EST-GAN: Enhancing Style Transfer GANs with Intermediate Game Render Passes

Martina Mittermueller*
University of Vienna
Vienna, Austria
a01306362@unet.univie.ac.at

Zhanxiang Ye*
University of Vienna
Vienna, Austria
zhanxiang.ye@univie.ac.at

Helmut Hlavacs
University of Vienna
Vienna, Austria
helmut.hlavacs@univie.ac.at

***Abstract*—In this paper we introduce EST-GAN, an approach to improve the realism of frames from unsophisticated game scenes. For this purpose, several Generative Adversarial Networks (GANs) structures are applied, which are extended to handle intermediate game render passes generated by conventional rendering pipelines. We present an image-to-image translation method that transforms simple low-poly game scenes into the style of an elaborately produced video game. Through our experiments, we show that the involvement of G-buffer information has a significant impact on the results of these translations and the usage of these leads to a reduction in artifacts.**

***Index Terms*—Games, Generative Adversarial Network, Style Transfer, G-Buffer, Texture, Graphics Pipeline**

I. Introduction

Deep learning is a major sub-area of machine learning that has gained more and more popularity in the last few years. The ability to process a huge number of attributes has opened new possibilities. This created many new experiments in the field of computer graphics and vision that may enable new realistic representations. In comparison, conventional computer graphics methods require physical principles and accuracy so that, e.g. the necessary geometry, camera settings, and surface properties can be qualitatively represented. Nowadays, ray-tracing or rasterization is mainly used to perform this task [1]. The creation of realistic scenes is a desirable goal in the field of computer graphics [2]. Although many approximations and improvements have been achieved nowadays, some of which have been integrated into real-time applications, the computations that need to be performed for these specific tasks are expensive [3]. To counter this problem, other works have already applied deep learning to create synthetic data that looks realistic [1]. However, most of these approaches are completely disconnected from the original rendering pipeline which is utilized by game engines. This often results in many distracting artifacts in the results, causing the quality of the synthetic data to suffer. But a new proposal of combining the information generated during game rendering with machine learning could provide a positive impact on the resulting output. Richter et al. [2] leveraged the Cityscapes dataset [4] to achieve a more realistic rendering of the video game Grand Theft Auto V based on this concept.

*These authors contributed equally.

Since we decided to follow the approach of Richter et al. [2] it is necessary to assemble our own dataset. Our dataset must correlate in its properties with our game scene, so we created a synthetic dataset using the game Red Dead Redemption 2 (RDR2), due to its variety of landscape scenes. According to the publisher [5], the usage of the game's material for non-commercial purposes is permitted. Moreover, we generate a dataset that not only contains final rendered frames of the game, but furthermore contains a subset of G-buffer information consisting of depth, normal, and albedo maps. In addition, we need to create a pixel-precise semantic label map for our task.

Regarding neural networks, we decided to utilize Generative Adversarial Networks (GANs), which find an increased application, especially in the field of image generation. This involves the process of creating synthetic images that should end up being indistinguishable from that of the original dataset [6]. To use GANs for our purposes, we adopt existing methods and adapt the architecture so that multiple inputs, consisting of the final frame and the G-buffer information, can be used as training data.

In order to perform various experiments, we utilized Unity [7] to create a simple nature scene that covers objects in the target domain.

While GANs have been used in several previous works to enhance rendered scenes, these tend to use existing scenes [2] rather than self-created scenes or do not include the rendering pipeline [8], [9] in their methods. This paper presents an approach that integrates self-created video game scenes and information from the graphics pipeline to make their simple rendering appear more realistic.

II. Related Work

A. Image Translation

In the context of machine learning, image translation is the ability of machines to take an input image and learn a mapping function so that it is transferred to another domain. Numerous works showed that this task is applicable in various settings [2], [8], [10]–[12]. These range from translating semantic label maps to photorealistic images to simple sketches mapped to real-looking objects. To learn this translation, a training dataset is needed upon which this specific mapping is based.

B. Image Segmentation

Semantically labeled data has become an indispensable aspect in the context of deep learning algorithms, as neural networks need a lot of data to produce meaningful and acceptable results. Krähenbühl [13] observes that the gathering of labeled data is as important as the construction of the neural network itself. There are numerous researchers who have addressed this problem and created multiple datasets that represent different content and provide the most accurate semantic segmentation possible, such as Cityscapes [4]. The challenge, however, lies in the resources that must be allocated to create such a dataset in the first place, as this is a very costly task to do manually.

C. Synthetic Datasets

An alternative solution to address the problem of costly data collection is to extract ground truth labels directly from video games. Consequently, data is collected for semantic labeling or other applications without having to be manually processed by humans. An advantage is the wide range of variation that video games offer, as well as the high quality graphics of Triple-A titles [13].

Richter et al. [14] explain in their work that modern real-time rendering architectures mostly use the concept of deferred shading. This simplifies the process of extracting individual pieces of information from the rendering pipeline. The acquisition of the desired data is achieved by intercepting the communication between the game and the graphics hardware. This results in the advantage that no source code needs to be provided to extract the video game data, and thus datasets can be created from commercial games. This approach can reduce the effort of conventional methods significantly [14].

