Complex Scene Image Translation between Minecraft and Real World

Rujia Yang[1,*] **Zhangchen Ye**[1,*] **Fuhang Kuang**[1,*]

Abstract

Unpaired image-to-image translation between Minecraft and real-world images is a unique and challenging task due to the lack of paired datasets and significant visual differences. This study explores unpaired image-to-image translation techniques for this task, summarizing paradigms such as Cycle Consistency Constraint, Feature Matching Loss, and Shared Feature Space, and examining methods like CycleGAN[7], CUT[4], and DDIB[5]. We then fine-tuned the CycleDiffusion[6] pipeline on the MCHouse dataset, comparing the performance of the original and fine-tuned models. Our experiments show that fine-tuning significantly improves image consistency and the model's understanding of Minecraft structures in both transformation directions. This research advances the field by providing insights into effective methodologies for complex scene image translation between distinct visual domains.

1 Introduction

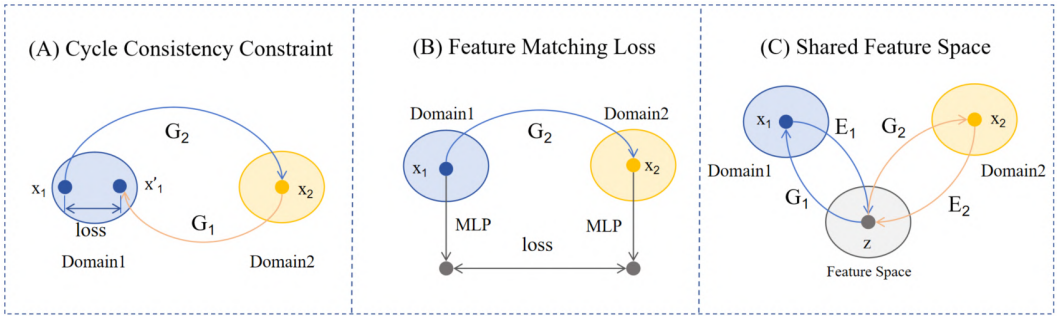

Figure 1: Comparison of Different Paradigms for Unpaired Image-to-Image Translation. (A) Cycle Consistency Constraint(CycleGAN[7], StarGAN[2]): Ensures that translating an image to another domain and back yields the original image, preserving core attributes. (B) Feature Matching Loss(CUT[4], EnCo[1]): Uses an MLP to extract high-level features and a feature matching loss to balance content preservation with style transformation. (C) Shared Feature Space(UNIT[3], DDIB[5]): Encodes images from both domains into a shared feature space, enhancing feature representation without additional loss functions, thus reducing hyper parameter tuning complexity. Here, E denotes "Encoder" and G denotes "Generator".

Description. Minecraft is a very popular and unique game, capturing the imaginations of millions of players around the world. Its pixelated art style sets it apart from the highly detailed and realistic

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

graphics seen in many modern games. If we could achieve image translation between Minecraft and the real world, it would be an intriguing and fascinating development, blending the virtual and real in a novel way.

This task of translating images between the Minecraft world and the real world is particularly challenging because there is no paired dataset available. However, the abundance of unpaired data offers a significant opportunity. Leveraging these unpaired datasets could greatly expand the scope and volume of data available for training models, potentially leading to more robust and versatile image translation capabilities.

Minecraft world and real world have an object-level corresponding. However there is huge difference up to pixel-level or shape-level. To achieve the world-image translation, we utilize unpaired image-to-image transformation techniques, leveraging the abundance of unpaired data. This field has advanced significantly with methods like CycleGAN[7], and diffusion model based method such as DDIB[5] and CycleDiffusion[6]. These methods fall into three paradigms, as shown in Figure [1]: Cycle Consistency Constraint, Feature Matching Loss, and Shared Feature Space. We have chosen representative methods from each paradigm to explore.

Figure 2: **CycleGAN[7] fails to do the complex task.**

Tryout:

- CycleGAN[7]: focusing on pixel-level supervision, which is not suitable for higher-level semantic translation.
- CUT[4]: Its results(7) are slightly better, indicating importance of learning better feature representation.
- DDIB[5]: tends to create irrelevant scenes, lacking semantic constraint between input and output.

