (Fig. 14). Consequently, the largest improvements can be observed in the metrics for low-level features ($sKVD_{1-2}$ and $sKVD_{2-2}$), where the gains match the level of more recent deep learning approaches.

Photo style transfer. We compare against a closed-form solution for fast photographic style transfer (PhotoWCT) [53] and the state of the art, based on wavelet transforms (WCT2) [54]. Both photo style transfer approaches require a style image and semantic segmentation maps for both source and style image. While color transfer methods apply transformations to individual pixel colors, photo style transfer methods perform transformations in learned higher-level feature spaces guided by semantic segmentations, and thus modify images more strongly. However, photo style transfer methods rely on a favorable style image that matches that input synthetic image. When the input image changes, as happens during interactive exploration of a synthetic environment, photo style transfer can produce unrealistic color shifts or temporal instability.

Conditional image synthesis. We compare against a representative approach to conditional image synthesis, SPADE [14], as it dominates preceding approaches (e.g., [10], [11], [12]). We use a model pretrained for synthesizing urban street scenes, provided by the authors. (The model is pretrained on segmentation ground-truth from Cityscapes, which is compatible with GTA.) SPADE considerably underperforms other methods. This can be explained by two factors. First, synthesizing a photo from only a semantic segmentation map is more challenging than modifying a given image. Second, since SPADE is trained to synthesize images from the Cityscapes dataset (and does so quite well), the distribution shift in scene layouts between Cityscapes and GTA takes this model far outside its training distribution.

Image-to-image translation. We compare against CyCADA [19], which was specifically designed for adapting synthetic images to real photos. To this end, it augments the commonly used pixel-level cycle-consistency with a feature-level cycle-consistency and a semantic consistency loss. The latter leverages corresponding semantic labels to preserve the content of the synthetic images. As CyCADA was originally trained on GTA V and Cityscapes already, we use images provided by the authors. We further compare against MUNIT [20], CUT [60], and TSIT [35]. MUNIT extends the CycleGAN [59] architecture to multi-modal translation and adds a domain-invariant perceptual loss. CUT and TSIT abandon the cycle consistency constraint. CUT builds instead on contrastive learning, and TSIT uses an exemplar style image for transferring style features.

We find that among the various types of baselines, image-to-image translation methods perform best, and within this group CyCADA demonstrates the best results. Although CyCADA makes use of more explicit semantic information than other methods that use a perceptual loss, it still hallucinates extraneous objects (see Fig. 11). A possible reason may be its segmentation network, which is pretrained on unmodified synthetic images and is not updated during training of the image synthesis network. Thus CyCADA implicitly assumes that the drift between synthetic and real images will be limited and the segmentation network will continue to work reliably. Another possible reason is training on large image patches (see Section 3.4 for the associated discussion).

4.3 Perceptual Experiment

We additionally compare our approach to all baselines in a large-scale crowdsourced perceptual experiment. The setup of the perceptual experiment follows prior work and is based on pairwise comparisons with no time limit [11]. We sample 500 images from GTA and compare enhancements from our method to baselines. For each image, our method is paired with each baseline. The pairs are presented to crowd workers. Each pairing is shown 10 times in total (across workers), resulting in 50,000 comparisons in all. The assignment of pairs to workers, their order of presentation, and the left-right order within pairs are all randomized. To filter unreliable workers we include sentinel tasks. The results are shown in Fig. 15. We find that images enhanced by our method are consistently considered more realistic than all baseline methods. The results are statistically significant for all methods (error bars indicate 95% confidence intervals).

Fig. 14. We compare our results to original GTA images and a number of baselines. Our results are structurally consistent with the input. ColorTransfer does not modify textures. WCT2 strongly depends on a favorable reference image. SPADE fails due to differences in the scene layouts. MUNIT hallucinates trees, and CyCADA and CUT hallucinate star logos. Additional results for all baselines are shown in the supplement, available online.

4.4 Controlled Experiments

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 Table 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 (Table 2, How to sample?). This may be explained by the benefits of diversity when uniformly sampling patches, offset by the distribution mismatch for higher levels.

