CLIP-guided Editing of Environment Maps using Inversion Techniques

Antonio D'Orazio
Student in Computer Science
Sapienza, University of Rome
dorazio.1967788@studenti.uniroma1.it

Abstract

This project explores the application of inverse problems to enhance environment maps by optimizing images or latent codes to maximize their similarity with the CLIP-embedded textual prompts. An iterative optimization pipeline is introduced to minimize the CLIP loss between the environment map and the prompt, resulting in the generation of novel environment map samples. The work also involves fine-tuning the CLIP model to improve its performance in this specific context. Additionally, I explore the optimization of the latent code of a Variational Autoencoder (VAE) to enhance the generation of unique and high-quality environment maps guided by different CLIP models.

1 Task description/Problem statement

The goal of the project is to experiment with inverse problems as a method to apply modifications to environment maps. The statement for the inverse problem is: "*Given an environment map and an input prompt as inputs, what is the best latent code of the image that, when decoded, the image has the highest similarities with the CLIP embedding of the prompt?*". To answer this question, I propose an iterative optimization pipeline, which computes the CLIP loss between the environment map and the textual prompt. The CLIP loss is defined as follows:

$$\mathcal{L}_{\text{CLIP}} = 1 - sim(\Theta_I, \Theta_T) = 1 - \frac{\Theta_I \cdot \Theta_T}{\|\Theta_I\|\|\Theta_T\|} \tag{1}$$

Where Θ_I, Θ_T represent the CLIP encodings of the image and the prompt, respectively, and sim is the cosine similarity.
By back-propagating on the CLIP loss, the pixel value of the images (or its latent code) is iteratively updated in order to minimize the CLIP loss, with the result of generating unseen samples of new environment maps.

The project is divided into two main subsections

- **CLIP-guided optimization:** The main task. Experiments proved that optimizing on plain pixels is not enough to have meaningful synthesis, this is because the optimizer has no prior knowledge of the environment maps structure, therefore I ran more tests optimizing the latent code of a Variational Autoencoder (VAE). Different CLIP fine-tunings are tested.

- **Fine-tuning of the CLIP model:** the general-purpose CLIP models are not trained specifically for environment maps. Although they are capable of handling generic conditions, such as lights or familiar elements in the image, my fine-tuning experiments aim to verify if the performance of the main task improves when CLIP gains more understanding of the concept of environment maps and their labels.