See more details in Appendix.

2 Methodology

2.1 Principle & Fact

Researchers have found that sampling with the same seed from two separately trained diffusion models on different distributions leads to similar images. CycleDiffusion[6] leverages this fact: use the noise predicted by one diffusion model to guide another diffusion model. This is a multi-layer information injection, which contains rich high-level semantic information (close to noise space) and low-level information (close to target image domain).

2.2 Setting

- Want to translate from domain A to domain B.
- Have diffusion models over A and B respectively.

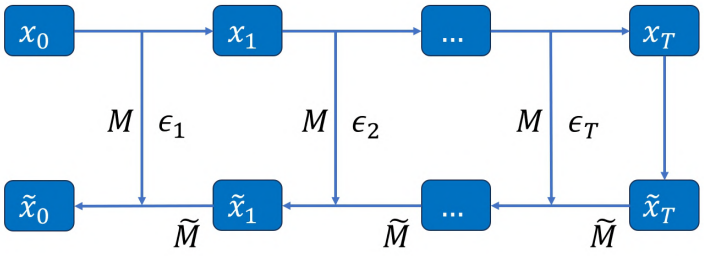

Figure 3: Pipeline of CycleDiffusion

2.3 Pipeline

1. Add noise to source image, get noisy image from each time step.
2. Predict the noises by source-domain model.
3. Denoise by target-domain model. At each denoising step, add a small perturbation (guidance) by the noises predicted by the source-domain model.

Algorithm 1: CycleDiffusion for zero-shot image-to-image translation

Input: source image $\boldsymbol{x} := \boldsymbol{x}_0$; source text $\boldsymbol{t}$; target text $\hat{\boldsymbol{t}}$; encoding step $T_{\text{es}} \leq T$

1. Sample noisy image $\hat{\boldsymbol{x}}_{T_{\text{es}}} = \boldsymbol{x}_{T_{\text{es}}} \sim q(\boldsymbol{x}_{T_{\text{es}}}|\boldsymbol{x}_0)$

for $t = T_{es}, \ldots, 1$ **do**

2. $\boldsymbol{x}_{t-1} \sim q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$
3. $\boldsymbol{\epsilon}_t = \big(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_T(\boldsymbol{x}_t, t|\boldsymbol{t})\big)/\boldsymbol{\sigma}_t$
4. $\hat{\boldsymbol{x}}_{t-1} = \boldsymbol{\mu}_T(\hat{\boldsymbol{x}}_t, t|\hat{\boldsymbol{t}}) + \boldsymbol{\sigma}_t \odot \boldsymbol{\epsilon}_t$

Output: $\hat{\boldsymbol{x}} := \hat{\boldsymbol{x}}_0$

Figure 4: Algorithm of CycleDiffusion on text-based translation

2.4 Details

- Diffusion model: Although the pre-trained diffusion model is not completely immune to having seen MC game screenshots, the data quality and quantity may not be satisfactory, so fine-tuning on a high-quality MC image dataset is necessary.
- Hyperparameters search: Diffusion model is sensitive to hyperparameters when generating pictures, and different tasks may achieve the best results under different hyperparameters. So an automatic mechanism to search through different hyperparameters and pick the best one is necessary. In the work of CycleDiffusion, this is achieved by a CLIP-guided score, while due to the particularity of this task, we incorporate SSIM score to emphasize the similarity between the generated picture and the original image.

3 Experiment

3.1 Datasets

MCHouse: Most of the images are downloaded from here. The dataset contains two sets of 512×512 resolution images: one with real-world photos and the other with Minecraft screenshots, both primarily depicting houses. Each category is divided into a training set of 1,300 images and a validation set of 150 images.

3.2 Experimental Settings

- *stable-diffusion-v1-4* as the pretrained diffusion model.
- LoRA finetune on MCHouse dataset for 500000 steps.
- Encoding text: “A house built in Minecraft(real) world."
- Hyperparameters search: CLIP score + SSIM score.