D. Pix2PixHD

A particular application of a GAN architecture is provided by Wang et al. [6] who introduced their method Pix2PixHD. The general goal behind this supervised approach is to generate photorealistic output by using semantic label maps, thus it is used for image domain transfer.

Pix2PixHD has the functionality to support high resolutions, which can scale up to 2048×1024 pixels. This technique is based on the application of training pairs, which can be notated as (s_i, x_i). Here, s_i is a semantic label map and x_i is the corresponding image of the target domain. The generator has been adapted to a coarse-to-fine generator, where first a smaller network is trained and a lower resolution result is generated. Once this is finished, the generator is extended by additional layers at the beginning and at the end to achieve high resolutions and also to stabilize them. The discriminator consists of multi-scale discriminators, which evaluate content on different scales, which also allows individual features and details to be reviewed. According to the authors, the usage of those leads to an improved result [6].

E. CycleGAN

In recent work, it has been found that it is advantageous in domain mapping to use GANs with cycle consistency constraints. These seem to give better results when transforming images from one domain to another. This is even possible without the use of image pairs, which is of significant value for many applications since these are not always available or require a lot of time and effort in their creation [9].

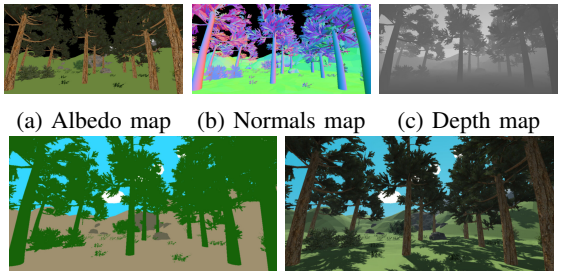
An unordered set of images X should be converted into the appearance of a collection of images Y. By using this method, it is intended that this can work both ways and images of the domain Y can also be translated into those of X [8]. In general, this mapping can be expressed as $G : X \rightarrow Y$, where it is linked to its inverse mapping $F : Y \rightarrow X$. Therefore, CycleGAN trains these two desired generators G and F. Their respective discriminators D_X and D_Y evaluate whether an image generated by G or F is a counterfeit or an original. During the training of a model, two so-called cycle consistency losses are used, by which the generated image is translated back into its original domain in each case. This aims to produce an input x_i to $\hat{Y}$ by a generator G and to convert this generated image again to its original domain by generator F, so that $x_i \approx \hat{x_i}$. This is also equally valid in the other direction [8].

III. Method

In this section we describe the way we generated our source and target datasets for image-to-image translation. Afterwards our objective is to find a suitable GAN G which is capable of translating our input samples X, consisting of custom created Unity landscape scenes, to the desired output Y, reflecting the RDR2 domain. By the end we should obtain a GAN with the mapping $G : X \rightarrow Y$. Furthermore, we define $\{x_i\}_{i=1}^{N}$ where $x_i \in X$ and $\{y_i\}_{i=1}^{M}$ where $y_i \in Y$ is a set of training samples. Since both domains are synthetic representations, it is possible to extract the semantic label maps and G-buffers. Let $\{x_i\}_{i=1}^{N}$ and $\{y_i\}_{i=1}^{M}$ both become sets of quintuples containing their corresponding images $\{(x, x_{depth}, x_{normals}, x_{albedo}, x_{label})_i\}_{i=1}^{N} \in X$ and $\{(y, y_{depth}, y_{normals}, y_{albedo}, y_{label})_i\}_{i=1}^{M} \in Y$ respectively. x_i and y_i both remain the final render of their corresponding domains.

A. Generating the Source Domain Dataset

Our custom created game scene does not visually deviate too much from our target domain RDR2, but simultaneously does not add too much complexity to the scene. For segmentation and extraction of the G-buffers we used ML-ImageSynthesis [15] as a starting point. These scripts replace the fragment shaders with an Uber Replacement Shader at runtime and render the desired output. Furthermore, we added an albedo shader which just renders the color value of a given texture. Additionally, it is essential that the categories (terrain, vegetation, sky, cloud, building, rock and water) of the objects must be accommodated as well to match our color scheme. This is achieved by assigning a specific layer to an Unity



(a) Albedo map (b) Normals map (c) Depth map

(d) Semantic labels (e) Final frame

Fig. 1: Sample of the Unity domain, showing our custom nature scene and the associated G-buffers.

component. Figure 1 shows one of the sampled frames of domain X.

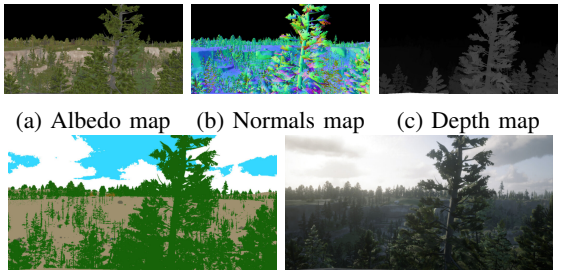
B. Generating the Target Domain Dataset

Unfortunately the modification to RenderDoc [16] by Richter et al. [14] for extracting GTA V frames were not suitable for our purpose. Instead, we settled with a standard up to date build of RenderDoc and utilized its Python API [17] to parse the captured frames. During the segmentation process, we encountered similar problems as Richter et al. which we discuss in more detail in the next subsections, however we solve them differently.

