Deep CG2Real: Synthetic-to-Real Translation via Image Disentanglement

Sai Bi[1] Kalyan Sunkavalli[2] Federico Perazzi[2]
Eli Shechtman[2] Vladimir G. Kim[2] Ravi Ramamoorthi[1]
[1]UC San Diego [2]Adobe Research

Abstract

We present a method to improve the visual realism of low-quality, synthetic images, e.g. OpenGL renderings. Training an unpaired synthetic-to-real translation network in image space is severely under-constrained and produces visible artifacts. Instead, we propose a semi-supervised approach that operates on the disentangled shading and albedo layers of the image. Our two-stage pipeline first learns to predict accurate shading in a supervised fashion using physically-based renderings as targets, and further increases the realism of the textures and shading with an improved CycleGAN network. Extensive evaluations on the SUNCG indoor scene dataset demonstrate that our approach yields more realistic images compared to other state-of-the-art approaches. Furthermore, networks trained on our generated "real" images predict more accurate depth and normals than domain adaptation approaches, suggesting that improving the visual realism of the images can be more effective than imposing task-specific losses.

1. Introduction

Deep learning-based image synthesis methods are generating images with increasingly higher visual quality [11, 18, 21, 36, 37] even from minimal input like latent codes or semantic segmentation maps. While impressive, one challenge with these approaches is the lack of fine-grained control over the layout and appearance of the synthesized images. On the other hand, it is possible to compose 3D scenes with a desired layout and appearance and render them to create photo-realistic images. However, this requires high-quality scene assets (geometries, materials, lighting) and compute-heavy physically-based rendering.

The goal of this work is to combine the advantages of these two approaches. Given a low-quality synthetic image of a scene—the coarse models in the SUNCG indoor scene dataset [32] rendered with a simple rendering engine like OpenGL—we train a deep neural network to translate it to a high-quality realistic image. One approach to this problem would be to train an unpaired image-to-image translation network, like CycleGAN [43], from synthetic OpenGL images to real photographs. However, the unpaired nature of this problem—it is not clear how we would create a dataset of synthetic images with their "real" counterparts—makes it challenging and results in images with signficiant artifacts like implausible lighting and texture distortions (Fig. 1(b)). In contrast, our result retains the layout and coarse appearance of the original image, but introduces realistic lighting including global illumination, improves the quality of the scene textures, and removes effects like aliasing (Fig. 1(c)).

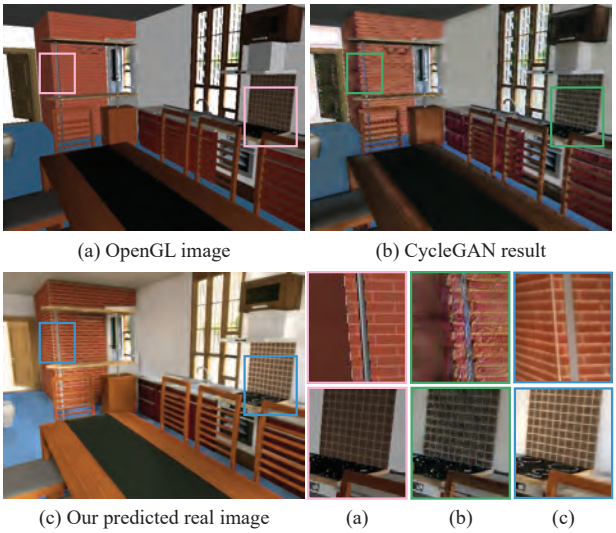


Figure 1: Our two-stage adversarial framework translates an OpenGL rendering (a) to a realistic image (c). Compared to single-stage prediction with CycleGAN (b), our result has more realistic illumination and better preserves texture details, as shown in the insets. (Best viewed in digital).

Improving the realism of a synthetic image requires improving the quality of both illumination and texture. Moreover, these two aspects need to be handled in different ways: illumination needs to be synthesized globally while textures can be modified locally. To this end, we propose disentangling a synthetic image into its constituent shading and albedo layers—i.e., an intrinsic image decomposition [1]—and training *separate* translation networks for each of them.

We leverage the intrinsic images disentanglement to change this problem from a purely unpaired setting to a

two-stage paired-unpaired setting. We render the synthetic scenes with a physically-based renderer ("PBR") to simulate realistic illumination and create paired OpenGL-PBR shading data. In the first stage of our pipeline, we use this data to train an OpenGL-to-PBR shading translation network that synthesizes realistic shading. We combine this new shading with the original OpenGL textures to reconstruct our intermediate PBR images.

In the second stage, we translate these PBR images to the real image domain in an unsupervised manner using a CycleGAN-like network. We train individual PBR-to-real generators for the shading and albedo layers; we use an encoder-decoder architecture for shading to increase global context, and a purely convolutional network, with no downsampling/unsampling for albedo. As in CycleGAN, the quality of this translation is best with a backward real-to-PBR cycle, and a cycle-consistency loss. In the absence of the disentangled shading-albedo layers for real images, we accomplish this with an asymmetric architecture and a PBR-domain intrinsic image decomposition network.

While our focus is on improving visual quality, our method can be used for domain adaptation. Deep networks can be trained on large-scale, labeled synthetic datasets [28, 41] and prior work has looked at adapting them to improve their performance on real data [9, 30, 25]. Many of these methods impose a task-specific loss on this adaptation [14, 26, 42]. In contrast, we show that by improving the overall visual realism of synthetic data, we can achieve similar improvements in real image performance on tasks such as normal and depth estimation without the need for task-specific losses, as demonstrated in Table 2 and Table 3.