4 Results

We conducted experiments using the CycleDiffusion[6] pipeline on the **MCHouse** dataset. Our study involved a comparative analysis of the original CycleDiffusion model and a variant incorporating a fine-tuned diffusion model. The results demonstrate that after fine-tuning the diffusion model on the MCHouse dataset, we observed a significant improvement in consistency and accuracy in output of both transformation directions. These findings suggest that our work is effective, providing enhanced performance in the context of unpaired image-to-image transformation tasks.

4.1 MC2Real

Since real-world objects have more complex shapes, this direction is more challenging. We initially focused on transforming Minecraft images into real-world images, a direction that challenges the model’s capabilities due to the higher information density of real-world images. Results are shown in Figure [5].

4.2 Real2MC

Given that our pipeline is based on the paradigm of a shared feature space, it allows for the easy extension of the unpaired image-to-image transformation task to the reverse direction. Following this pipeline, we can transform a real-world image into a Minecraft-style image without any additional training procedure. This task requires the model to understand the corresponding objects in Minecraft world. The results of this translation are presented in Figure [6].

5 Conclusions & Discussion

CycleDiffusion[6] is a great work that leverages the power of pre-trained large models for zero-shot image-to-image translation. It has proven to be effective in our project. In addition, to achieve better results on our specific task, we finetune the pre-trained models and propose better evaluation metrics to guide the hyperparameter search.

References

[1] Xiuding Cai, Yaoyao Zhu, Dong Miao, Linjie Fu, and Yu Yao. Rethinking the paradigm of content constraints in unpaired image-to-image translation, 2024.

[2] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation, 2018.

[3] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks, 2018.

[4] Taesung Park, Alexei A. Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation, 2020.

[5] Xuan Su, Jiaming Song, Chenlin Meng, and Stefano Ermon. Dual diffusion implicit bridges for image-to-image translation, 2023.

[6] Chen Henry Wu and Fernando De la Torre. Unifying diffusion models’ latent space, with applications to cyclediffusion and guidance, 2022.

