Continuous Style Control

Jonathan Swinnen
ETH Zürich
jswinnen@ethz.ch

Patrick Eppensteiner
ETH Zürich
eppatric@ethz.ch

Yannick Biehl
ETH Zürich
ybiehl@ethz.ch

Abstract

Generative Adversarial Networks (GANs) such as StyleGAN can be used for style control of the generated images by manipulating the input latent code. Within their continuous latent space latent directions correlating with image characteristics can be found and used for interpolating between styles such as hair color, skin tone for generated face images. In this work, we seek to interpolate continuously between styles using Stable Diffusion models. We explore different methods of interpolation within attention maps, CLIP embeddings and latent space representations, and we found that especially the attention map interpolation looks quite promising. Our implementation of this technique is based on the Prompt-to-Prompt framework and extends it with an interpolation controller. This controller takes any Prompt-to-Prompt edit and interpolates between the base and edited attention maps to generate a new image. Finally, we also interpolate between real images with the help of Null-text-Inversion.

1 Introduction

In recent years diffusion models like Stable Diffusion [Rombach et al., 2022] have shown their incredible capabilities as generative models for image generation. To generate these results diffusion models rely on a user provided text prompt. With techniques like Prompt-to-Prompt [Hertz et al., 2022] generated images can be edited by changing items or the style image while maintaining the overall scene. However these text edits are discrete and often yield a strong and abrupt change in the image. Therefore, in this work we try to offer more fine-grained control over the strength of the change, going as far as offering continuous control over the style of the generated image.

2 Related Work

[Härkönen et al., 2020] have analyzed the latent space of Generative Adversarial Networks (GANs) and were able to identify important latent directions based on Principal Component Analysis (PCA). These directions can be used to do style changes to the image such as change of viewpoint, aging, lighting, and time of day. [Ramesh et al., 2022] have used the contrastive model CLIP with diffusion models for image generation with DALL-E-2. Their two stage model consisting of a prior and a decoder can also be used to produce variations between two image by interpolating their CLIP image embedding and then decoding it with the diffusion model. [Hertz et al., 2022] have observed that the cross-attention layers of text-driven synthesis models primarily influence the mapping between each word of a user given prompts and the spatial layout of the generated image. The authors provide several methods for text-driven image editing such as global editing by adding more words to the prompt, localized editing by replacing a word, and lastly manipulating the weight of a certain word within the prompt for the image generation. Therefore, Prompt-to-Prompt can be used for style control by replacing a word, however these style changes are discrete and Prompt-to-Prompt as such does not implement interpolating continuously between styles. [Mokady et al., 2023] provide a method for text-based modification of real images using only text guidance. Their algorithms first apply DDIM inversion on the input image to yield an initial diffusion trajectory. This trajectory is

then used as a pivot for their Null-text optimization technique, which for each diffusion step optimizes the unconditional textual embedding that is used for classifier-free guidance with a reconstruction MSE loss between the predicated latent code to the pivot. This enables applying Prompt-to-Prompt editing capabilities to real images and also give latent representations of real images, which can then be used to interpolate between them.

[Brack et al., 2023] have used the classifier free guidance mechanism to steer image generation towards or away from an edit prompt by varying the guidance scale.

3 Methods

3.1 Clip interpolation

As a baseline, we first attempt interpolation of CLIP text embeddings. Given two prompts p_1 and p_2, we first get their CLIP embedding $c_1 = \psi(p_1)$ and $c_2 = \psi(p_2)$ by running them through the encoder ψ. Then, we interpolate between these two embeddings to get c_α. We can use either linear or spherical interpolation.

$$c_\alpha = (1-\alpha)c_1 + \alpha c_2 \qquad \text{(linear)}$$
$$c_\alpha = slerp(c_1, c_2, \alpha) \qquad \text{(spehrical)}$$

We use these interpolated embeddings to generate the interpolated images. Results are shown in Figure 1. We can see that this approach on its own does not look suitable for controlling images continuously. There are quite a few large discontinuous jumps, and the start and endpoint look completely different. This is because the generation from p_1 and p_2 are completely independent of each other and nothing constrains them to have a similar structure.

