ML3D Final Report: Geometry Free View Synthesis

Mohamed Kotb
Technical University of Munich
mohamed.kotb@tum.de

Leonardo Fernandes Oliveira
Technical University of Munich
leonardo.oliveira@tum.de

Margarita Piscenco
Technical University of Munich
margarita.piscenco@tum.de

Jasmina Stratorska
Technical University of Munich
jasmina.stratorska@tum.de

Abstract

This report explores several modifications implemented based on the Geomtry Free Single View Synthesis paper [5], as part of the Machine Learning for 3D course project. Specifically, we focused on three key modifications: enhancing scene style consistency, integrating text-based influence in synthesis using CLIP, and predicting segmentation maps concurrently with image synthesis. Project repo: https://github.com/piscenco/geometry-free-view-synthesis

1. Motivation

In our project modifications, we categorize them into two main groups. The first involves influencing the synthesis process itself, while the second entails adding an additional output head to predict extra information. In the first category, we experimented with:

- Enhancing model style consistency across consecutive frames by conditioning on a ”Scene code,” a latent vector containing high-level scene style information.
- Utilizing the CLIP model to influence image synthesis through test-time optimization without requiring additional training.

For the second category, we aimed to investigate the feasibility of simultaneously synthesizing segmentation maps alongside image synthesis.

2. Introduction

Geometry-Free View Synthesis is a method for synthesizing single views. It employs a transformer architecture to model the distribution of plausible destination images (x_{dst}) given a source image (x_{src}) and camera transformation T for a new viewpoint. Learning this distribution, as represented by Equation (1), necessitates a model capable of capturing long-range interactions between the source and target views to implicitly represent geometric transformations. Transformer architectures naturally fulfill these requirements as they are not limited to short-range relations like convolutional neural networks (CNNs) with their convolutional kernels, and they exhibit state-of-the-art performance [7]. However, likelihood-based models have been shown [6] to allocate excessive capacity to short-range interactions of pixels when directly modeling images in pixel space. Therefore, inspired by [2], a two-stage training approach is employed. In the first stage, adversarially guided discrete representation learning (VQGAN) is performed, yielding an abstract latent space that has proven to be well-suited for efficiently training generative transformers. In the second stage, a Transformer architecture is trained on the latent space obtained from VQGAN instead of training the transformers directly in image space.

$$x_{\text{dst}} \sim p(x_{\text{dst}}|x_{\text{src}}, T) \tag{1}$$

3. Method

3.1. Synthesizing using CLIP encoding

The idea of this part of the project was to influence the image generation with text prompt. Our approach in order to achieve this consisted of the following steps:

1. The initial image is processed by GeoFree’s VQGAN transformer architecture, which is responsible for generating a synthesized image
2. The output from the model is a synthesized image that is meant to be a preliminary version before being refined according to the text prompt.
3. This synthesized image is then encoded using CLIP, which can understand and encode images and text into a similar space, allowing for comparisons. Also, a target image is also encoded by CLIP. This target image

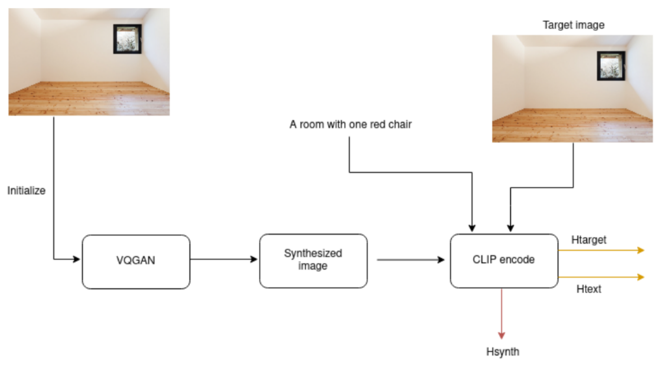

Figure 1. Overview of the optimization process for influencing image generation via text prompts

can be understood as an additional prompt. This is necessary because we want the newly generated scene to be somewhat similar to our target image.