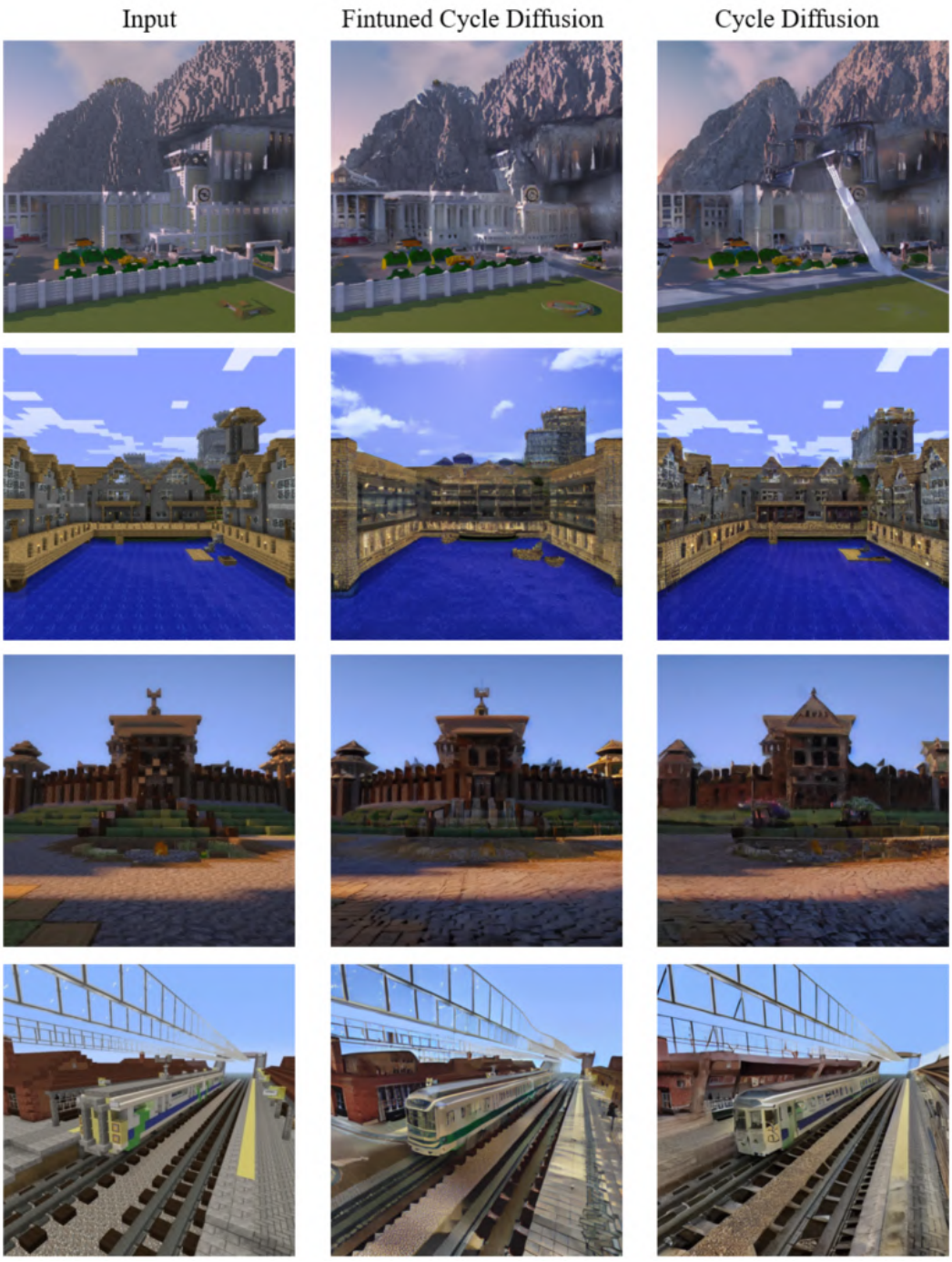

Figure 5: **Minecraft-to-Real.** In the results, We observed that using a diffusion model fine-tuned on the Minecraft dataset yielded better results, enhancing overall image consistency. The model demonstrated improved understanding of Minecraft's unique structures. For example, the clouds and waves in the second line images. It also has greater stability in handling long-tail objects, such as the manhole cover in the first line images, the boat in the second line images and the train in the fourth line images.

Figure 6: **Real-to-Minecraft.** The model translates the bricks well. Trees and grass are translated excellently.

[7] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2020.

A Appendix

A.1 CUT

Figure 7: **CUT[4] model can translate the Minecraft style to real world style, but it will not maintain the original texture and color.**

A.2 DDIB

DDIB[5] views the noising and denoising process as two Schrodinger Bridges that bridge between source domain and latent space, and between latent space and target domain. The high-level idea is: first do ODE solving from source domain to noise space, and then do reverse ODE solving from noise space to target domain (see Figure 8).

Algorithm 1 High-level Pseudo-code for DDIBs

Input: data sample from source domain $\mathbf{x}^{(s)} \sim p_s(\mathbf{x})$, source model $v_\theta^{(s)}$, target model $v_\theta^{(t)}$.
Output: $\mathbf{x}^{(t)}$, the result in the target domain.
$\mathbf{x}^{(l)} = \text{ODESolve}(\mathbf{x}^{(s)}; v_\theta^{(s)}, 0, 1)$ // obtain latent code from source domain data
$\mathbf{x}^{(t)} = \text{ODESolve}(\mathbf{x}^{(l)}; v_\theta^{(t)}, 1, 0)$ // obtain target domain data from latent code
return $\mathbf{x}^{(t)}$

Figure 8: high-level algorithm for DDIB[5]

Here are some results of conversion using DDIB.