Figure 1: CLIP interpolation between prompts: "A charcoal drawing of a squirrel eating a burger", and "A painting by Van Gogh of a squirrel eating a burger". (top: linear, bottom: spherical)

3.2 Attention interpolation

Using the Prompt-to-Prompt framework, we generate a base image using a prompt p_1, and then generate a target image using an edited prompt p_2. The target image will look structurally similar to the base image, which makes interpolating between them much easier. Our algorithm modifies the Prompt-to-Prompt framework to also generate a third image together with the base and target images. This image is generated using attention maps which are linear interpolations of those of the base and the target image. The interpolated attention maps are injected only for time steps $t < \tau$ to not over-constrain the generation process.

Only interpolating between attention maps is not enough, however. Recall the definition of the attention layers:

$$M = Softmax\left(\frac{QK^T}{\sqrt{d}}\right)$$
$$Att = MV$$

To get the output of an attention layer, the attention maps M are multiplied by the values V. However, for the cross-attention layers, these values V are calculated from the CLIP embedded prompt. Just interpolating the attention maps without also using an interpolated CLIP embedding will not produce

a good result. When we apply both attention map and CLIP interpolation, the results do look very promising, as can be seen in figure Figure 2. A more detailed description of the algorithm can be found in the appendix.

Figure 2: Linear attention & CLIP interpolation between prompts: "A charcoal drawing of a squirrel eating a burger", and "A painting by Van Gogh of a squirrel eating a burger".

3.3 SEGA Interpolation

To compare our method, we also consider a second alternative to continuously control image generation, based on semantic guidance. This technique does not touch the attention maps, but instead modifies the classifier free guidance mechanism to be able to steer image generation towards or away from an edit prompt. The semantic guidance scale, which determines how strongly the generation is steered by an edit prompt, can be changed continuously. This means we can attempt to interpolate this parameter to continuously control the style of a generated image in a certain semantic direction. An advantage of this method over the Prompt-to-Prompt-based method, is that it is possible to handle multiple edit prompts. Each prompt has its own guidance scale which can be controlled independently. This means it's possible to have more control over the interpolation path and apply a sequence of many different edits over multiple steps, while the Prompt-to-Prompt setting only allows us to apply a single edit prompt to interpolate towards. An example of this is shown in Figure 3.

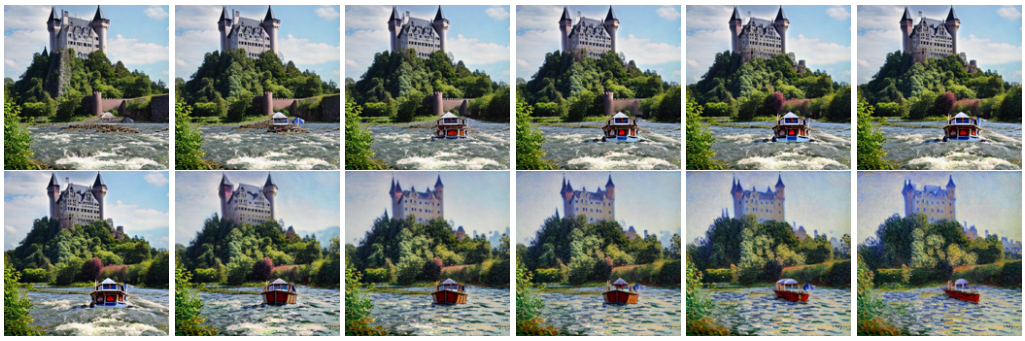

Figure 3: Two steps of SEGA interpolation. Original prompt: "a castle next to a river". We first steer towards "boat on a river, boat" (top), and then to "monet, impression, sunrise" (bottom)

3.4 Latent space interpolation with Null-text Inversion

Another approach for continuously interpolating between style of images is to interpolate between their latent representation. For that, we first apply Null-text Inversion on two given real images to yield their latent representation z_1 and z_2 which are tensors of size 4x64x64. In the next step we interpolate linearly between these two tensors and finally apply the decoder to create the interpolated image $z_{new} = z_1 * \alpha + (1 - \alpha) * z_2$.