2. Related Work

Image-to-Image Translation. To improve the realism of synthetic images, Johnson et al. [20] retrieve similar patches to the input from real image collections to synthesize realistic imagery. Recently deep neural networks have been widely used for this task. When paired training data is available, previous methods have proposed training conditional generative models with a combination of supervised reconstruction losses and adversarial losses (pix2pix [18], StackGAN [40]). Such mappings are challenging to learn in the unsupervised setting with only adversarial losses, and prior work has utilized cycle-consistency losses (CycleGAN [43]) or a shared latent space for the two domains (UNIT [24]). These approaches have been extended to handle multi-modal output (BicyleGAN [44], MUNIT [15]), multiple domains [5], higher-resolution images (pix2pix-HD [36]), and videos [35]. These methods—especially the unsupervised approaches—can introduce undesired structural changes and artifacts when there is a large domain difference, as there is between synthetic OpenGL images and real images. We handle this by working in a disentangled shading-albedo space. This allows us to a) use a two-stage pipeline that goes from OpenGL images to PBR images and then to the real domain, and b) design separate shading and albedo networks to avoid artifacts.

Domain adaptation. Domain adaptation methods seek to generalize the performance of a "task" network, trained on one domain, to another domain; for example, to train networks on large-scale labeled synthetic datasets and apply them on real images. Feature-space domain adaptation methods either match the distributions of source and target domain features [33] or learn to produce domain-agnostic features using feature-space adversarial losses [7, 8, 34].

Instead, image-space domain adaptation methods seek to match image distributions. The key challenge here is to avoid changing image content in ways that will impair the performance of the task network. This can be handled by using paired source-target data to regularize the translation [9]. In the unpaired setting, prior work has constrained the translated images to be close to the source images [30] but this only works for small domain shifts. Most current methods use a combination of task-specific losses (i.e., preserving the task network's output after translation) [25], image-space and feature-space adversarial losses, cycle-consistency losses, and semantic losses (i.e., preserving the semantics of the image after translation) [14, 26, 42].

Our contributions are orthogonal to this direction of work. We demonstrate that translating images in the shading-albedo space leads to higher visual quality, which improves performance on real data. We do this without using task-specific or semantic losses and are thus not constrained to a specific task. Our work can be combined with other domain adaptation ideas to further improve results.

3. Method

Our goal is to improve the visual realism of a low-quality synthetic image. In particular, we focus on translating images of indoor scenes, I_o, from the domain of OpenGL-rendered images $\mathcal{O}$, to the domain of real photographs, $\mathcal{R}$. This is an unpaired problem and has to be learned without direct supervision. The translation has to handle two aspects: first, simple rendering engines like OpenGL do not model complex, real-world lighting, and second, synthetic scenes usually do not have realistic real-world materials and textures. We propose handling this by explicitly manipulating the synthetic shading and albedo separately. Moreover, we find that directly translating from OpenGL to real images is challenging due to the large domain gap between them; changing OpenGL shading to real-world shading requires large non-local transformations that are challenging to learn in an unsupervised manner. However, synthetic scenes can be rendered with physically-based renderers that can generate more realistic shading. We leverage this to propose a two-stage translation. First, we translate the OpenGL

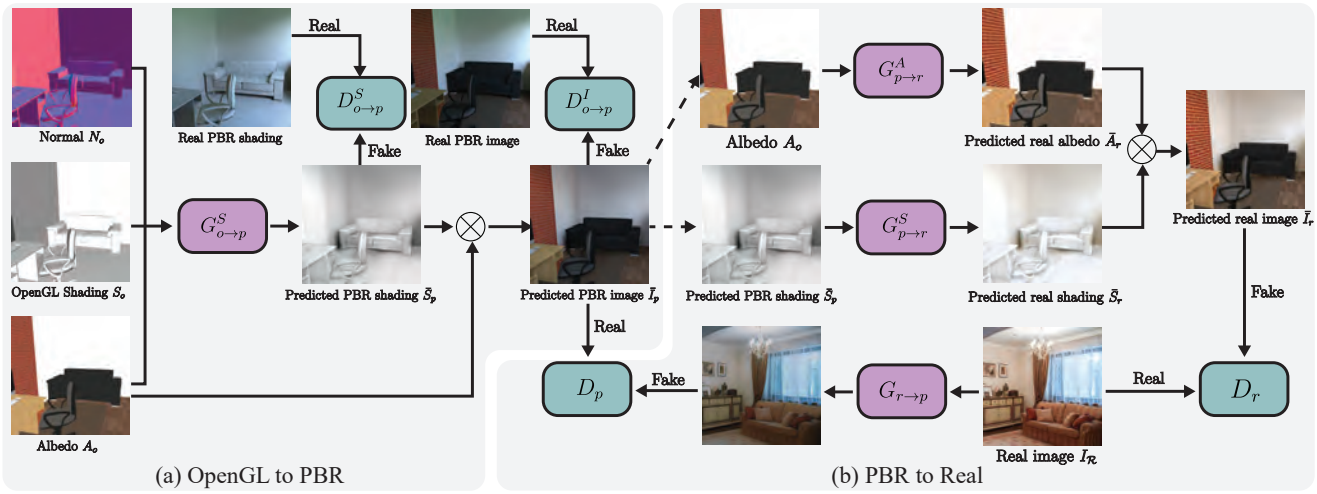


Figure 2: The framework of our two-stage OpenGL to real translations.

image I_o to the physically-based rendering (PBR) domain, $\mathcal{P}$, by transforming only the shading using paired OpenGL-PBR images. We then translate these PBR images to the real domain, $\mathcal{R}$, by refining both the albedo and shading using two separate networks; this is done in an unsupervised manner. Figure 2 shows an overview of our two-stage framework, and we describe them in following sections.