4. After this process, we have three encodings in a latent space: h_{synth}, h_{target} and h_{text}, which correspond to the CLIP encodings for the synthesized image, target image and text prompt respectively.

5. We then optimize the following cosine similarities: (h_{synth}, h_{text}), (h_{synth}, h_{target}). The main idea is to force the synthesized image to have similar CLIP features to both the text and the target image.

Figure 1 illustrates the generation of the CLIP encodings h_{synth}, h_{text} and h_{target}, which are then aligned using cosine similarity during the optimization process.

3.2. Consistent style synthesizing per scene

Our approach to enhance the model's consistency in synthesizing images from the same scene involves introducing additional bias to the synthesis process by conditioning on a latent vector designed to encapsulate global scene information shared across all images synthesized from that scene. We propose a method (refer to Figure 2) to learn these scene codes, inspired by the training approach introduced in the deepSDF paper [4]. We utilize a learned embedding that maps a scene ID to a latent vector and optimize this embedding during the same training loop used for image synthesis. Given that this latent vector is shared by all images of a scene and has a relatively small size (in our case, 1024), it serves as a regularizer during the optimization process, encouraging the encoding of only high-level and global information relevant to synthesizing scene images. This concept is referred to as a ”scene style code.” However, training the transformer using the realestate10K dataset, which contains over 70,000 scenes, required more computational resources than available. Therefore, in the following paragraphs, we outline a proof-of-concept method we employed for such training.

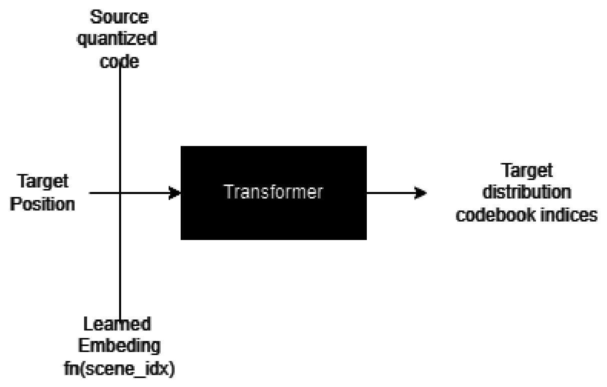
Source quantized code
Target Position
Transformer
Target distribution codebook indices
Learned Embeding fn(scene_idx)

Figure 2. Learning a scene code as a prior.

From a high-level perspective, our objective is to employ a model to tackle a non-trivial task, such as novel view synthesis in our case, and learn a prior that aids in solving this task effectively. In our approach, we replace the original non-trivial task of novel view synthesis with a more manageable yet still non-trivial using a variational autoencoder reconstruction task (refer to Figure 3). The rationale behind this substitution is that if a non-trivial task like reconstruction from a very compressed space can benefit from the learned scene information, then it is plausible that novel view synthesis could also benefit from it.

During inference, we would use a test-time optimization technique to optimize the scene code using all synthesized images. When a new image is synthesized, the scene code is reoptimized to better fit the scene, thereby enhancing the consistency of the synthesized images as we explore a scene.

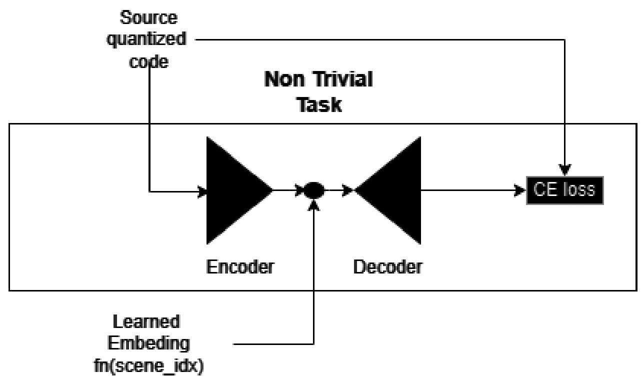

Figure 3. Emulating a non-trivial task with a vae reconstruction task.