1.1 Method

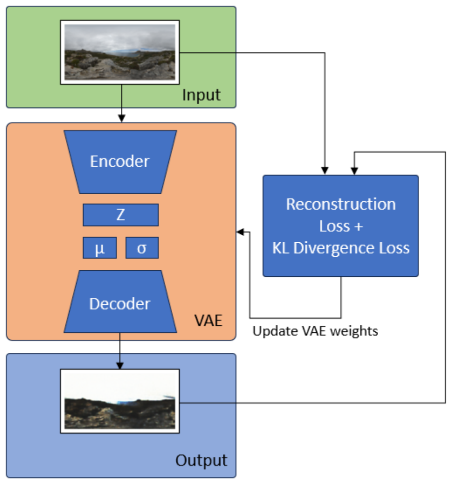


Figure 1: Stage 1: pre-training of the VAE

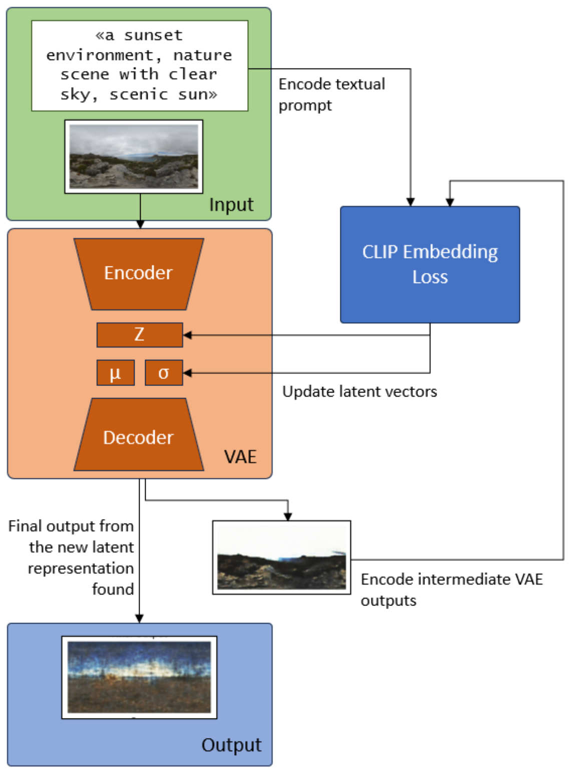

«a sunset environment, nature scene with clear sky, scenic sun»
Input
Encode textual prompt
CLIP Embedding Loss
Encoder
z
μ
σ
Update latent vectors
Decoder
VAE
Final output from the new latent representation found
Encode intermediate VAE outputs
Output

Figure 2: Stage 2: optimization pipeline with the VAE's weights frozen

The model expects as input a pair of [*Starting environment map, Textual prompt*]. The output is a new environment map generated starting from the existing one, following the modifications exposed in the prompt.

Starting with the initial VAE encoding of the input image, we want to locate a latent space vector that minimizes the CLIP loss between the CLIP embeddings of the VAE's output image and the input text. In mathematical notation:

$$\arg\min_{z} \mathcal{L}_{CLIP}(\psi(D(z)), \psi(t)) \tag{2}$$

Where z is the latent code vector, $\mathcal{L}_{CLIP}$ is the CLIP loss defined at (1), ψ is the CLIP encoding model, $D(z)$ is the VAE's decoding of the latent vector, and t is the textual prompt.

If we're optimizing directly for the parameters of the Gaussian distribution encoded by the VAE, we can reformulate the optimization problem as follows:

$$\arg\min_{\mu,\sigma} \mathcal{L}_{CLIP}(\psi(D(\mu, \sigma)), \psi(t)) \tag{3}$$

In this situation, the decoding process begins one step ahead, once the latent code z has already been converted into μ and σ.

1.2 Real-world applications

The availability of free high-quality environment maps is limited to a few collections like Poly-Haven [1]. Environment maps are mostly used "as is", meaning that the artists will not edit them to stylize the environment, but they have to select the most suitable one from a collection of pre-existing images.

The project is meant to be used by computer graphics artist who need proper lighting of their scene by using environment maps. With this approach, artists can perform modifications on existing environment maps generating unseen samples, using textual prompts as input.

2 Related work

- **StyleCLIP** [4]: StyleCLIP can manipulate images by iteratively altering the latent space of a Generative Adversarial Network (GAN) minimizing the CLIP-based loss between the textual prompt and the resulting image.

- **Text2Light** [3]: Text2Light translates the input prompt to a low dynamic range (LDR) panorama using a dual-codebook discrete representation. First, the input text is mapped to the text embedding by the pre-trained CLIP model. Then, a text-conditioned global sampler learns to sample from the global codebook according to the input text. The result is then upscaled to high resolution and converted to high dynamic range (HDR).

3 Datasets and benchmarks

- **PolyHaven** [1]: This dataset is a collection of environment maps publicly available from the website polyhaven.com. It provides different environment maps ordered by categories, such as *[outdoor, natural light, low contrast]*, and tags, for example *[tree, winter, dry, field]*. The environment maps are used to train the variational autoencoder, while the tags and the categories are used to fine-tune CLIP.

4 Existing tools, libraries, papers with code

- **CLIP** (Contrastive Language-Image Pre-training) [5]: a deep learning model that understands and connects images and text. It does so by learning to encode both images and text into a common representation space, where it can measure the similarity between them. This enables CLIP to perform tasks like image classification, text-based image retrieval, and more, by finding the closest match between text and images in this shared space. It's trained using a large dataset of images and text descriptions in a way that encourages it to understand the relationships between different concepts.

- **StyleCLIP** [4]: I took the code of the CLIP-based loss from this project, but then I modified it in order to use the Cosine similarity for a better understanding of the score.
- **CLIP guided loss** [2]: This work provides a standalone library of the CLIP-based loss. During the early stages of this homework, I did some experiments with the code provided in this work. However, the code was meant for older versions of PyTorch so it led to incompatibilities. Therefore, it is not used in my homework project.