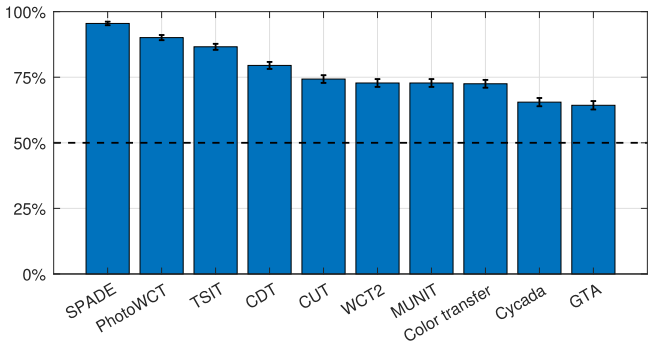

Fig. 15. Perceptual experiment. Randomized blind unlimited-time pairwise comparisons of corresponding images. Values indicate percentage of comparisons in which our method was considered more realistic than the alternative. Chance is at 50%, indicated by the dashed line. Error bars indicate 95% confidence intervals.

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 [87]. (The set of G-buffers available for VIPER includes normal, reflection, depth, normal · view, and semantic segmentation.) To investigate 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 [87] (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 sKVD_{1-2} 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 strategies 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*. [85]. 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 [14] (SPADE). The third condition is our full approach, which uses our RAD modules (last row of Table 2).

TABLE 2
Controlled Experiments. Each Specific Idea in Our Approach Outperforms the Respective Baselines. In Each Condition, We Train for 600K Iterations on GTA and Cityscapes. Lower is Better

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

Gray values indicate standard deviation.

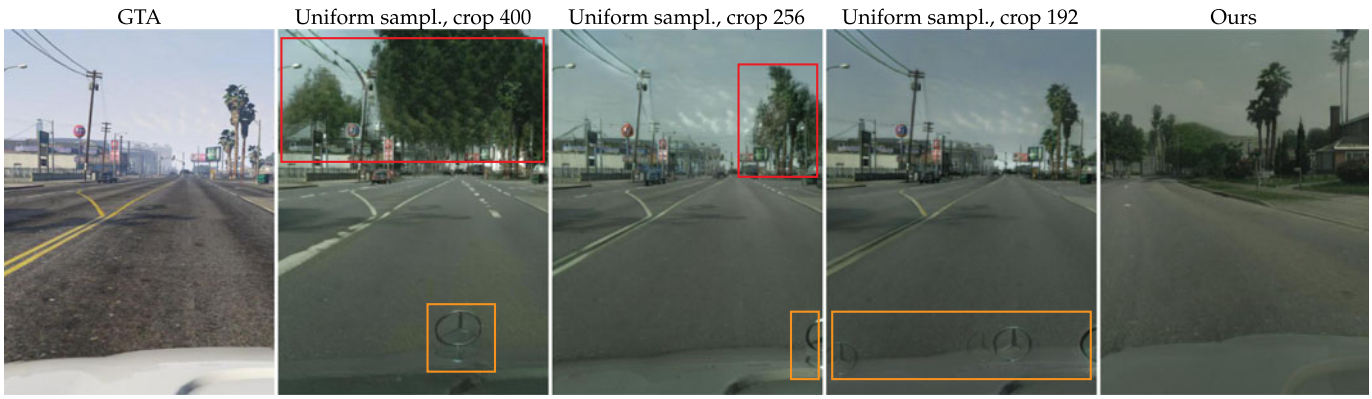

Fig. 16. How to sample? Uniform sampling (columns 2–4) and larger patch sizes (columns 2 and 3) lead to greater mismatch between source and target datasets, inducing artifacts such as hallucinated trees (red) and stars (orange). These can be avoided with our sampling strategy (right).

Fig. 17. How to ingest G-buffers? Enhancement of GTA images (left) using SPADE modules (middle) is highly volatile across the dataset (compare the top and bottom rows). Our RAD modules yield more consistent results (right).