Identifying relevant function calls. There are some rendering patterns that help us to identify the relevant function calls. RDR2 does deferred shading in one go, which means that it does not do anything else during that period, thus we can easily extract the G-buffers. Additionally we can use this behavior to start segmenting. Objects are drawn one by one and each of the draw calls contain certain meshes, textures and shaders. By identifying them we can associate a label to the drawn region. Once the game is finished with the G-buffers, it will copy the depth map to another resource. RenderDoc marks these actions as copy actions and this particular one also includes the other textures of the G-buffer, therefore we were able to retrieve the albedo and normals maps as well. The starting point on the other hand was harder to locate. We could not detect any patterns related to the starting of the deferred shading, thus we naively looked for draw calls which contain fragment shaders that render objects onto the frame. The sky and water regions are always rendered separately after deferred shading. Water always gets drawn by a specific set of fragment shaders. After determining the correct draw call, we can extract a heatmap of pixels which contain water. The same procedure applies to the sky. The only difference is that it consistently uses one particular texture in the draw call, therefore we do not even have to look for particular fragment shaders.

Identifying resources. Resources in RenderDoc can represent anything related to the game engine such as shaders, textures, and meshes [18]. Since resource identifiers are different in every frame in RDR2 as well, we adapted the idea to calculate the hash of the memory content.

Semantic labeling. Since we only have 7 different categories for our segmentation, we manually collected variations of textures and shaders used to render their corresponding object belonging to a label category.



(a) Albedo map (b) Normals map (c) Depth map

(d) Semantic labels (e) Final frame

Fig. 2: Sample of a nature scene belonging to the RDR2 domain and its corresponding G-buffers.

Capturing frames. As mentioned before, we could not use the already existing RenderDoc modification by Richter et al. [14] for our purposes, so automated capturing was not possible. We resorted to manually capturing each frame, resulting in a smaller sample set of $M = 2286$, which is only a fraction of the dataset Richter et. al managed to process. Nonetheless the quality overall is quite acceptable as demonstrated in Figure 2.

C. Pix2PixHD: Paired Image-to-Image Translation

Training image translations works ideally when paired training data is used. Since we do not have a direct mapping from X to Y, we first trained a conditional GAN using Pix2PixHD [6]. This framework is able to produce a solid mapping between semantic label maps and the desired target domain and is of particular interest due to its ability to process multiple inputs, in addition to the semantic label map. This gives us a baseline to concatenate the sampled G-buffers of the Y domain. This method was also used by AlHaija et al. [19], furthermore evaluated by Richter et al. [2] and apparently this simple amendment resulted in satisfactory outputs.

Preparing training data of Y. For our purpose the resolution of the global generator in Pix2PixHD is sufficient, so we proceed to crop the RDR2 images to 1024×512, however, this introduces a new problem. After cropping the depth maps their values are not consistent among the training samples anymore. To overcome this issue we decided to normalize the depth maps between 0 and 255 afterwards.

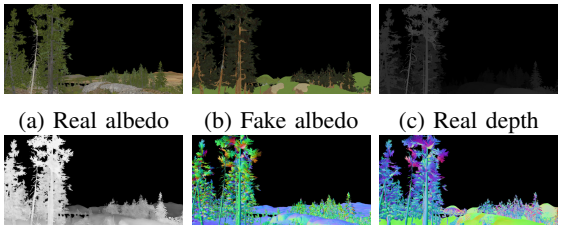
Our trained model $p_{model(y)}$ generates images that resemble the target domain to a large extent. The G-buffers are especially helpful for the network to, i.e. recreate details in the textures with the normals map, retaining the colors with the albedo map and reproduce perceptual focus with the depth map.

Preparing training data of X. The samples collected from the Unity engine already match the resolution of the global generator of Pix2PixHD, therefore no cropping is necessary here. The depth map itself on the other hand is inverted. We

simply invert and normalize all depth maps in our training samples.

Plugging in samples from X delivers unsatisfying results. This can be argued by the fact that the G-buffers of both domains simply do not have any similarities at all, thus a translation of the G-buffers of the Y to the X domain must be found.

D. CycleGAN: Unpaired Image-to-Image Translation



(a) Real albedo (b) Fake albedo (c) Real depth

(d) Fake depth (e) Real normals (f) Fake normals

Fig. 3: Sample of domain translated G-buffers by CycleGAN.

We settled with CycleGAN [8] to find the mapping of the G-Buffers from domain Y to X. This requires some modifications, we want to translate all G-buffers at the same time, hence we set the number of input channels to 7 ($N_{depth} = 1, N_{normals} = 3, N_{albedo} = 3$) to cover the entirety of the G-buffers, in consequence the number of output channels are equally 7. Figure 3 shows a set of G-buffers that were successfully translated. We define the translated images as $\hat{y}_{depth}, \hat{y}_{normals}, \hat{y}_{albedo}$. They seem to match the domain of X, but on further inspection it is quickly noticeable that the network hallucinates vegetation must cover a majority of each frame, even when it is not the case in the original scene.