3.1. OpenGL-to-PBR Image Translation

While rendering scenes with OpenGL is computationally fast, the resulting images have low visual realism (e.g., see Figure 1(a)). One of the reasons is that standalone OpenGL only supports simple lighting models (such as directional or point lights) and does not model complex lighting effects like global illumination. In comparison, a physically based renderer can generate images with photorealistic shading by simulating the propagation of light through a scene, albeit at the cost of significantly more processing time. Therefore, we render the same scene with OpenGL and a physically based renderer Mitsuba [19] respectively. Both these images have the same geometry and material properties, and differ only in the quality of the shading. We train a neural network on these image pairs to translate the OpenGL image to the PBR domain; this network thus learns to synthesize more realistic shading from a simple OpenGL scene, bypassing the cost of physically-based rendering.

We use paired conditional generative adversarial networks [18, 36] for this task. Let I_o, I_p be a pair of images in the OpenGL and PBR domains, respectively. Since both the images differ only in their shading components, we decompose them into albedo and shading layers and train a shading generator, $G^S_{o\to p}(\cdot)$, that transforms OpenGL shading, S_o, to PBR shading, S_p. Instead of using only the OpenGL shading as input, we use the auxiliary buffers of the synthetic scene including albedo A_o and surface normals, N_o, as the input to the generator. These additional buffers encode semantic (via albedo) and geometric (via normals and shading) information about the synthetic scene and aid in the translation. Finally, we multiply the synthesized PBR shading, $\bar{S}_p$, with the original OpenGL albedo, A_o, to reconstruct the PBR image $\bar{I}_p$:

$$\bar{S}_p = G^S_{o\to p}(S_o, A_o, N_o), \quad \bar{I}_p = \bar{S}_p * A_o \tag{1}$$

Our intrinsic image model assumes Lambertian shading. While this is an approximation to real world reflectance, it is sufficient for many regions of indoor images, and is widely used [2, 3, 23]. To recover high dynamic shading values, we predict the shading in the logarithmic space.

Similar to pix2pixHD [36], we use a combination of a perceptual reconstruction loss (based on VGG features) and adversarial losses. We utilize adversarial losses via a discriminator on the predicted image, $D^I_{o\to p}$, as well as one on the predicted shading, $D^S_{o\to p}$. This ensures that both the generated images and shadings are aligned with the distribution of PBR images and shading. We used conditional discriminators that also take albedo and normals as input.

By translating only the shading and multiplying it with

(a) Without $D^S_{o\to p}$ (b) With $D^S_{o\to p}$

Figure 3: With the shading discriminator, our network is able to predict more accurate shading images and get rid of problems such as inconsistent colors.

the original albedo, we ensure that the textural structure of the albedo is explicitly preserved. Synthesizing the shading separately is also an easier task for the network because the shading component is spatially smooth and does not have the high-frequency details of the image. Moreover, this also allows us to incorporate the shading discriminator, $D^S_{o\to p}$, that provides stronger supervision for training. As shown in Figure 3, without the shading discriminator the network predicts unrealistic illumination with inconsistent colors while our full network avoids this problem.

3.2. PBR-to-Real Image Translation

The first stage of our pipeline generates images with more physically accurate shading and global illumination. However, these images may still have a domain shift from real photographs. The albedos are still from the low-quality scenes and may have unrealistic colors. The shading is also still a function of the synthetic scenes and might not align with the intensity, color, contrast and spatial distribution of real-world scenes. In the second stage, we seek to bridge this remaining domain gap to a target real image dataset.

Unlike the first stage, there is no one-to-one correspondence between predicted PBR images and real images. Zhu et al. [43] introduced the CycleGAN framework to tackle such unpaired problems and we build on their framework for our second stage. Different from the original CycleGAN network which performs the translation in image space, we propose translating the albedo and shading components separately with different generators. This novel design is based on the insight that the albedo contains high frequency details, and should be modified locally to better preserve structural details. On the other hand, shading is a global phenomenon (as it is a function of scene and light source geometry) and should take global context into consideration. Similar observations have been made in [10], which uses different operators for global irradiance and local radiosity.

The output of the first stage is the predicted PBR image, $\bar{I}_p$. As noted before, this is the product of the predicted (OpenGL-to-PBR) shading, $\bar{S}_p$, and the original scene albedo, A_p (which is the same as A_o). As shown in Figure 2, to translate from the PBR domain to the real domain ($p \to r$), we use two generators $G^A_{p\to r}(\cdot)$ and $G^S_{p\to r}(\cdot)$ to synthesize the real albedo and shading, $\bar{A}_r$ and $\bar{S}_r$, respectively. The final predicted real image, $\bar{I}_r$, can then be reconstructed as:

$$\bar{A}_r = G^A_{p\to r}(A_p),\ \bar{S}_r = G^S_{p\to r}(\bar{S}_p), \quad \bar{I}_r = \bar{A}_r * \bar{S}_r \quad (2)$$

We use different architectures for G^A_p and G^S_p. For the albedo, we use a fully convolutional network without downsampling or upsampling blocks. This results in a small receptive field for the network and better preserves the texture details while avoiding large structural changes [16, 17]. As shown in Figure 6, allowing downsampling blocks in the albedo generator leads to serious high-frequency artifacts in the textures. Our architecture removes this problem and achieves results with higher quality. In contrast, the shading generator uses downsampling blocks for a larger receptive field in order to allow global changes.