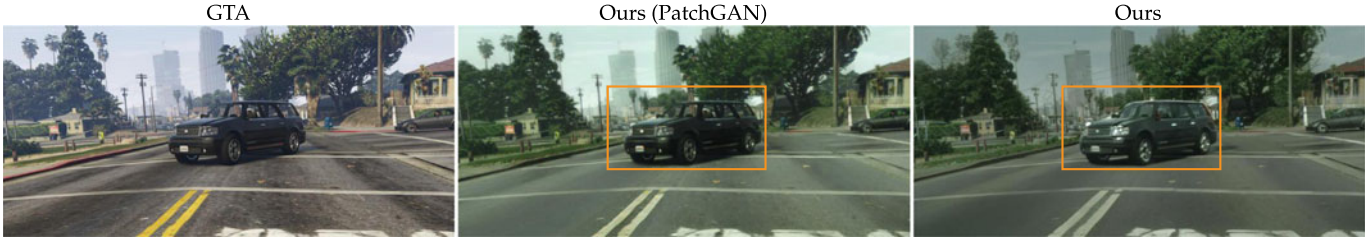

Fig. 18. Which discriminator? Our perceptual discriminator (right) is more effective than a PatchGAN discriminator (middle). Note the stronger enhancements such as lights and reflections on the car on the right.

Fig. 19. Which discriminator?, continued. Adding the projection of semantic embeddings to the discriminator removes artifacts.

The results indicate that simple concatenation yields better results than the SPADE modules. The results with SPADE modules are volatile across the dataset, ranging from realistic images to complete failures with strong artifacts and color shifts (Fig. 17, middle column). In contrast, our results with RAD modules are of consistently high quality (Fig. 17, right column). This is confirmed by all metrics in Table 2.

Which discriminator? We now assess the importance of specific ideas in the perceptual discriminator. As a baseline we pick the PatchGAN discriminator since it is standard in image-to-image translation [10], [12], [14], [20], [32], [34], [35]. Specifically, the PatchGAN condition uses four discriminator networks that each directly ingest images at a different scale, without applying a VGG-16. The network stems share the same architecture as our discriminator networks, except for the projection of segmentations, which is not part of the PatchGAN architecture. That is, they only consist of a CGL-block and a CLC-layer. In the next condition, we remove the robust segmentation network from our approach (No projection). In the third condition, we train the discriminator without adaptively throttling the different discriminator networks as described in Section 3.5 (No adaptive backprop). The fourth condition is our full approach (last row of Table 2).

As reported in Table 2, the results with the PatchGAN discriminator are significantly less realistic. This is illustrated in Fig. 18. The projection layer and adaptive backpropagation both help, but at different perceptual levels. Removing adaptive backpropagation hurts sKVD at all but the highest level. Removing the projection layer increases sKVD at the higher levels. Its effect is illustrated in Fig. 19. The combination of projection and adaptive backpropagation is beneficial when all levels are taken into account.

Fig. 20. More results of enhancing GTA images with Mapillary Vistas as the target dataset. Our method rebuilds roads and makes car paint more glossy (rows 1 & 2). It further increases saturation to match the vibrant colors of Vistas (row 3), reduces haze (row 4), and adjusts the intensity of Fresnel reflections (row 5). Insets magnify marked regions.

4.5 Targeting Other Datasets

In addition to the Cityscapes dataset, we demonstrate enhancing images of GTA to mimic the appearance of Mapillary Vistas [30] and KITTI [29]. Fig. 2 showcases the different appearances. We provide more examples from Vistas in Fig. 20 and in the supplemental material, available online. In Table 3, we compare original GTA images and images enhanced by our approach via the KID. Here we refrain from

TABLE 3
Targeting Different Datasets

Method	Cityscapes	Vistas	KITTI
GTA	41.44 ± 1.5	28.48 ± 1.6	62.05 ± 2.6
Ours	$\mathbf{10.95 \pm 0.8}$	$\mathbf{20.22 \pm 1.5}$	$\mathbf{35.65 \pm 1.9}$

We enhance images from GTA with mapillary vistas and KITTI as targets. As labels for both datasets are not directly compatible with GTA, we evaluate only using KID.

computing the sKVD as the existing ground truth semantic annotations of Vistas and KITTI differ from GTA and manual annotation of the datasets would be a non-trivial amount of work. We observe a considerable reduction of the KID for our enhancements on all datasets. This indicates that our method significantly enhances the realism of GTA images regardless of the dataset being used as a target.