Semantic loss. In order to improve the translations, we add semantic loss to the overall loss function, similar to the work of Cherian et al. [20]. They proposed to train a segmentation network for the source domain X and the target domain Y. Using this network we can estimate the accuracy of the mappings of CycleGAN $G : X \rightarrow Y$ and $F : Y \rightarrow X$ in terms of semantic consistency. Let the segmentation network of X and Y be S_X and S_Y respectively.

For our purpose, we used a slightly different loss function which is defined as the following:

$$\begin{aligned}\mathcal{L}_{seg}(S_X, S_Y, G, F) = \\ \mathbb{E}_{x\sim p_{data}(x)}[\|S_X(x) - S_Y(G(x))\|_1 + \\ \|S_X(x) - S_X(F(G(x)))\|_1]\lambda_X + \\ \mathbb{E}_{y\sim p_{data}(y)}[\|S_Y(y) - S_X(F(y))\|_1 + \\ \|S_Y(y) - S_Y(G(F(y)))\|_1]\lambda_Y\end{aligned}$$

Similar to the cycle loss our $\mathcal{L}_{seg}$ also considers the segmentation of $G(F(y)) \approx y$ and $F(G(x)) \approx x$. Furthermore, we simply used the L1 loss to compare two segmentations. This works since we used the DeepLabV3 [21] framework with ResNet 101 [22] to train both S_X and S_Y with the

(a) Real albedo map (b) Baseline (c) Semantic loss

Fig. 4: Improved domain transfer by CycleGAN using semantic loss.

G-buffers as input and the output of DeepLab tells us the probability of each pixel belonging to a certain semantic label. Here, we aim for a matching probability distribution over all labels and therefore we used the L1 loss. Finally, we add two weights λ_X and λ_Y to set the importance of each of these two losses. Due to time constraints we do not know if this method produces better results than the findings of Cherian et al., but nonetheless our extended version of CycleGAN improved the results significantly as can be seen in Figure 4.

E. Pix2PixHD with Semantic Loss

Since we already added semantic loss to CycleGAN, we decided to also extend the Pix2PixHD framework and apply a semantic loss to it. This idea already has been pioneered by Liu et al. [23]. In order to do so, we train a model S to segment RDR2 frames and add the same loss function to Pix2PixHD as Liu et al. did. With the trained segmentation network for RDR2 frames, we can also simultaneously train the generator G with Unity inputs. For each iteration we do not only use $(y_{label}, \hat{y}_{depth}, \hat{y}_{normals}, \hat{y}_{albedo}, y)$ as input, but also $(x_{label}, x_{depth}, x_{normals}, x_{albedo})$ with $\hat{y}$ being the translations of CycleGAN. Our semantic loss can be described as:

$$\begin{aligned}\mathcal{L}_{seg} = \lambda * [\mathbb{E}_{y\sim p_{data}(y)}\mathbb{E}_{\hat{y}\sim CycleGAN_{model}(y)} \\ \mathcal{L}(S(G(y_{label}, \hat{y})), y_{label}) + \\ \mathbb{E}_{x\sim p_{data}(x)} \\ \mathcal{L}(S(G(x_{label}, x_{depth}, x_{normals}, x_{albedo})), x_{label})]\end{aligned}$$

$\mathcal{L}(P, Q)$ is the cross entropy loss function and λ is the existing feature weight of Pix2PixHD. $\mathcal{L}_{seg}$ is then simply added to the overall loss of the framework.

F. Increase Similarity of G-buffers in Domain Y

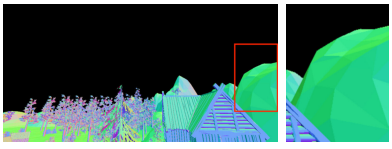


Fig. 5: There are sharp edges visible in the normals map of Unity, due to the low polygon assets.

Although the translations of CycleGAN are providing promising results, there are still obvious differences observable. Due to the low polygon nature of a few assets in our Unity scene, normals maps tend to have a higher amount of sharp edges as demonstrated in Figure 5, this will result in

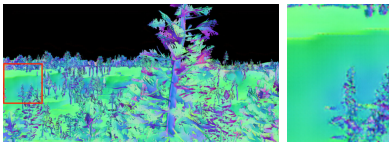


Fig. 6: Translation of RDR2 with CycleGAN still contains smoother normals compared to the Unity domain.

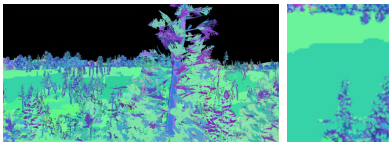


Fig. 7: Clustering of translated normals map of RDR2 which reproduces the desired sharp edges.

unpleasing looking outputs. In order to avoid this issue, we reduce the amount of values in the normals of the $\hat{Y}$ domain. We achieve this by clustering the images by instances and their regions using the K-Means algorithm. We apply the same clustering to albedo maps as well, to get rid of some artifacts caused by CycleGAN. The before and after results of the clustering of the normals can be observed in Figure 6 and 7 respectively.