Figure 4: Linear interpolation of latents between two real images with prompts "A photo of a red sports car" and "A photo of a yellow sports car" with α: 0.0, 0.25, 0.5, 0.75, 1.0 from left to right

This approach works well for continuous color changes and can also be used to extrapolate with $\alpha < 0.0$ and $\alpha > 1.0$ as shown in Figure 14 and Figure 15 (see appendix).

However this method has problems when morphing two real images with different scenes and objects and creates blurry images during the interpolation, which is shown in Figure 16.

Lastly, the attention interpolation method from Section 3.2 can be applied to interpolate between a real image and a Prompt-to-Prompt edit of this image, as demonstrated in Figure 5.

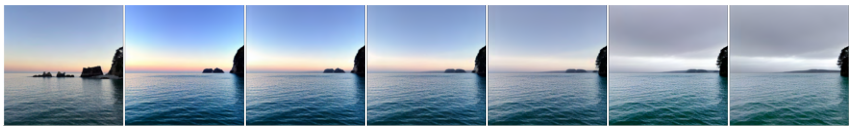

Figure 5: Full attention interpolation of real image with prompt "a photo of the calm sea" on the left and an edited image with prompt "a photo of the stormy sea" on the right

4 Results and Conclusion

Using the aforementioned techniques, mainly the combination of attention map and CLIP interpolation, we can achieve continuous style control. We enable the ability to continuously morph between two generated images, where the images can differ in color, style and even sometimes concepts as challenging as shape, as shown in Figure 6. Compared to results from the alternative SEGA interpolation method, shown in Figure 7, the morphs produced by attention control are more continuous and have less sudden jumps in styles, shapes or other undesirable discontinuities. For more examples, we refer to Figure 8, 9 and 10 in the appendix.

We also succeeded in interpolating between real images using Null-text-Inversion and latent interpolation, and given the compatibility of Null-text-Inversion and Prompt-to-Prompt, it is also possible to apply attention interpolation to real images. In our current implementation the optimized unconditional embeddings from Null-text Inversion are not yet used for the full interpolation leading to less continuous style and scene changes, as seen in Figure 5. It is to be expected that the interpolation would be smoother by taking these unconditional embeddings into account, which is to be examined in future work.

There are still some limitations however. We have no control over spatial features, as this is not possible using Prompt-to-Prompt. Also, if the original image and the target image are too different, continuously morphing might be too challenging for our method and can sometimes produce blurry results. As a result of having to generate three images in parallel, our method also requires a relatively high amount of VRAM (around 13-14 GB). It is also only possible to use a single edit prompt, while SEGA allows multiple edit directions to be controlled independently. Our method could perhaps be extended to allow interpolation between multiple edit prompts, but this would require generating even more images in parallel, increasing the VRAM requirements further.

Figure 6: Attention & CLIP interpolation between "squirrel eating a burger" and "lion eating a burger"

Figure 7: SEGA interpolation with initial prompt "squirrel eating a burger", with edit prompts steering away from "squirrel" and towards "lion"

References

Manuel Brack, Felix Friedrich, Dominik Hintersdorf, Lukas Struppek, Patrick Schramowski, and Kristian Kersting. Sega: Instructing diffusion using semantic dimensions. *arXiv preprint arXiv:2301.12247*, 2023.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. *Advances in Neural Information Processing Systems*, 33:9841–9850, 2020.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022.

Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6038–6047, 2023.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.

5 Appendix

5.1 Attention Interpolation Algorithm

We adopt a notation inspired by the Prompt-to-Prompt paper, where $DM\left(z^{(t)}, c, t, s\right)$ is the computation of a single denoising step t given a CLIP embedding c, which outputs the noisy latent $z^{(t-1)}$ and the attention maps $M^{(t)}$. We use the notation $DM\left(z^{(t)}, c_\alpha, t, s\right)\{M \leftarrow \widehat{M}^{(t)}\}$ for a denoising step where the attention maps are replaced by $\widehat{M}^{(t)}$. The $Edit$ function represents a Prompt-to-Prompt edit as defined in their paper. The local editing option was omitted from the algorithm for simplicity, but the final implementation is also compatible with this option.