5 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 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 conventional 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 runs at 2fps on a Geforce RTX 3090 GPU. Since G-buffers 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: https://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 [27], [87]. To support future research, we will release enhanced images for the GTA V and VIPER datasets.

References

[1] M. E. Newell and J. F. Blinn, "The progression of realism in computer generated images," in *Proc. ACM Annu. Conf.*, 1977, pp. 444–448.

[2] M. Pharr, W. Jakob, and G. Humphreys, *Physically Based Rendering: From Theory to Implementation*, 3rd ed. San Mateo, CA, USA: Morgan Kaufmann, 2016.

[3] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis, "Geometrical considerations and nomenclature for reflectance," U.S. Dep. Commerce, National Bureau of Standards, Tech. Rep. 003-003-01793, 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. 78:1–78:13, 2017.

[6] T. Akenine-Möller, E. Haines, N. Hoffman, A. Pesce, M. Iwanicki, and S. Hillaire, *Real-Time Rendering*, 4th ed. Boca Raton, FL, USA: Peters/CRC Press, 2018.

[7] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–12.

[8] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–11.

[9] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4401–4410.

[10] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5967–5976.

[11] Q. Chen and V. Koltun, "Photographic image synthesis with cascaded refinement networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1520–1529.

[12] T. Wang, M. Liu, J. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional GANs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8798–8807.

[13] X. Qi, Q. Chen, J. Jia, and V. Koltun, "Semi-parametric image synthesis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8808–8816.

[14] T. Park, M. Liu, T. Wang, and J. Zhu, "Semantic image synthesis with spatially-adaptive normalization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2337–2346.

[15] X. Liu, G. Yin, J. Shao, X. Wang, and H. Li, "Learning to predict layout-to-image conditional convolutions for semantic image synthesis," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 568–578.

[16] A. Mallya, T. Wang, K. Sapra, and M. Liu, "World-consistent video-to-video synthesis," in *Proc. 16th Eur. Conf. Comput. Vis.*, 2020, pp. 359–378.

[17] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2242–2251.

[18] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M. Yang, "Universal style transfer via feature transforms," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 386–396.

[19] J. Hoffman *et al.*, "CyCADA: Cycle-consistent adversarial domain adaptation," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1994–2003.

[20] X. Huang, M. Liu, S. J. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *Proc. 15th Eur. Conf. Comput. Vis.*, 2018, pp. 179–196.

[21] H. Lee, H. Tseng, J. Huang, M. Singh, and M. Yang, "Diverse image-to-image translation via disentangled representations," in *Proc. 15th Eur. Conf. Comput. Vis.*, 2018, pp. 36–52.

[22] Z. Wu *et al.*, "DCAN: Dual channel-wise alignment networks for unsupervised scene adaptation," in *Proc. 15th Eur. Conf. Comput. Vis.*, 2018, pp. 535–552.

[23] A. Cherian and A. Sullivan, "Sem-GAN: Semantically-consistent image-to-image translation," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2019, pp. 1797–1806.

[24] L. Ma, X. Jia, S. Georgoulis, T. Tuytelaars, and L. V. Gool, "Exemplar guided unsupervised image-to-image translation with semantic consistency," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–10.

[25] A. Dundar, M. Liu, Z. Yu, T. Wang, J. Zedlewski, and J. Kautz, "Domain stylization: A fast covariance matching framework towards domain adaptation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 7, pp. 2360–2372, Jul. 2021.

[26] T. Saito and T. Takahashi, "Comprehensible rendering of 3-D shapes," in *Proc. SIGGRAPH*, 1990, pp. 197–206.

[27] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *Proc. 14th Eur. Conf. Comput. Vis.*, 2016, pp. 102–118.

[28] M. Cordts *et al.*, "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 3213–3223.