3.3. Decoding segmentation maps from latent space

In our next experiment we introduced a modification to the model so that it predicts segmentation map for every generated image. We presumed that the quantized encoding of the original model contains data regarding object classes and their positions and we use this idea to obtain segmentation from the quantized encodings.

Our intention was to use segmentation in order to help in our other modifications, primarily in the consistent synthesizing of scenes. Another reason why we wanted to use segmentation was to have an additional constraint for training, or another task we optimize for in order to get a better encoder and decoder for the image synthesis. It was also our intention to use segmentation to enforce consistency of different views of the same scene.

Our approach for obtaining segmentation maps as an additional output of the model involves creating a branch after the encoder layers and passing the latent code to our segmentation model. The segmentation model consists of four transposed convolutional layer with a batch normalization and ReLU activation after each one, and a final 1x1 convolutional layer which has 21 output channels. The transposed convolutional layers decrease the channels from 256 to 128, then to 64, and finally to 32. A padding and kernel size of two is used for the transposed convolutional layers.

The encoder weights were frozen during training due to two reasons: retraining the whole original model would be technically challenging to do so, and as discussed above, we presume that the quantized encodings contain the information needed for obtaining the segmentation maps.

4. Results

4.1. Synthesizing using CLIP encodings

Referencing the methodology, our results showcase the experiments of combining VQGAN and CLIP encodings to guide image synthesis per textual prompts and a target image. The synthesized image encodings (h_{synth}) were optimized for alignment with the text encodings (h_{text}) and target encodings (h_{target}), with the aim of achieving text-visual coherence. Figure 4 illustrates the obtained result for a selected input.

4.2. Consistent synthesizing per scene

As detailed in the methodology section, we trained a VAE to compress the image code into a highly condensed dimensionality, then attempted to reconstruct the image code while conditioning on the learnable scene code embedding, which is shared by all images within a scene. Figure 5 illustrates the model trained on fewer than 25 scenes, demonstrating the effectiveness of the scene code in aiding the model to solve non-trivial tasks.

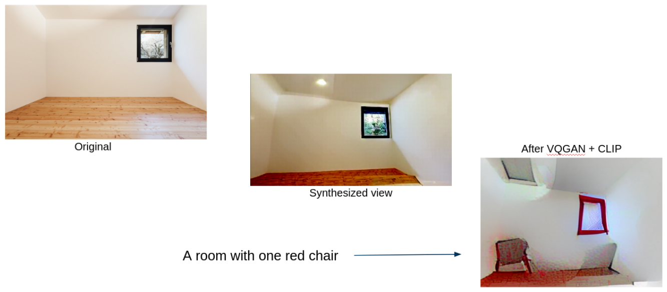

Figure 4. Qualitative result of the training process described in section 3.1.

To assess the approach's ability to construct a latent space representing scenes, we clustered the learned scene codes. Despite training on fewer than 25 scenes, which is insufficient for fully learning the latent space of scenes, we observed common high-level similarities between scenes. Figure 6 displays a sample of images from scenes within the same cluster, showcasing predominantly kitchen scenes.

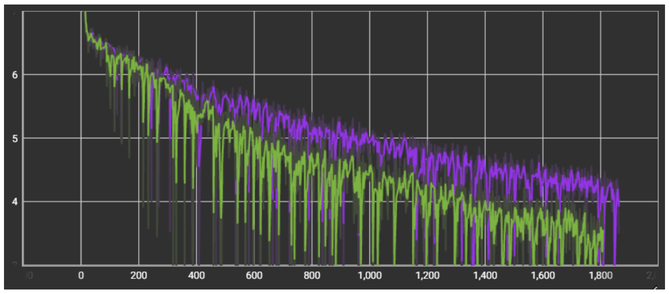

Figure 5. Training with scene code helps the non-trivial task.The y-axis represents cross entropy. Green curve is the model trained with scene codes as prior and purple curve is trained without the prior.