Algorithm 1 Attention Map Interpolation using Prompt-to-Prompt

Input: A base prompt p_1, a target prompt p_2, an interpolation alpha $0 \leq \alpha \leq 1$ and a random seed s
Output: A base image x_1, a target image x_2 and an interpolated image x_α

$c_1 \leftarrow \psi(p_1)$
$c_2 \leftarrow \psi(p_2)$
$c_\alpha \leftarrow (1-\alpha)c_1 + \alpha c_2$
$z_1^{(T)} \sim \mathcal{N}(0,\, I)$
$z_2^{(T)} \leftarrow z_1^{(t)}$
$z_\alpha^{(T)} \leftarrow z_1^{(t)}$
for $t = T, T-1, ..., 1$ **do**
  $z_1^{(t-1)}, M_1^{(t)} \leftarrow DM\left(z_1^{(t)}, c_1, t, s\right)$
  $M_2^{(t)} \leftarrow DM\left(z_2^{(t)}, c_2, t, s\right)$
  $\widehat{M}_2^{(t)} \leftarrow Edit\left(M_1^{(t)}, M_2^{(t)}, t\right)$
  $z_2^{(t-1)} \leftarrow DM\left(z_2^{(t)}, c_2, t, s\right)\left\{M \leftarrow \widehat{M}_2^{(t)}\right\}$
  if $t < \tau$ **then**
    $M_\alpha^{(t)} \leftarrow (1-\alpha)M_1^{(t)} + \alpha\widehat{M}_2^{(t)}$
    $z_\alpha^{(t-1)} \leftarrow DM\left(z_\alpha^{(t)}, c_\alpha, t, s\right)\left\{M \leftarrow M_\alpha^{(t)}\right\}$
  else
    $z_\alpha^{(t-1)} \leftarrow DM\left(z_\alpha^{(t)}, c_\alpha, t, s\right)$
  end if
end for
$x_1 \leftarrow DecodeLatent\left(z_1^{(0)}\right)$
$x_2 \leftarrow DecodeLatent\left(z_2^{(0)}\right)$
$x_\alpha \leftarrow DecodeLatent\left(z_\alpha^{(0)}\right)$
return x_1, x_2, x_α

5.2 More Results

Figure 8: Attention & CLIP interpolation between "A realistic painting of a white boy" and "A realistic painting of a black girl"

Figure 9: Attention & CLIP interpolation between "A portrait of a Roman Emperor" and "A portrait of a Chinese Empress"

Figure 10: Attention & CLIP interpolation between "A children's painting of a large castle on a hill in the distance" and "A drawing drawing of a large fantasy castle on a hill in the distance"

5.3 Experiment: "Blurpolation"

We experimented with an alternative method of interpolating the attention maps, instead of using simple linear interpolation. One such method we tried is described in Algorithm 2.

Figure 11 shows an interpolation between two MNIST digits [Deng, 2012] using this method. To use it for attention interpolation, we replace the linear interpolation on line 13 of Algorithm 1 with this function. Note that this method is very experimental, and has no guarantees of always looking good. In fact, in many cases it does not look as good as linear interpolation, but we did find several cases where it did work better. Two examples of this are shown in Figure 12 and 13.

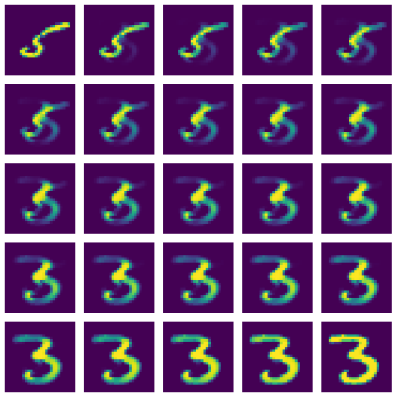

Figure 11: "Blurpolation" between two MNIST digits with parameters: $f = 1$, $\sigma = 21.6$, $k = 5$, $p = 20$ and $q = 0.67$