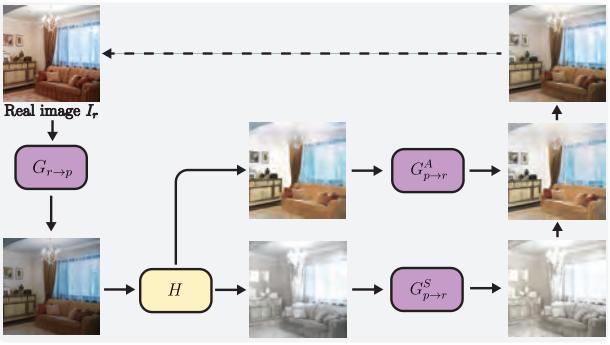


Figure 4: To finish the backward cycle, the real image is first translated to the PBR domain. Afterwards we use the pretrained intrinsic decomposition network H to decompose it into its albedo and shading, which are further fed to corresponding generators. Finally we multiply the output albedo and shading to reconstruct the original real image.

Similar to CycleGAN, we constrain the forward $p \to r$ translation using a backward $r \to p$ translation. Unlike the PBR domain, we do not have access to albedo and shading layers for real images. Therefore, we use an image-space generator $G_{r\to p}(\cdot)$ that transforms real images to the PBR domain. We utilize two discriminators, D_r and D_p, to distinguish real and fake samples in real and PBR domains respectively. Here, D_r distinguishes between the PBR images translated to the real domain ($\bar{I}_r$ from Equation 2) and real images. D_p on the other hand discriminates between PBR image (synthesized from the first stage), $\bar{I}_p$, and real images translated to the PBR domain, $G_{r\to p}(I_r)$. Note that while the generators for the $p \to r$ direction are applied to albedo and shading, the discriminator is applied to the image computed as their product. We train the network by optimizing the standard GAN loss $\mathcal{L}_{\text{GAN}}(G^A_{p\to r}, G^S_{p\to r}, D_r)$ and $\mathcal{L}_{\text{GAN}}(G_{r\to p}, D_p)$ for the forward translation $p \to r$ and backward translation $r \to p$, respectively.

Only having the GAN loss is not sufficient for learning meaningful translations because of the lack of pixel-level correspondence [39]. Similar to CycleGAN, we also include forward and backward cycle consistency losses. The forward cycle consistency, $p \to r \to p$ is trivial: we project the predicted real image $\bar{I}_r$ back to the PBR domain by feeding it to the generator $G_{r\to p}$, and minimize the L_1 loss between the output and the PBR source image $\bar{I}_p$:

$$\begin{aligned}\mathcal{L}_{\text{for}}(G^A_{p\to r}, G^S_{p\to r}, G_{r\to p}) &= ||G_{r\to p}(\bar{I}_r) - \bar{I}_p||_1 \\ &= ||G_{r\to p}(G^A_{p\to r}(A_p) * G^S_{p\to r}(\bar{S}_p)) - \bar{I}_p||_1 \quad (3)\end{aligned}$$

Specifying the backward cycle is more challenging. We

can map a real image I_r to the PBR domain as $G_{r\to p}(I_r)$. However, translating it back to the real domain via the forward $p \to r$ translation requires an albedo-shading separation that we do not have for these images—we only have them for our original synthetic images. We tackle this by training an intrinsic decomposition network [23, 3, 4] to predict the albedo and shading layers for PBR images.

Let H be the intrinsic decomposition network. Given a PBR image I and corresponding albedo A_I and shading S_I, H is trained by optimizing the following loss function:

$$\mathcal{L}(H) = ||H_A(I) - A_I||_2^2 + ||H_S(I) - S_I||_2^2, \quad (4)$$

where $H_A(I)$ and $H_S(I)$ are the predicted albedo and shading by the network H. We adopt the network architecture used in Li et al. [23], which contains one encoder and two decoders with skip connections. While they use a scale-invariant mean square error (MSE) loss, we use MSE because we require the product of the albedo and shading to be identical to the original image. The intrinsic decomposition network H is pretrained on the predicted PBR images from the first stage, $\bar{I}_p$, where we have ground truth albedo and shading. Afterwards, it is fixed during the training of the image translation networks.

We use the intrinsic decomposition network, H, to decompose $G_{r\to p}(I_r)$ into its albedo and shading. Then we can translate each component via the forward $p \to r$ translation to synthesize the result of the full backward cycle, leading to the following backward cycle consistency loss:

$$\begin{aligned} I'_p &= G_{r\to p}(I_r) \\ I''_r &= G^S_{p\to r}\big(H_S(I'_p)\big) * G^A_{p\to r}(H_A(I'_p))\big) \\ \mathcal{L}_{\text{back}}(G^A_{p\to r}, G^S_{p\to r}, G_{r\to p}) &= ||I''_r - I_r||_1 \end{aligned} \quad (5)$$

Figure 4 shows the formulation of our backward cycle.

Note that our network is asymmetric; the forward translation takes our PBR albedo and shading layers, translates and combines them to construct the result. Our backward translation does the opposite; it first translates real images to the PBR domain and then decomposes them there. This allows us to bypass the requirement for real albedo-shading data, and instead train a PBR albedo-shading decomposition *for which we have ground truth supervision*. As can be seen in Figure 5, utilizing this novel backward consistency cycle significantly reduces artifacts and improves the visual quality of our PBR-to-real translations.

Our final loss function for PBR-to-real translation is a combination of the two GAN losses (from discriminators D_p and D_r), and the forward and backward cycle consistency losses (Equations 3 and 5).