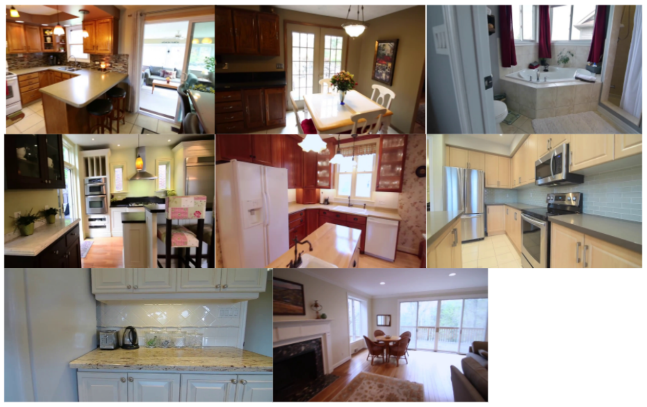

Figure 6. Sample of images of scenes from the same cluster after learning the latent space of scenes. Most of the scenes in the cluster are scenes of kitchens.

4.3. Decoding segmentation maps from latent space

The original datasets the model was trained on contained indoor scenes and did not have ground truth segmentation. Due to this we tried subsets of indoor scenes that are close to the distribution of the original data the model was trained on. Due to the lack of segmentation maps for training in these datasets, we generated the segmentation maps using the current state of the art model for segmentation, DeepLab3+ [1]. Once we trained our segmentation model on these generated segmentation maps, the results we obtained were not good.

Then we used the PASCAL VOC dataset [3] to train the segmentation model and we achieved better results. Since the PASCAL VOC dataset contains only 21 segmentation classes, our model also predicts 21 classes.

Currently our model achieves IOU of 0.70137. We trained the model for 50 epochs. The training loss curve is shown in Fig. 7.

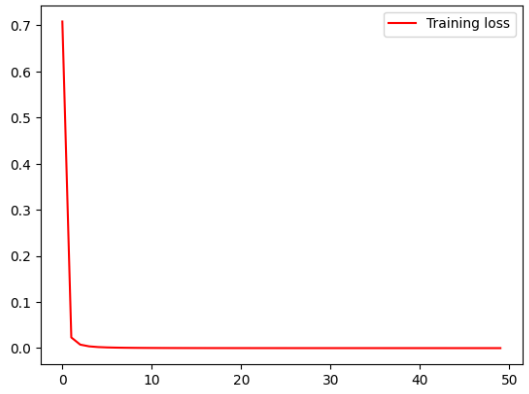

Figure 7. Training loss curve

5. Conclusion

Synthesizing novel views from a single image is an extremely difficult task, demanding a deep comprehension of both the geometry and semantics of 3D scenes. In this project, we explored various approaches to enhance the synthesis process. This included enabling the model to learn scene-specific priors instead of relying on manually hard-coded ones like depth maps. Additionally, we introduced an auxiliary task of predicting segmentation maps alongside RGB images to improve the model's semantic understanding of scenes.

Despite our efforts, the main challenge of this project revolved around the significant resources required to modify a transformer architecture trained on the Realestate10K dataset. While we implemented workarounds to mitigate the need for retraining the model, it remains intriguing to consider how these modifications could benefit the original problem of novel view synthesis if incorporated into the training process of the model introduced by Geometry Free View Synthesis [5].

References

[1] Semantic segmentation on PASCAL VOC 2012 test. https://paperswithcode.com/sota/semantic-segmentation-on-pascal-voc-2012. 4

[2] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis, 2021. 1

[3] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. 4

[4] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation, 2019. 2

[5] Robin Rombach, Patrick Esser, and Björn Ommer. Geometry-free view synthesis: Transformers and no 3d priors, 2021. 1, 4

[6] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications, 2017. 1

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 1