The result of the translated depth maps is not usable for us, because it seems like some details from the normals and albedo maps were taken into account as well, therefore we decided to resort to the original ones. Due to the bigger scene in RDR2 compared to our custom created one, the depth map values of RDR2 provide a bigger range of distances. By using gamma color correction it is possible to increase the lower depth values and decrease the visual differences between the two domains. Therefore we are moving the background of the depth nearer to the camera. Figure 8 presents a overview of all processing steps required to train our modified instance of Pix2PixHD.

IV. Evaluation

A. Analysis of the Training Dataset

In order to further elaborate on the synthetic data generated by EST-GAN in the later evaluation, it is crucial to inspect the training data that we used. For this reason, we focused on the composition of our training data in more detail and analyzed the distribution of its visual classes, as has been done, for example, with the synthetic GTA V [14] or even the real Cityscapes dataset [4].

In total, we extracted 2286 frames including the respective G-buffer information from RDR2. For training, we utilize 2145 frames and for the subsequent validation, we divided 141 frames as a test set. 504 frames were extracted from our Unity scene to train CycleGAN and Pix2PixHD. Furthermore, additional 145 were allocated for the testing of both GANs. Our RDR2 dataset includes a total of 7 individual labels. We chose these categories due to the fact that they are the most common objects in RDR2 and are furthermore characteristic for landscape scenes.

Figure 9 shows the result of our collected data extracted from RDR2 and used for the training. The categories are arranged according to the number of pixels they contain. It is noticeable that the classes sky, terrain and vegetation are most frequently displayed, while water is by far the smallest fraction contained in the extracted frames. This can be explained by the fact that we occasionally received erroneous results when segmenting water, due to its diverging rendering methods in the game as described in section III-B.

Within this work, we rather included buildings that were found in rural parts of the game and completely avoided small cities, since here again the rendering differs from other areas and in this case, similarly, segmentations sometimes yielded erroneous results. Due to the low pixel count of the water category, we decided to exclude it completely from our Unity dataset. Figure 10 shows an example of a translation of a scene containing water. The water is hardly recognizable and its presence also negatively affects the surrounding objects.

B. Model Variations

We performed a total of 3 different experiments, each of which either had a modified GAN structure or differed with respect to the parameters or the input they were provided with. The goal of these experiments was to achieve a steady improvement of our results, shown in terms of their visual quality:

- Final: Our final method with the translated training samples, clustered normals and albedo maps with K-Means, corrected depth maps with gamma correction and semantic losses added to both GAN frameworks.
- Version 1: Same as Final, but without additional clustering of normals and albedo, depth map translated by CycleGAN instead of manually correcting them and semantic loss using L1 loss instead of the cross entropy loss in Pix2PixHD.
- Version 2: Same as Final, but no translated training samples with CycleGAN, no gamma correction applied to depth maps and no semantic losses added to Pix2PixHD. Clustering of albedo and normals still applies.
- Pix2PixHD: Baseline, only utilizes label and instance maps.

In this section, we will perform a manual visual evaluation of the results to assess their quality. Moreover, the sample images that were selected for evaluation were selected randomly from 145 test results.

Figure 11 shows a total of four versions that were created during the execution of the implementation and during intermediate evaluations. To illustrate the progress of our work, we also show a result of the original Pix2PixHD method that we trained with our own dataset.

The scene shown for this example (Fig. 11) includes terrain, buildings, sky, clouds, rocks as well as vegetation and thus represents all classes except water. All variants can achieve a good result when it comes to sky and clouds. However,