4. Implementation

OpenGL-to-PBR. Our first stage network architecture is based on pix2pixHD [36]. We use a 70×70 PatchGAN [18] for the discriminator. The generator $G^S_{o\to p}$ contains two sub-nets including a global and a local network each with 9 and 3 residual blocks as proposed by Wang et al. [12]. While the shading directly predicted by the generator is reasonable, it may have noise, discontinuities or typical GAN blocking artifacts, which degrade the finale image quality. We leverage the inherent spatial smoothness of the shading and remove potential artifacts by applying a *guided filter layer* [38] after the output layer of the shading generator. The guided filter layer $h(\cdot)$ takes the OpenGL shading, S_o, and translated OpenGL shading, $G^S_{o\to p}(S_o)$, as input, and outputs the predicted shading, $\bar{S}_p$. We set radius to 4 and regularization parameter to 0.01.

(a) Without $\mathcal{L}_{\text{back}}$ (b) With $\mathcal{L}_{\text{back}}$

Figure 5: Without the backward cycle, the network tends to generate outputs with undesired new structures. Adding the backward cycle with a pretrained intrinsic decomposition network is able to generate images with higher quality.

(a) With downsampling (b) Without downsampling

Figure 6: Comparison between with and without downsampling blocks in the albedo generator. From the result, we can see that the generator without downsampling blocks could better preserve the texture structures of the input.

PBR-to-Real. For our second stage, we use the same PatchGAN architecture for the discriminator. Both the shading generator $G^S_{p\to r}$ and real-to-PBR generator $G_{r\to p}$ have 2 convolutional blocks with stride of 2 to downsample the input, followed by 4 residual blocks and 2 convolutional blocks with upsampling layers. The downsampling/upsampling gives the shading generator a large receptive field allowing it to capture global information about the scene and make global changes such as adjusting shading colors and intensity. The albedo generator $G^A_{p\to r}$ has a similar architecture as $G^S_{p\to r}$, except that it does not have any downsampling/upsampling. This keeps the receptive field small and forces the albedo generator to only modify the

albedo locally thereby preserving textural details.

Training data. For the OpenGL and PBR images we use the synthetic dataset from Li et al. [23], which contains about 20000 480×640 images of indoor scenes from the SUNCG dataset [32], rendered with both OpenGL and Mitsuba [19], a physically based renderer. We use 5000 image pairs for training the first stage. Then the first-stage network is used to translate another 5000 OpenGL images to the PBR domain, which are used for second-stage training. Two real image datasets are used in the second stage: the real estate dataset from Poursaeed et al. [27] and the NYUv2 dataset [31]. The real estate dataset contains high quality real images of indoor scenes, and we use it for comparison on image quality (Section 5.1). The NYUv2 dataset is used for domain adaptation experiments (Section 5.2). Finally an extra 5000 OpenGL images are used for testing the full pipeline. We select the training and testing OpenGL images from different scenes to avoid overlap.

Training details. Both stages are trained separately. We use Adam [22] with an initial learning rate of 0.0002 for training both stages. The networks are trained for 100 epochs, with the first 50 epochs trained with the initial learning rate, and the remaining 50 epochs with a linearly decaying learning rate. We randomly crop patches from the input images for training with a patch size of 400×400 for the first stage and 256×256 for the second stage.

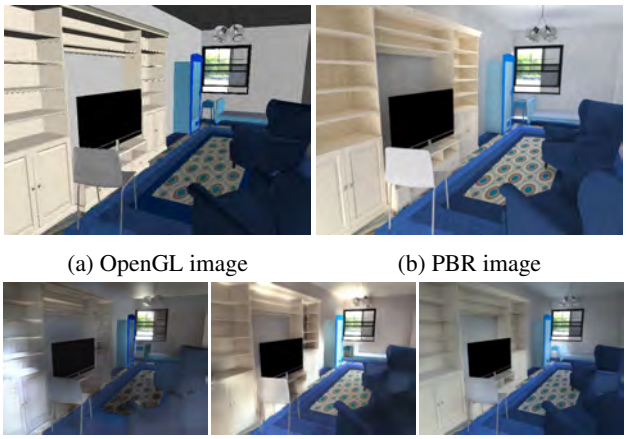
5. Results

In this section, we first show the comparison on visual quality of translations against baseline methods (Section 5.1). Afterwards we show that our two stage pipeline could boost the performance of network models on real images when trained with our translated images (Section 5.2).

5.1. Comparison on visual quality

We compare the results of our method against different baselines for each separate stage and the whole pipeline. In addition to qualitative comparisons, we also quantitatively measure the visual realism of images generated with different methods with two metrics. The first is the FID score [13]which has been shown to be effective in measuring the distance between two distributions and is consistent with image noise and distortion. In addition, we also conduct a human perceptual study on Amazon Mechanical Turk. For each task, workers are shown the images output by different methods, and asked to select the most realistic result with the fewest artifacts. For each task, we have 100 images, each evaluated by 10 different workers, for a total of 1000 judgements. We discuss each stage in detail below.

OpenGL to PBR translation. Our proposed network takes the albedo, normal and OpenGL shading as input to predict the PBR shading, which is multiplied by the albedo to reconstruct the desired PBR image. We compare our method with two baselines: the pix2pix-HD network [36] that predicts the PBR image conditioned on the OpenGL image only (denoted as *p2pHD-OpenGL* in Table 1), and the same network that takes the same input as ours but directly predicts the PBR images (*p2pHD-S+A+N*). Both baselines are trained with the same generator architecture and parameter settings as ours. The only differences are the generator inputs/outputs and the lack of shading discriminators.



(a) OpenGL image (b) PBR image

(c) p2pHD-OpenGL (d) p2pHD-S+A+N (e) Our result

Figure 7: OpenGL to PBR comparison. A PBR renderer like Mitsuba needs around 20 mins to render a noise-free image as in (b). In comparison our network can generate a high quality result in 0.03 secs. (e). Compared to the pix2pix-HD framework that directly predicts the output images using OpenGL (c) or auxiliary buffers (d), with inconsistent shading such as abrupt highlights on the cabinet, our method generates images with much higher visual realism.