5 State-of-the-art evaluation

Given that the work is substantially a novel approach, there are fewer resources to compare it with the state-of-the-art. A similar work like StyleCLIP, although in a different domain, evaluates the result by comparing the average similarity score by fixing the image generator.
I used the Cosine Similarity as a metric, for a more understandable analysis. For the evaluation of the CLIP fine-tuning, I evaluated the cosine similarity between the textual caption and the image from the test set. The final score is the average of all the similarities. The baseline is the default CLIP model is **ViT-B/32**.

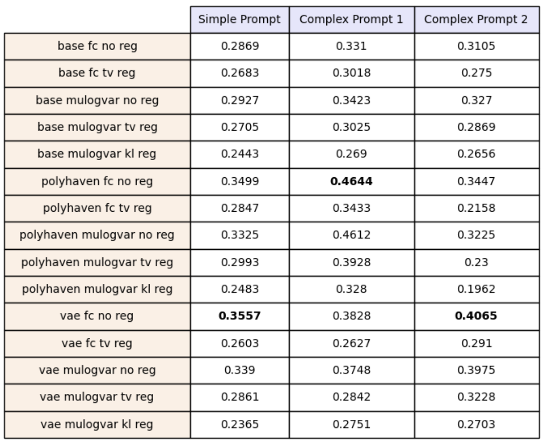
6 Comparative evaluation

6.1 Main Task

The most challenging part of the evaluation is to decouple the performance of the CLIP-based optimization from the quality of the image generator chosen. To do further exploration, I fixed a batch of nine images and tested them with different combinations. The metric is the cosine similarity, and the combination can be interpreted as follows:

- **CLIP Model** *[base, polyhaven, vae]*: Represents the three different CLIP models tested for computing the CLIP-based loss.
  - Base: The original pre-trained CLIP ViT-B/32 model;
  - PolyHaven: Fine-tune of the ViT-B/32 model on the PolyHaven image dataset with the PolyHaven captions;
  - VAE: Fine-tune of the ViT-B/32 model on the VAE transformation of the PolyHaven image dataset, with the PolyHaven captions associated with the input images of the VAE.
- **Variables optimized** *[fc, mulogvar]*: Whether the latent space optimization has been performed on the latent space Z that comes before its transformation (figure 1) into the parameters [mu, logvar] of the Gaussian distribution, or on the parameters themselves.
- **Regularization** *[no reg, tv reg, kl reg]*: whether the optimization has no regularization, the total variation regularizer, or the KL-divergence regularization (for mu-logvar only).
- **Prompts**:
  - Simple Prompt: "*sunset*"
  - Complex Prompt 1: "*A sunset in a forest, cloudy, with trees*"
  - Complex Prompt 2: "*A sunny day at the beach, clear sky, sand terrain*"

The optimization was performed for each combination of the attributes described above. The final output image was then measured against the initial textual prompt to compare the CLIP-based cosine-similarity between them reached at the end of the optimization.
The similarities were averaged for every image in the batch grouped by the approach taken. The following table on the next page shows the final measurements.



	Simple Prompt	Complex Prompt 1	Complex Prompt 2
base fc no reg	0.2869	0.331	0.3105
base fc tv reg	0.2683	0.3018	0.275
base mulogvar no reg	0.2927	0.3423	0.327
base mulogvar tv reg	0.2705	0.3025	0.2869
base mulogvar kl reg	0.2443	0.269	0.2656
polyhaven fc no reg	0.3499	**0.4644**	0.3447
polyhaven fc tv reg	0.2847	0.3433	0.2158
polyhaven mulogvar no reg	0.3325	0.4612	0.3225
polyhaven mulogvar tv reg	0.2993	0.3928	0.23
polyhaven mulogvar kl reg	0.2483	0.328	0.1962
vae fc no reg	**0.3557**	0.3828	**0.4065**
vae fc tv reg	0.2603	0.2627	0.291
vae mulogvar no reg	0.339	0.3748	0.3975
vae mulogvar tv reg	0.2861	0.2842	0.3228
vae mulogvar kl reg	0.2365	0.2751	0.2703

Figure 3: Average final cosine similarities between image and prompt for the randomly sampled mini-batch

6.2 CLIP Finetuning

The fine-tuning of CLIP has been evaluated by computing the average cosine similarity between the environment map and the label across all the samples in the test set. The following chart shows the results achieved.