Figure 12: Attention blurpolation (top) vs linear interpolation (bottom) between prompts "A picture of a lion lying in the grass" and "A picture of a tiger lying in the grass".
In this case, linear interpolation causes a quick transition from lion to tiger at 50%, while blurpolation seems to transition more gradually between the lion and tiger's textures

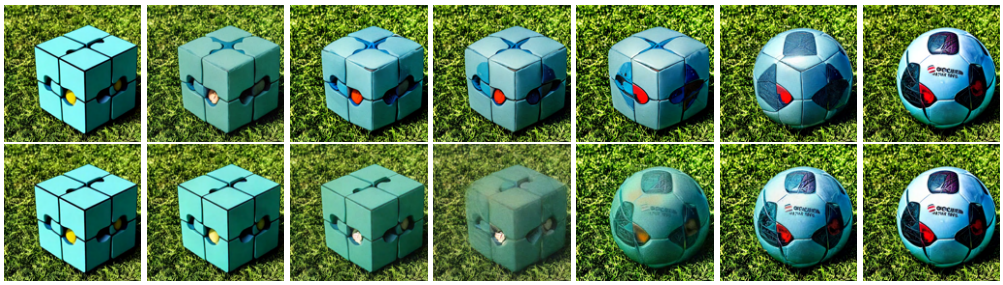

Figure 13: Attention blurpolation (top) vs linear interpolation (bottom) between prompts "A puzzle cube in the grass" and "A soccer ball in the grass"
In this case, linear interpolation produces blurrier results when compared to blurpolation.

Algorithm 2 "Blurpolation"

Input: Two attention maps M_1 and M_2 of equal dimensions, the interpolation α, the Gaussian blur kernel size k and σ, a downsampling factor f and two parameters affecting interpolation acceleration p and q.

Output: An interpolated attention map M

$\widehat{M_1} \leftarrow Downsample(M_1, f)$
$\widehat{M_2} \leftarrow Downsample(M_2, f)$
$B_1 \leftarrow GaussianBlur(Abs(\widehat{M_1})/Max(Abs(\widehat{M_1})), k, \sigma)$
$B_2 \leftarrow GaussianBlur(Abs(\widehat{M_2})/Max(Abs(\widehat{M_2})), k, \sigma)$
$B \leftarrow \frac{B_1+B_2}{2}$
$S \leftarrow Sign(\widehat{M_2} - \widehat{M_1})$
$C \leftarrow (2Pow(B, q) - 1) \odot pS$
$\widehat{A} \leftarrow (Pow(\alpha C, e) - 1) \oslash (Pow(C, e) - 1)$
$A \leftarrow Upsample(\widehat{A}, f)$
$M' \leftarrow (1 + A) \odot M_1 + A \odot M_2$
$M \leftarrow \frac{(1-\alpha)\|M_1\|_2+\alpha\|M_2\|_2}{\|M'\|_2} M'$
return M

(Functions $Pow, Abs, Sign$ are applied elementwise, Max returns the largest element, and the notations $\odot$ and $\oslash$ are used for elementwise multiplication and division, respectively)

5.4 More Latent Interpolation Results

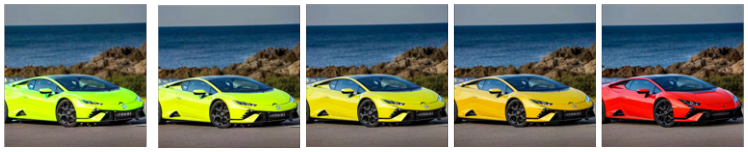

Figure 14: Linear interpolation of image latents for α values: -1.0, -0.5, -0.25, 0.0, 1.0 from left to right

Figure 15: Linear interpolation of image latents for α values: 1.0, 1.25, 1.5, 2.0, 3.0 from left to right

Figure 16: Linear interpolation between two real images with α values: 0.0, 0.25, 0.5, 0.75, 1.0 producing blurry images