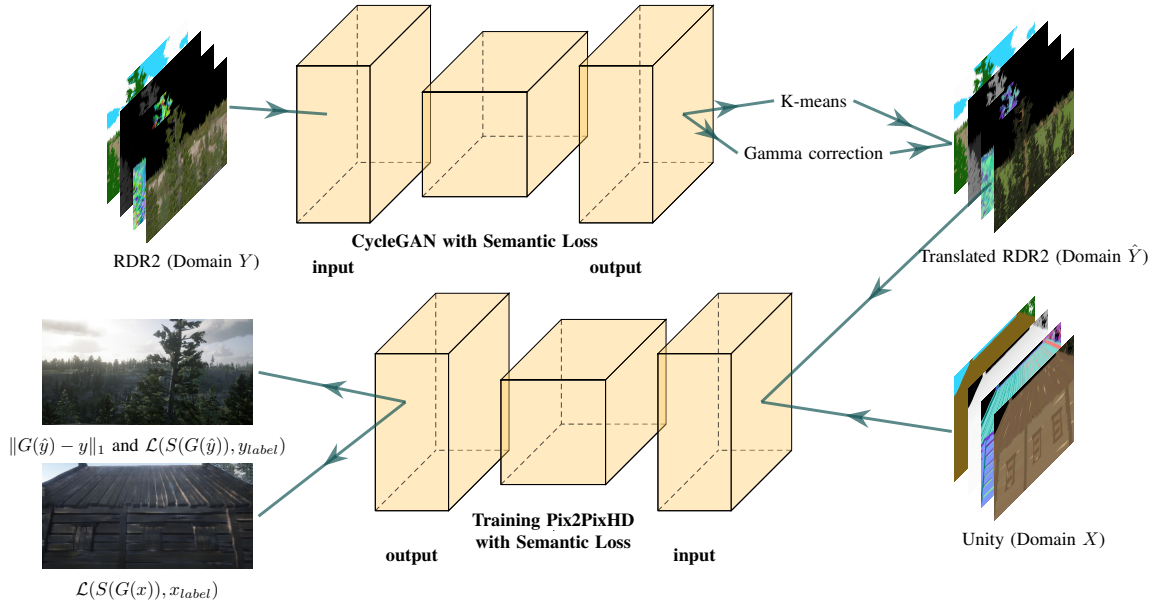


Fig. 8: Overview of Pix2PixHD's training pipeline. Our CycleGAN model translates preprocessed training data of the Y domain to the $\hat{Y}$ domain which ressembles the look of our Unity scene (X domain). We train Pix2PixHD to perform the mapping of $G : X \rightarrow Y$ with both the translated RDR2 inputs and sampled data of Unity as training sets.

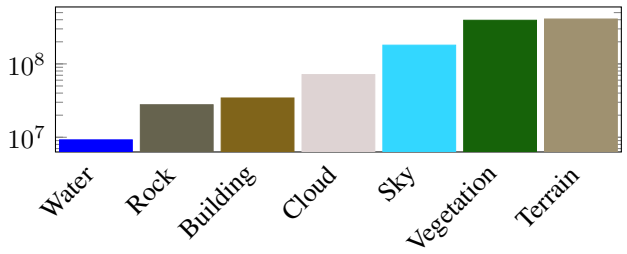
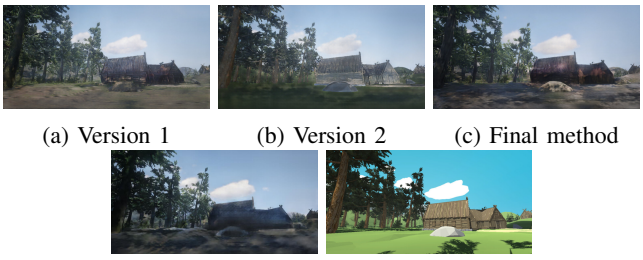


Fig. 9: The total number of annotated pixels per class of the RDR2 training dataset, using the logarithmic scale.

(a) Translated scene (b) Segmentation (c) Unity frame

Fig. 10: The presence of water has a negative impact on the whole scene, since the number of pixels corresponding to the water category is the lowest in our dataset.

clear differences become obvious for the other categories when being observed individually. Adjustments made to our final method allows it to achieve visually pleasing results overall, as well as provide a decent approximation of the RDR2 domain. However, this version contains some deficiencies, as seen in Figure 11c. Although buildings are generated in very similar colors to the RDR2 domain, they lose some details and appear blurry in some areas. Nonetheless, the final version achieves the best results when observing the trees more closely. The



(a) Version 1 (b) Version 2 (c) Final method

(d) Pix2PixHD (e) Unity frame

Fig. 11: All variations of our modifications trained (a,b,c), as well as the standard Pix2PixHD version (d), which have translated a Unity frame (f) into the RDR2 domain.

colors of the leaves, as well as the wood, are clearly separated and do not blend. Likewise, this version shows clear details of the bark on the trees, whereas these are not preserved in such detail in all other versions. Another problem becomes visible with Version 1. In this version not only the leaves of the trees are colored green, but unfortunately also parts of the tree barks are in the same color. In addition, this version contains some artifacts when displaying stones and terrain. However, with Version 2, further deterioration of the terrain becomes visible, resulting in worse blurred results.

Overall, all our extensions can deliver better results than the Pix2PixHD baseline, though. The baseline achieves solid results in the categories sky, cloud and vegetation, but strong artifacts appear in all other categories, as Figure 11d clearly demonstrates. E.g., houses contain strong artifacts that make them almost unrecognizable. Meanwhile, Version 1 (Fig. 11a) as well as our final method (Fig. 11c) achieve the best results

when manually evaluated, which can approximate the target domain the most.

C. Quantitative Evaluation

Comparison	FID
RDR2 vs. Unity	236.539
Unity vs. Final	**214.473**

TABLE I: FID (lower is better) between the test set of RDR2/Unity and between Unity/our final method.

1) Fréchet Inception Distance: The Fréchet Inception Distance (FID) is used to measure the distance between the distribution of two image collections. In this case, the real dataset can be compared with the synthetic images generated by the GAN [24]. The final score gives information about how similar two groups of images are statistically [25].