We calculate the FID scores between the predicted PBR images and ground truth PBR images, and we also conduct a user study to ask workers to select the most visually realistic PBR predictions among outputs of three methods. From the results in Table 1 ($\mathcal{O} \rightarrow \mathcal{P}$), we can clearly see that our method achieves a much lower FID score compared to other baselines, implying that our predictions are more aligned with the distribution of the ground truth PBR images. In addition, we obtain a user preference rate of 60.6%, much higher than the other two baselines, demonstrating that doing predictions in shading space generates images with much higher quality. A visual comparison is shown in Figure 7.

PBR to Real translation. In this stage, we train the network to translate the output of the first stage to the real domain. We compare to the naïve CycleGAN framework that performs the translations in image space. FID score is calculated between the translated images and real images from the real estate dataset. Similarly, from Table 1 we can see that disentangled translations on albedo and shading with our proposed framework achieves lower FID scores

		FID	User Preference
$\mathcal{O} \rightarrow \mathcal{P}$	p2pHD-OpenGL [36]	21.01	10.2%
	p2pHD-S+A+N [36]	11.63	29.2%
	Ours	**7.40**	**60.6%**
$\mathcal{P} \rightarrow \mathcal{R}$	CycleGAN [43]	54.80	27.9 %
	Ours	**53.48**	**72.1%**
$\mathcal{O} \rightarrow \mathcal{R}$	CycleGAN [43]	59.42	16.9%
	T^2Net [42]	65.33	4.7%
	Ours	**53.48**	**78.4%**

Table 1: Comparison on FID and user preference score.

(a) Predicted PBR (b) CycleGAN

(c) Ours (b) (c)

Figure 8: PBR to Real comparison. Given the predicted PBR image (a) in the first stage, we further translate it to real domain. Compared to the original CycleGAN [43] that performs the translation in image space, our method could better preserve the structure of the textures and generate images with many fewer artifacts.

and much higher user preference rate. As shown in Figure 8, our network is able to modify the albedo and shading of the input while preserving the texture structures. In contrast, the original CycleGAN network introduces obvious artifacts which degrade image quality.

Full pipeline. We also compare our two-stage translations to previous methods that translate directly from OpenGL to real: the CycleGAN method and T^2Net [42]. T^2Net uses a single generator and discriminator for synthetic to real translation. In addition to standard GAN loss, it introduces an identity loss to guarantee that when real images are fed to synthetic generator, the outputs are similar to the inputs. The results show that our full pipeline significantly outperforms the single-stage translations used in both baselines. The visual comparison in Figure 9 shows that single-stage translation is not able to produce realistic shading and generates results with noise and artifacts due to the large gap between OpenGL images and real images, which our proposed method addresses satisfactorily.

		Lower is better		Higher is better(%)		
		Mean	Median	< 11.25	< 22.5	< 30
$\mathcal{O} \rightarrow \mathcal{P}$	OpenGL	37.73	31.91	17.93	38.36	48.52
	Ours-PBR	34.82	27.66	21.16	42.82	53.25
$\mathcal{P} \rightarrow \mathcal{R}$	CycleGAN [43]	33.90	27.24	22.28	43.99	54.38
$\mathcal{O} \rightarrow \mathcal{R}$	CycleGAN [43]	36.33	30.25	19.15	39.19	50.15
	T^2Net [42]	36.93	30.93	18.93	39.23	50.49
	Ours-full	**33.15**	**26.27**	**23.52**	**44.95**	**55.28**
	Real*	28.18	21.95	28.14	52.21	62.35

Table 2: Normal estimation results on NYUv2 dataset.