[29] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.

[30] G. Neuhold, T. Ollmann, S. R. Bulò, and P. Kontschieder, "The mapillary vistas dataset for semantic understanding of street scenes," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 5000–5009.

[31] E. Reinhard, A. A. Efros, J. Kautz, and H. Seidel, "On visual realism of synthesized imagery," *Proc. IEEE*, vol. 101, no. 9, pp. 1998–2007, Sep. 2013.

[32] T. Wang *et al.*, "Video-to-video synthesis," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1152–1164.

[33] M. Wang *et al.*, "Example-guided style-consistent image synthesis from semantic labeling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1495–1504.

[34] P. Zhu, R. Abdal, Y. Qin, and P. Wonka, "SEAN: Image synthesis with semantic region-adaptive normalization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 5103–5112.

[35] L. Jiang, C. Zhang, M. Huang, C. Liu, J. Shi, and C. C. Loy, "TSIT: A simple and versatile framework for image-to-image translation," in *Proc. 16th Eur. Conf. Comput. Vis.*, 2020, pp. 206–222.

[36] P. E. Debevec, C. J. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach," in *Proc. SIGGRAPH*, 1996, pp. 11–20.

[37] H. Shum and S. B. Kang, "Review of image-based rendering techniques," in *Proc. Vis. Commun. Image Process.*, 2000, vol. 4067, pp. 2–13.

[38] D. N. Wood *et al.*, "Surface light fields for 3D photography," in *Proc. SIGGRAPH*, 2000, pp. 287–296.

[39] C. Buehler, M. Bosse, L. McMillan, S. J. Gortler, and M. F. Cohen, "Unstructured lumigraph rendering," in *Proc. SIGGRAPH*, 2001, pp. 425–432.

[40] J. Kopf, M. F. Cohen, and R. Szeliski, "First-person hyper-lapse videos," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 1–10, 2014.

[41] P. Hedman, J. Philip, T. Price, J. Frahm, G. Drettakis, and G. J. Brostow, "Deep blending for free-viewpoint image-based rendering," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 1–15, 2018.

[42] M. Broxton *et al.*, "Immersive light field video with a layered mesh representation," *ACM Trans. Graph.*, vol. 39, no. 4, pp. 86:1–86:15, 2020.

[43] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *Proc. 16th Eur. Conf. Comput. Vis.*, 2020, pp. 405–421.

[44] G. Riegler and V. Koltun, "Stable view synthesis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 12211–12220.

[45] M. K. Johnson, K. Dale, S. Avidan, H. Pfister, W. T. Freeman, and W. Matusik, "CG2Real: Improving the realism of computer generated images using a large collection of photographs," *IEEE Trans. Vis. Comput. Graphics*, vol. 17, no. 9, pp. 1273–1285, Sep. 2011.

[46] J. Liao, Y. Yao, L. Yuan, G. Hua, and S. B. Kang, "Visual attribute transfer through deep image analogy," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 120:1–120:15, 2017.

[47] E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley, "Color transfer between images," *IEEE Comput. Graph. Appl.*, vol. 21, no. 5, pp. 34–41, Jul./Aug. 2001.

[48] F. Pitié, A. C. Kokaram, and R. Dahyot, "N-dimensional probability density function transfer and its application to colour transfer," in *Proc. 10th IEEE Int. Conf. Comput. Vis.*, 2005, pp. 1434–1439.

[49] F. Pitié, A. C. Kokaram, and R. Dahyot, "Automated colour grading using colour distribution transfer," *Comput. Vis. Image Underst.*, vol. 107, no. 1/2, pp. 123–137, 2007.

[50] F. Luan, S. Paris, E. Shechtman, and K. Bala, "Deep photo style transfer," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6997–7005.

[51] R. Mechrez, E. Shechtman, and L. Zelnik-Manor, "Photorealistic style transfer with screened poisson equation," in *Proc. Brit. Mach. Vis. Conf.*, 2017, pp. 153.1–153.12.

[52] M. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 700–708.