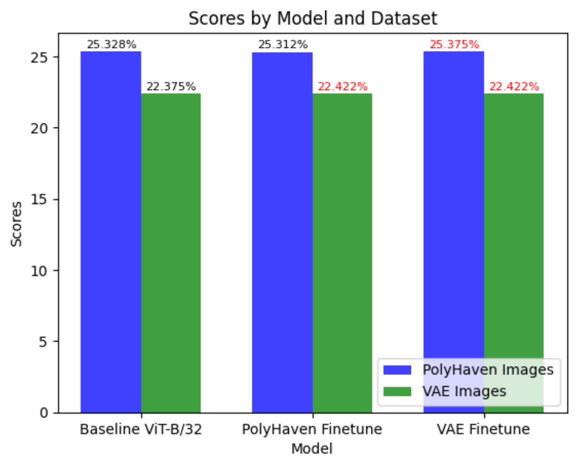


Figure 4: Average cosine similarities between images and captions in the test set

6.3 Results

The following section showcases the most interesting results achieved with the task. For more in-depth results, the notebook *optimization_evaluation.ipynb* contains detailed visualization for the different test cases.

Success cases

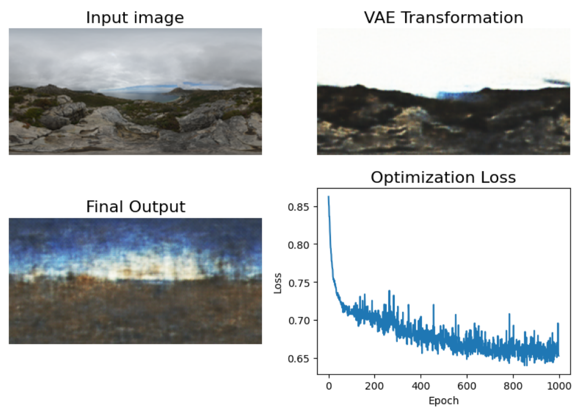


Figure 5: "*a sunset environment, nature scene with clear sky, scenic sun*". CLIP-PolyHaven, Total Variation regularizer, optimized on Z

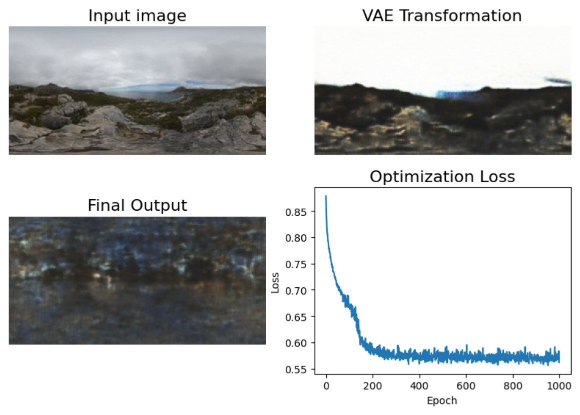


Figure 6: "*dark environment, at night time, with trees in a forest*". CLIP-PolyHaven, Total Variation regularizer, optimized on Z

Failure cases

Optimizing directly in the Mu-Logvar space instead of the Z space results in more failed optimizations that would have otherwise worked in the Z space.

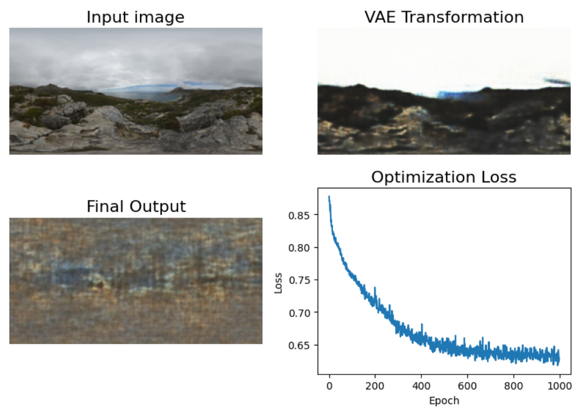


Figure 7: "*dark environment, at night time, with trees in a forest*". CLIP-PolyHaven, Total Variation regularizer, optimized on Mu-Logvar

Obtaining an indoor scene from an outdoor scene is one of the most problematic failure cases. Although we can see that the floor is approaching an indoor color, the result was clearly not desirable.