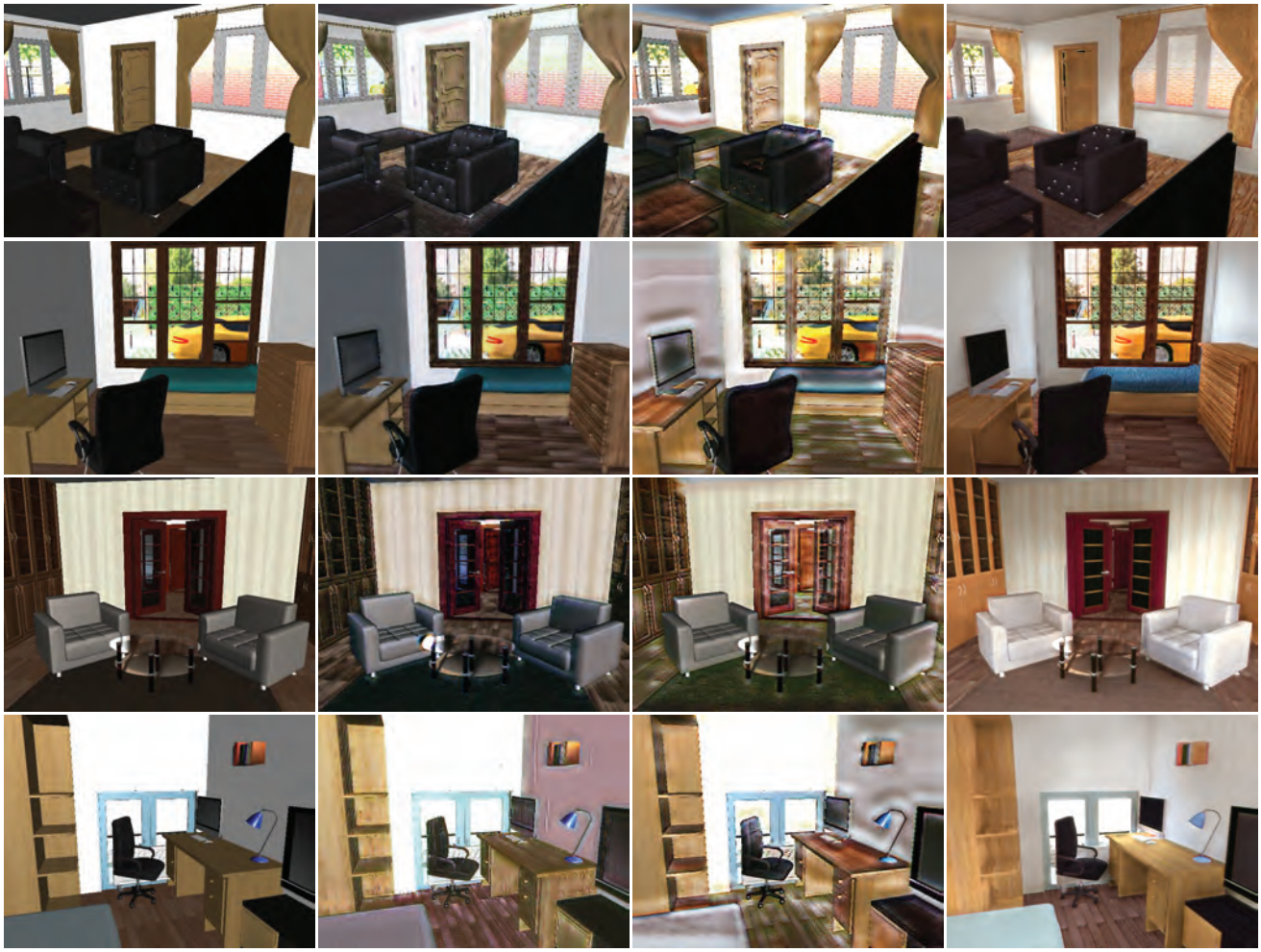
		Lower is better		Higher is better (%)		
		RMSE	RMSE-log	< 1.25	$< 1.25^2$	$< 1.25^3$
$\mathcal{O} \rightarrow \mathcal{P}$	OpenGL	1.0770	0.3873	43.53	73.60	89.75
	Ours-PBR	1.0293	0.3514	46.23	75.41	91.35
$\mathcal{P} \rightarrow \mathcal{R}$	CycleGAN [43]	0.9824	0.3394	48.24	78.61	92.25
$\mathcal{O} \rightarrow \mathcal{R}$	CycleGAN [43]	1.0328	0.3726	45.46	75.49	91.13
	T^2Net [42]	1.0085	0.3548	47.49	77.57	91.94
	Our-full	**0.9774**	**0.3328**	**49.59**	**79.54**	**93.14**
	Real*	0.7070	0.2516	67.76	88.75	96.35

Table 3: Depth estimation results on NYUv2 dataset. *Real** is the model trained on 5000 real images from NYUv2.

5.2. Domain adaptation

Network models trained with synthetic images often perform poorly on real images due to the domain gap. Our two-stage network improves the visual realism of synthetic images, thereby boosting the performance of network models trained with the translated images. We compare the performance of network models trained with images generated with different methods on two tasks including normal estimation and depth estimation. We train the task networks on 5000 images from SUNCG dataset after translation to the real domain, and evaluate them on the test dataset in NYUv2, which contains 1449 images with ground truth normal and depth.

Normal estimation. We use a U-Net [29] with 7 downsampling blocks for the normal estimation task, and apply the inverse dot product between ground truth normal and predicted normal as the loss function. The network is trained with an Adam optimizer for 200 epochs with a learning rate of 2×10^{-4} for the first 100 epochs and a linearly decaying rate for the remaining 100 epochs. We test the network on the NYUv2 dataset. In Table 2, we report the mean and median angle between predicted normal and ground truth normal, as well as the percentage of pixels where the angle is below a certain threshold. From the table we can see compared to the model trained on OpenGL images, training on our predicted PBR images significantly reduces the average angle, and translation to the real domain further reduces it to 33.15, which demonstrates that both stages help reduce the domain gap and help improve network performance. In addition, compared to the naïve CycleGAN in



(a) OpenGL image (b) CycleGAN (c) T^{2}Net (d) Ours

Figure 9: Comparison on the full pipeline from OpenGL to Real against single stage translation with CycleGAN and T^{2}Net.

image space for the second stage, our disentangled design is also able to achieve higher accuracy. Finally, compared to going directly from OpenGL to real images with methods such as CycleGAN and T^{2}Net, our two-stage translation significantly outperforms them on all metrics.

Depth estimation. We use the network architecture in Zheng et al. [42] for depth estimation, and adopt the same training protocol. We evaluate the performance of different approaches on the NYUv2 dataset using metrics such as relative mean squared error (RMSE) between the prediction and ground truth, RMSE in log space as well as the percentage of pixels whose ratios to ground truth are below a threshold [6]. Table 3 summarizes the scores of different network models. From the table we can see that training with our predicted PBR images achieves much higher accuracy than training on the synthetic OpenGL images, and our full pipeline further improves the accuracy by translating the PBR images to real domain. Our two-stage translation is also able to outperform single-stage translation with T^{2}Net and CycleGAN and leads to a lower error.