[53] Y. Li, M. Liu, X. Li, M. Yang, and J. Kautz, "A closed-form solution to photorealistic image stylization," in *Proc. 15th Eur. Conf. Comput. Vis.*, 2018, pp. 468–483.

[54] J. Yoo, Y. Uh, S. Chun, B. Kang, and J. Ha, "Photorealistic style transfer via wavelet transforms," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 9035–9044.

[55] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2414–2423.

[56] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, "Instance Normalization: The Missing Ingredient for Fast Stylization," 2016, *arXiv:1607.08022*.

[57] X. Huang and S. J. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1510–1519.

[58] X. Li, S. Liu, J. Kautz, and M. Yang, "Learning linear transformations for fast image and video style transfer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3809–3817.

[59] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2242–2251.

[60] T. Park, A. A. Efros, R. Zhang, and J. Zhu, "Contrastive learning for unpaired image-to-image translation," in *Proc. 16th Eur. Conf. Comput. Vis.*, 2020, pp. 319–345.

[61] T. Park *et al.*, "Swapping autoencoder for deep image manipulation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 7198–7211.

[62] Z. Yi, H. R. Zhang, P. Tan, and M. Gong, "DualGAN: Unsupervised dual learning for image-to-image translation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2868–2876.

[63] J. Kim, M. Kim, H. Kang, and K. Lee, "U-GAT-IT: Unsupervised generative attentional networks with adaptive layer-instance normalization for image-to-image translation," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–11.

[64] K. Park, S. Woo, D. Kim, D. Cho, and I. S. Kweon, "Preserving semantic and temporal consistency for unpaired video-to-video translation," in *Proc. ACM Int. Conf. Multimedia*, 2019, pp. 1248–1257.

[65] A. Carlson, K. A. Skinner, R. Vasudevan, and M. Johnson-Roberson, "Sensor transfer: Learning optimal sensor effect image augmentation for sim-to-real domain adaptation," *IEEE Robot. Automat. Lett.*, vol. 4, no. 3, Jul. 2019.

[66] H. Fu, M. Gong, C. Wang, K. Batmanghelich, K. Zhang, and D. Tao, "Geometry-consistent generative adversarial networks for one-sided unsupervised domain mapping," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2427–2436.

[67] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 95–104.

[68] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2642–2651.

[69] H. Tseng, H. Lee, L. Jiang, M. Yang, and W. Yang, "RetrieveGAN: Image synthesis via differentiable patch retrieval," in *Proc. 16th Eur. Conf. Comput. Vis.*, 2020, pp. 242–257.

[70] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," 2014, *arXiv:1411.1784*.

[71] T. Miyato and M. Koyama, "cGANs with projection discriminator," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–13.

[72] E. Ntavelis, A. Romero, I. Kastanis, L. V. Gool, and R. Timofte, "SESAME: Semantic editing of scenes by adding, manipulating or erasing objects," in *Proc. 16th Eur. Conf. Comput. Vis.*, 2020, pp. 394–411.

[73] E. Schönfeld, V. Sushko, D. Zhang, J. Gall, B. Schiele, and A. Khoreva, "You only need adversarial supervision for semantic image synthesis," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–13.

[74] J. Lambert, Z. Liu, O. Sener, J. Hays, and V. Koltun, "MSeg: A composite dataset for multi-domain semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2876–2885.

[75] G. Ros, L. Sellart, J. Materzynska, D. Vázquez, and A. M. López, "The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 3234–3243.

[76] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training generative adversarial networks with limited data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 12104–12114.

[77] D. Zhang and A. Khoreva, "PA-GAN: *Improving GAN Training by Progressive Augmentation*," 2019, *arXiv:1901.10422*.

[78] Z. Jia *et al.*, "Lipschitz regularized CycleGAN for improving semantic robustness in unpaired image-to-image translation," 2020, *arXiv:2012.04932*.

[79] L. M. Mescheder, A. Geiger, and S. Nowozin, "Which training methods for GANs do actually converge?," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 3478–3487.

[80] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," in *Proc. 14th Eur. Conf. Comput. Vis.*, 2016, pp. 702–716.