original picture reverse ODE to source domain reverse ODE to target domain

Figure 9: DDIB conversion from MC landscape to real landscape

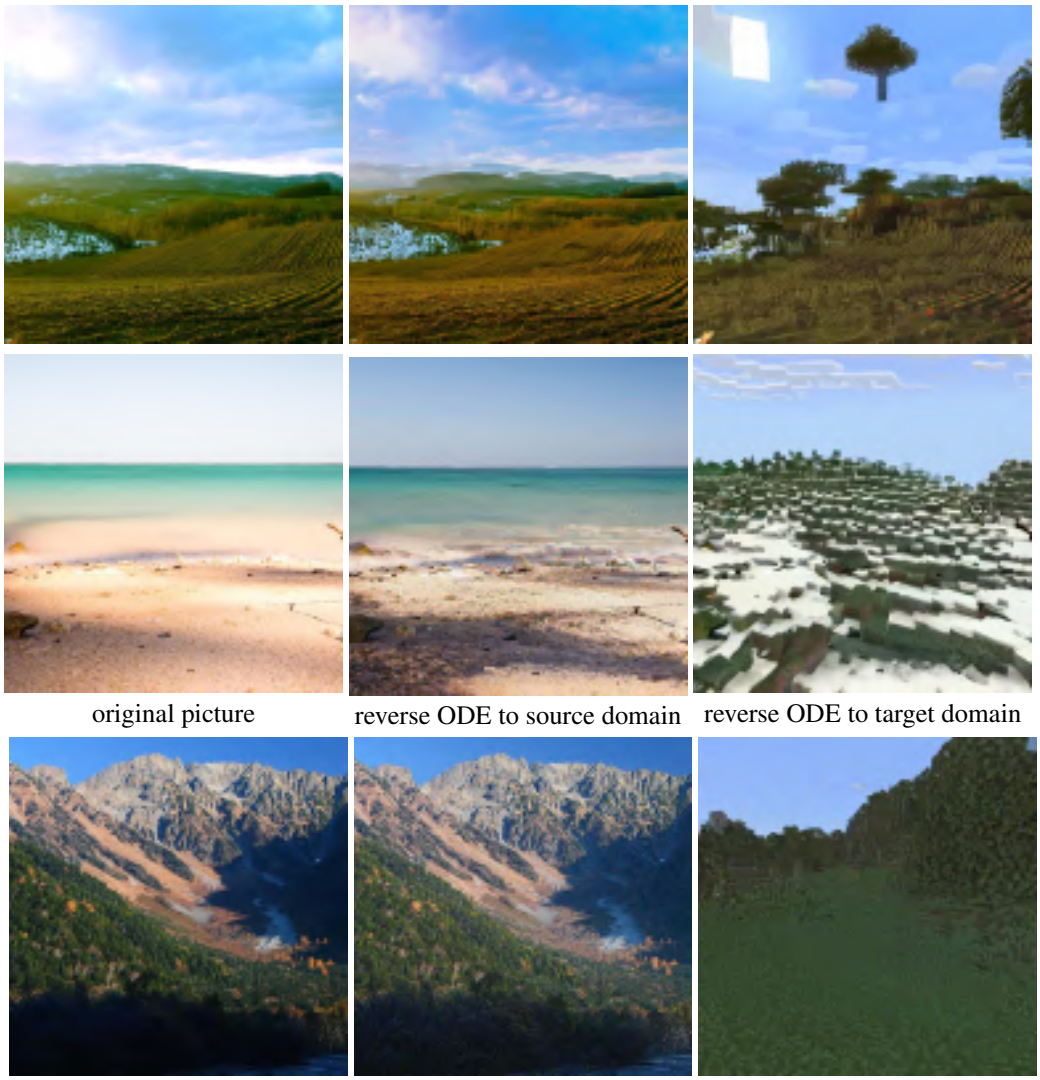

Figure 10: DDIB conversion from real landscape to MC landscape

We can see that the conversion from one domain back to it is reasonably well. But the conversion from one domain to another fails in semantic meaning, although the major color is the same. The DDIB cannot learn the semantics well.

A.3 Cycle-consistent Diffusion

We also borrowed the idea of CycleGAN[7] and applied it to latent diffusion model. Here is the process: first add noise to the source latent, and denoise it to the target latent. The loss contains both DDPM noise-prediction loss and cycle reconstruction loss. We expected the latent variables simply contain the semantic-level information rather than pixel-level information.
But the training process is unstable, the conversion result fails easily (even we set the source and target to be the same domain, see Figure 11), and the tuning of hyperparameters is hard.

Figure 11: apply cycle-consistency constraint to latent diffusion model. Upper: source images. Lower: converted images.

A.4 More Examples

Here are some photos taken from the campus of Tsinghua University, and they were converted to MC style.

Figure 12: **Photos taken on campus.**