As can be observed in Table II, the variance of the metrics is notably high. Even the baseline Pix2PixHD framework scores better than our final method. Although it is apparent that our results are visually more appealing due to the availability of the G-buffers. This most likely occurs because of the structural dissimilarity between the RDR2 and Unity domains. This assumption is backed up by the results of Table I. The distance between Unity and the final outputs of our method is on the smaller side, hence the structural features sampled by the Inception model might have a more significant weighting. Another pattern we have noticed is that all the distances achieve a smaller value in their results than the one between RDR2 and Unity visible in Table I, thus the style transfer does seem to be successful disregarding what method was applied.

RDR2 vs. Version	FID
Final	176.649
Version 1	**165.975**
Version 2	172.623
Pix2PixHD	169.271

TABLE II: FID metric (lower is better) of all versions. Version 1 shows better results than our final method.

Regarding Table II, we observe that Version 1 shows stronger results than our final method by a significant margin. We assume that due to the use of the depth map translated by CycleGAN, the buildings in Version 1 are visually more appealing. This suspicion is backed up by the DRN IoU scores of each class listed in Table IV. Objects that are further away from the camera in RDR2 appear blurrier and less detailed. Since our final method was trained with depth maps that were gamma corrected, the generator might learn that only the objects directly in front of the camera should be visually more detailed and sharp, despite the fact that the background should be near enough to still be in focus.

2) DRN: Since we have access to the ground truth labels of each translated scene, we can use a segmentation network to perform semantic interpretability on our results. Therefore, we trained a DRN-D-22 model [26] to perform semantic segmentation against our RDR2 training set. Then we compute the mean average precision (mAP), pixel-wise accuracy (pixAcc) and average class accuracy (classAcc) as used in other works [27]. Our final method exceeds the baseline Pix2PixHD in the mAP metric during training as represented in Figure 12. The only other Version that does keep up and even outperforms this method in the classAcc and the IoU of the building class metric is the first one as shown in the tables III and IV. The clustering of the albedo and normals maps increases the quality of the outcome noticeably. This procedure removes artifacts created by the translation of the CycleGAN generator. Furthermore, the newly formed edges caused by the reduction of vectors in the normals maps assist the generator of Pix2PixHD to smooth them out in the final image when Unity samples are used. Additionally, the usage of the cross entropy loss instead of the L1 loss contributes to the better scores achieved by our final method since it doesn't have to match the probability distribution anymore.

Version	mAP	picAcc	classAcc
Final	**0.605**	**0.907**	0.807
Version 1	0.584	0.889	**0.840**
Version 2	0.541	0.822	0.737
Pix2PixHD	0.481	0.858	0.704

TABLE III: DRN metrics (higher is better) of all versions. Our final method seems to perform fairly well and only gets exceeded by Version 1 in the classAcc metric.

Version	mAP	Sky	Cloud	Terrain	Rock	Building	Vegetation
Final	**0.605**	**0.922**	**0.707**	**0.857**	**0.528**	0.398	**0.823**
Version 1	0.584	0.897	0.621	0.829	0.394	**0.540**	0.811
Version 2	0.541	0.916	0.669	0.680	0.421	0.354	0.749
Pix2PixHD	0.481	0.882	0.514	0.785	0.357	0.041	0.788

TABLE IV: IoU values (higher is better) of each class.

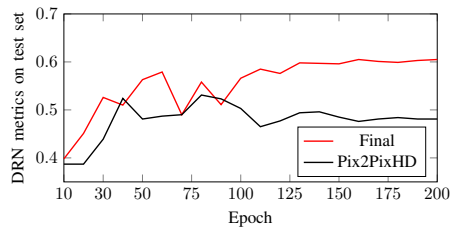


Fig. 12: mAP metrics (higher is better) of Final and Pix2PixHD during training. The semantic loss used in our final method contributes to the higher score of all three metrics.

V. Conclusion

When extracting the data from RDR2, we limited it to a subset of G-buffer data, since Richter et al. [2] confirmed in their work that a high level of improvement can already be achieved even if only a smaller number of G-buffers are available. This statement proved to be true as shown in our results. As discussed in section IV, we investigated the utility of G-buffers of the rendering pipeline in our work. The related results demonstrated that the overall image quality could be improved by using these intermediate buffers. Furthermore, we were able to see improvements by using multiple networks

such as CycleGAN with semantic loss for the Unity G-buffers to reduce the gap between domains.

We introduced a method, based on existing GANs to translate frames of custom games without the need for expensive post-processing techniques that require a lot of computing power and performance. In future works the extraction of additional G-buffers could improve the overall quality and even real-time rendering might be considered.

References

[1] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner *et al.*, "State of the art on neural rendering," in *Computer Graphics Forum*, vol. 39, no. 2. Wiley Online Library, 2020, pp. 701–727.

[2] S. R. Richter, H. A. AlHaija, and V. Koltun, "Enhancing photorealism enhancement," *arXiv:2105.04619*, 2021.

[3] T. Akenine-Möller, E. Haines, and N. Hoffman, *Real-time rendering*. AK Peters/crc Press, 2019.