[81] G. Li, G. Kang, W. Liu, Y. Wei, and Y. Yang, "Content-consistent matching for domain adaptive semantic segmentation," in *Proc. 16th Eur. Conf. Comput. Vis.*, 2020, pp. 440–456.
[82] X. Peng, B. Usman, N. Kaushik, J. Hoffman, D. Wang, and K. Saenko, "VisDA: The Visual Domain Adaptation Challenge," 2017, *arXiv:1710.06924*.
[83] P. Li, X. Liang, D. Jia, and E. P. Xing, "Semantic-aware Grad-GAN for virtual-to-real urban scene adaption," in *Proc. Brit. Mach. Vis. Conf.*, 2018, Art. no. 73.
[84] O. Nalbach, E. Arabadzhiyska, D. Mehta, H. Seidel, and T. Ritschel, "Deep shading: Convolutional neural networks for screen space shading," *Comput. Graph. Forum*, vol. 36, no. 4, pp. 65–78, 2017.
[85] H. A. Alhaija, S. K. Mustikovela, A. Geiger, and C. Rother, "Geometric image synthesis," in *Proc. Asian Conf. Comput. Vis.*, 2018, pp. 85–100.
[86] S. Bi, K. Sunkavalli, F. Perazzi, E. Shechtman, V. G. Kim, and R. Ramamoorthi, "Deep CG2Real: Synthetic-to-real translation via image disentanglement," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 2730–2739.
[87] S. R. Richter, Z. Hayder, and V. Koltun, "Playing for benchmarks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2232–2241.
[88] J. Wang *et al.*, "Deep high-resolution representation learning for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 10, pp. 3349–3364, Oct. 2021.
[89] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 586–595.
[90] M. Deering, S. Winner, B. Schediwy, C. Duffy, and N. Hunt, "The triangle processor and normal vector shader: A. VLSI system for high performance graphics," in *Proc. SIGGRAPH*, 1988, pp. 21–30.
[91] N. Thibieroz, "Deferred shading with multiple render targets," in *ShaderX^2: Introductions & Tutorials with DirectX 9*, W. Engel, Ed., Plano, TX, USA: Wordware, 2004.
[92] A. Shafaei, J. J. Little, and M. Schmidt, "Play and learn: Using video games to train computer vision models," in *Proc. Brit. Mach. Vis. Conf.*, 2016, pp. 1–13.
[93] P. Krähenbühl, "Free supervision from video games," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2955–2964.
[94] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–14.
[95] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–9.
[96] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.
[97] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," 2017, *arXiv:1702.08734*.
[98] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–11.
[99] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–14.
[100] I. Susmelj, E. Agustsson, and R. Timofte, "ABC-GAN: Adaptive blur and control for improved training stability of generative adversarial networks," in *Proc. Int. Conf. Mach. Learn. Workshop Implicit Models*, 2017, pp. 1–8.
[101] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2226–2234.
[102] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local nash equilibrium," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6626–6637.
[103] M. Binkowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying MMD GANs," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–15.
[104] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2818–2826.

Stephan R. Richter received the BSc and MSc degrees in IT systems engineering from the Hasso-Plattner-Institute in Potsdam, Germany, and the PhD degree in computer science from TU Darmstadt, Germany. He is currently a research scientist with the Intelligent Systems Lab at Intel in Munich, Germany. His research interests span topics in computer vision and related areas.

Hassan Abu Alhaija received the BSc degree in informatics engineering from Damascus University, Syria in 2011 and the European master's degree in color informatics and media technology (CIMET) in 2014. He is currently a research scientist with PCH Innovations in Berlin, Germany, and working toward the PhD degree with Heidelberg University, Germany. His research interests include neural rendering, 3D generative models and synthetic media.

Vladlen Koltun is currently the chief scientist for Intelligent Systems with Intel. He directs the Intelligent Systems Lab, which conducts high-impact basic research in computer vision, machine learning, robotics, and related areas. He has mentored more than 50 PhD students, postdocs, research scientists, and PhD student interns, many of whom are now successful research leaders.