6. Conclusion

We propose a novel two-stage framework to translate synthetic OpenGL images to the real domain. We achieve this by manipulating the albedo and shading layers of an image: we first translate them to the PBR domain by training on paired data, followed by an unsupervised translation to the real domain. We have demonstrated that our approach leads to translations with higher visual quality and better performance for domain adaptation on scene inference tasks. We believe that operating on the albedo-shading decomposition is a way to incorporate physical structure into generative models and would like to explore applications of this idea to other tasks like image synthesis, image editing and style transfer.

Acknowledgements This work was supported in part by ONR grant N000141712687, Adobe Research, a Samsung GRO grant, and the UC San Diego Center for Visual Computing. Part of this work was done while Sai Bi was an intern at Adobe Research.

References

[1] Harry G. Barrow and J. Martin Tenenbaum. Recovering intrinsic scene characteristics from images. *Computer Vision Systems*, pages 3–26, 1978. 1

[2] Sean Bell, Kavita Bala, and Noah Snavely. Intrinsic images in the wild. *SIGGRAPH*, 33(4), 2014. 3

[3] Sai Bi, Nima Khademi Kalantari, and Ravi Ramamoorthi. Deep hybrid real and synthetic training for intrinsic decomposition. In *EGSR 2018*, pages 53–63, 2018. 3, 5

[4] Lechao Cheng, Chengyi Zhang, and Zicheng Liao. Intrinsic image transformation via scale space decomposition. In *CVPR*, June 2018. 5

[5] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sung Kim, , and Jaegul Choo. StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation. In *CVPR*, 2018. 2

[6] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, pages 2366–2374, 2014. 8

[7] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, pages 1180–1189, 2015. 2

[8] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016. 2

[9] Baris Gecer, Binod Bhattarai, Josef Kittler, and Tae-Kyun Kim. Semi-supervised adversarial learning to generate photorealistic face images of new identities from 3D morphable model. In *ECCV*, September 2018. 2

[10] Reid Gershbein, Peter Schröder, and Pat Hanrahan. Textures and radiosity: Controlling emission and reflection with texture maps. In *SIGGRAPH*, pages 51–58. ACM, 1994. 4

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014. 1

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, volume 5, page 6, 2015. 5

[13] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, pages 6626–6637, 2017. 6

[14] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, and Trevor Darrell. CyCADA: Cycle consistent adversarial domain adaptation. In *ICML*, 2018. 2

[15] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *ECCV*, 2018. 2

[16] Andrey Ignatov, Nikolay Kobyshev, Radu Timofte, Kenneth Vanhoey, and Luc Van Gool. DSLR-quality photos on mobile devices with deep convolutional networks. In *ICCV*, 2017. 4

[17] Andrey Ignatov, Nikolay Kobyshev, Radu Timofte, Kenneth Vanhoey, and Luc Van Gool. WESPE: Weakly supervised photo enhancer for digital cameras. In *CVPR Workshops*, pages 691–700, 2018. 4

[18] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017. 1, 2, 3, 5

[19] Wenzel Jakob. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org. 3, 6

[20] Micah K Johnson, Kevin Dale, Shai Avidan, Hanspeter Pfister, William T Freeman, and Wojciech Matusik. CG2Real: Improving the realism of computer generated images using a large collection of photographs. *IEEE TVCG*, 17(9):1273–1285, 2010. 2

[21] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2018. 1

[22] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 6

[23] Zhengqi Li and Noah Snavely. Cgintrinsics: Better intrinsic image decomposition through physically-based rendering. In *ECCV*, 2018. 3, 5, 6

[24] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *NIPS*, pages 700–708, 2017. 2

[25] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. Ganerated hands for real-time 3D hand tracking from monocular rgb. In *CVPR*, pages 49–59, 2018. 2

[26] Zak Murez, Soheil Kolouri, David Kriegman, Ravi Ramamoorthi, and Kyungnam Kim. Image to image translation for domain adaptation. In *CVPR*, June 2018. 2

[27] Omid Poursaeed, Tomáš Matera, and Serge Belongie. Vision-based real estate price estimation. *Machine Vision and Applications*, 29(4):667–676, 2018. 6

[28] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *ECCV*, pages 102–118. Springer, 2016. 2

[29] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 7

[30] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, volume 2, page 5, 2017. 2

[31] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *ECCV*, pages 746–760. Springer, 2012. 6

[32] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *CVPR*, pages 190–198. IEEE, 2017. 1, 6

[33] Baochen Sun and Kate Saenko. Deep CORAL: Correlation alignment for deep domain adaptation. In *ECCV*, pages 443–450. Springer, 2016. 2

[34] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, volume 1, page 4, 2017. 2

[35] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In *NIPS*, 2018. 2

[36] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional GANs. In *CVPR*, 2018. 1, 2, 3, 5, 6, 7

[37] Xiaolong Wang and Abhinav Gupta. Generative image modeling using style and structure adversarial networks. In *ECCV*, pages 318–335. Springer, 2016. 1

[38] Huikai Wu, Shuai Zheng, Junge Zhang, and Kaiqi Huang. Fast end-to-end trainable guided filter. In *CVPR*, pages 1838–1847, 2018. 5

[39] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation. In *ICCV*, pages 2849–2857, 2017. 4

[40] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, pages 5907–5915, 2017. 2

[41] Yinda Zhang, Shuran Song, Ersin Yumer, Manolis Savva, Joon-Young Lee, Hailin Jin, and Thomas Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. In *CVPR*, pages 5057–5065. IEEE, 2017. 2

[42] Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai. T2Net: Synthetic-to-realistic translation for solving single-image depth estimation tasks. In *ECCV*, pages 767–783, 2018. 2, 7, 8

[43] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 1, 2, 4, 7

[44] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *NIPS*, pages 465–476, 2017. 2