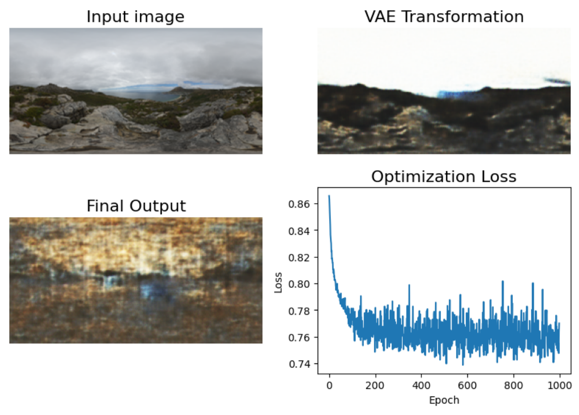


Figure 8: "*an indoor scene of a hangar*". CLIP-PolyHaven, Total Variation regularizer, optimized on Z

When optimizing on plain pixels, the CLIP loss is not able to properly generate a new sample, due to the sparsity of the pixel space and having no knowledge of the image domain, even if the loss was successfully minimized.

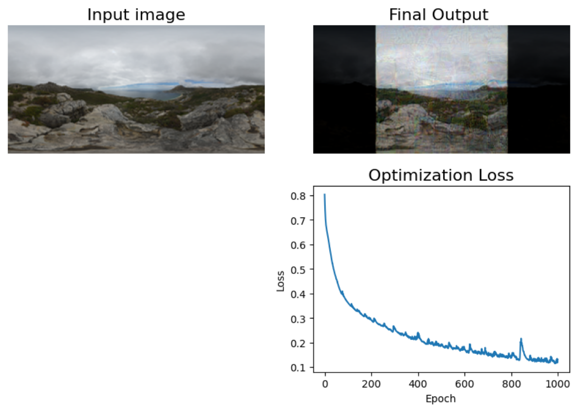


Figure 9: "*a sunset environment, nature scene with clear sky, scenic sun*". CLIP ViT-B/32, plain pixels optimization

6.4 Discussion

Limitations of the project are found in the visual model exploited for the optimization. A small toy example of a Variational Autoencoder isn't enough to realize the proper synthesis of high-quality environment maps. The scarcity of the PolyHaven dataset also contributes to the low performances, although fine-tuning CLIP has helped in finding images more adherent to the prompt. However, further improvement can be reached by using image captioning models to synthesize more complex and natural captions instead of simple keywords derived from the tags being grouped together.

Another interesting conclusion is that a quantitative evaluation based on the similarity reached is not a good index of the performances of the optimization process. In fact, even if the metric didn't improve enough during the evaluation of the CLIP finetune, the results were vastly different, with the CLIP-PolyHaven finetune being able to display more desirable results. A motivation for that phenomenon could be that the CLIP similarities approach the optimization without the underlying knowledge of an environment map. This is clearly visible from the first optimization on plain pixels, which shows the loss function going down smoothly even if the results didn't match the user expectations.

The Total Variation regularizer helped the optimization process to reach a smoother image, while the KL-Divergence regularizer resulted in a damaged image. Further work can be done towards improving this problem, such as switching to diffusion models, quantized variational autoencoders (VQVAE), or generative adversarial networks, like

the work done in StyleCLIP.
With less severe resource constraints, the results can improve and reach more accurate realism.

References

[1] Polyhaven dataset of hdri environment maps. https://polyhaven.com/. 2

[2] Pytorch clip guided loss. https://github.com/bes-dev/pytorch_clip_guided_loss. 3

[3] Zhaoxi Chen, Guangcong Wang, and Ziwei Liu. Text2light: Zero-shot text-driven hdr panorama generation, 2023. 2

[4] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery, 2021. 2, 3

[5] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 2