[4] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[5] Rockstar Games, "Policy on posting copyrighted rockstar games material," https://support.rockstargames.com/articles/200153756/Policy-on-posting-copyrighted-Rockstar-Games-material, accessed: 2021-12-03. [Online]. Available: https://support.rockstargames.com/articles/200153756/Policy-on-posting-copyrighted-Rockstar-Games-material

[6] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[7] "Unity," https://unity.com/, accessed: 2022-01-24. [Online]. Available: https://unity.com/

[8] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[9] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell, "Cycada: Cycle-consistent adversarial domain adaptation," in *International conference on machine learning*. PMLR, 2018, pp. 1989–1998.

[10] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CVPR*, 2017.

[11] P. Li, X. Liang, D. Jia, and E. P. Xing, "Semantic-aware grad-gan for virtual-to-real urban scene adaption," *arXiv preprint arXiv:1801.01726*, 2018.

[12] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 172–189.

[13] P. Krähenbühl, "Free supervision from video games," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2955–2964.

[14] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *European Conference on Computer Vision (ECCV)*, ser. LNCS, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., vol. 9906. Springer International Publishing, 2016, pp. 102–118.

[15] Unity Technologies, "Image synthesis for machine learning," https://bitbucket.org/Unity-Technologies/ml-imagesynthesis, accessed: 2022-01-08. [Online]. Available: https://bitbucket.org/Unity-Technologies/ml-imagesynthesis

[16] B. Karlsson, "Renderdoc," https://renderdoc.org/, accessed: 2022-01-08. [Online]. Available: https://renderdoc.org/

[17] ——, "Renderdoc," https://renderdoc.org/docs/python_api/index.html, accessed: 2022-01-08. [Online]. Available: https://renderdoc.org/docs/python_api/index.html

[18] ——, "Quick start," https://renderdoc.org/docs/getting_started/quick_start.html, accessed: 2022-01-08. [Online]. Available: https://renderdoc.org/docs/getting_started/quick_start.html

[19] H. A. Alhaija, S. K. Mustikovela, A. Geiger, and C. Rother, "Geometric image synthesis," 2018.

[20] A. Cherian and A. Sullivan, "Sem-gan: Semantically-consistent image-to-image translation," 01 2019, pp. 1797–1806.

[21] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 06 2017.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 06 2016, pp. 770–778.

[23] J. Liu, Y. Zou, and D. Yang, "Semanticgan: Generative adversarial networks for semantic image to photo-realistic image translation," 05 2020, pp. 2528–2532.

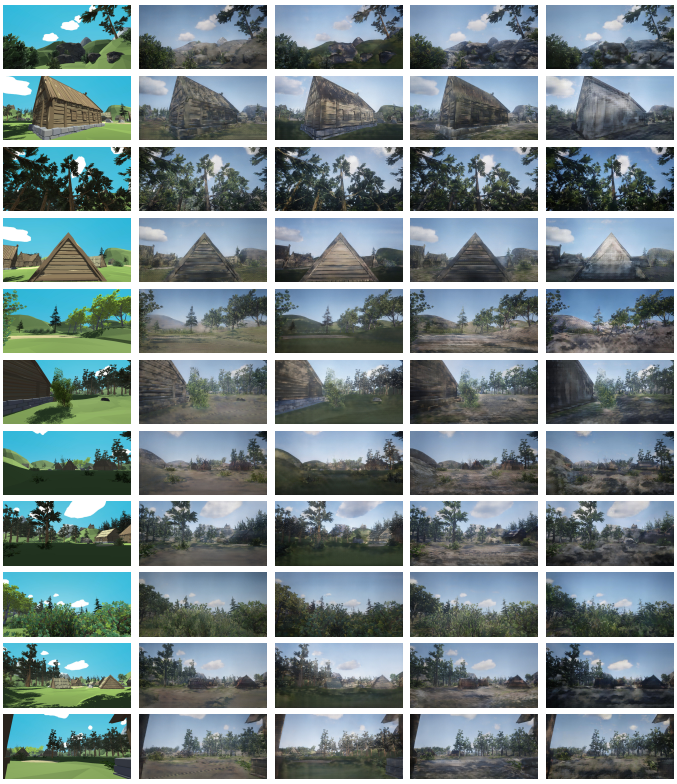
[24] Z. Hao, A. Mallya, S. Belongie, and M.-Y. Liu, "Gancraft: Unsupervised 3d neural rendering of minecraft worlds," *arXiv preprint arXiv:2104.07659*, 2021.

[25] 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," *Advances in neural information processing systems*, vol. 30, 2017.

[26] F. Yu, V. Koltun, and T. Funkhouser, "Dilated residual networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 472–480.

[27] T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu, "Contrastive learning for unpaired image-to-image translation," in *European Conference on Computer Vision*, 2020.

Appendix



(a) Unity (b) Version 1 (c) Version 2 (d) Final (e) Baseline

Fig. 13: Further results of translations (zoom in to view pictures in full quality). Version 1 (b) results contain few sharp edges caused by the simplicity of Unity normals maps. Version 2 (c) has the problem that the GAN relies on the original G-Buffers too much, e.g. mapping the colors of the albedos map with almost no translations. Our final method (d) occasionally display visually unpleasant artifacts in form of color banding, that is most likely due to the Unity depth maps containing the same banding patterns.