Text-driven Video Manipulation with GPT4V

Tom James, John Feng, Willion Liang

1. Introduction

Generative Adversarial Networks (GANs) [11] have revolutionized image synthesis including faces and scenes, with recent style-based generative models [4, 8, 13, 14, 29]. Traditionally, researchers used the Gaussian distribution as the input latent code and then generate images. Recently, GAN inversion techniques are utilized to retrieval the latent code by the pretrained model from the real images. After that, the GAN generator can reconstruct the image using the estimated latent code, which provides challenges and opportunities for the image manipulation task. In our project, we work on semantic manipulation meaning changing the semantic attributes of an object.

Recent work [1, 2, 9, 12, 14] has two directions including optimizing the latent code and predicting the latent code via an image encoder directly. For image-level editing applications several approaches [1, 13] find specific semantic directions in the latent space, e.g., changing poses, colors, or age, while others [15] aim to change the global style, e.g., photo → sketch, photo → cartoon. More recently, StyleCLIP [29] was proposed to edit the image simply by words or phrases, like “angry”, “sad” and “getting older”. It takes advantage of CLIP [30], a vision and language model which was trained on 400 million real-world text-image pairs by using contrastive learning algorithm. With these image GAN inversion-based semantic editing approaches, how can we extend them to videos?

One straightforward method is to use state-of-art GAN inversion techniques and semantic editing like StyleCLIP to manipulate each frame in the video and then combine them to produce the video. However, the edited frames from a same video may not be consistent due to the lack of temporal constraint.

In this project, we start from the existing GAN inversion approaches to obtain the latent code frame by frame. Then we apply semantic editing like StyleCLIP to edit the latent code for each frame. To deal with the limitation of lack of temporal constraint, we apply hierarchical pair-based optimization by minimizing the bi-directional photometric loss and the local generator regularization. In our experiments, we show that our new optimization approach helps achieve significantly improved temporal consistency while maintaining the editing style than baseline methods.

2. Related Work

Multimodal learning [19, 29, 30, 36] is the task of using machine learning algorithms to learn knowledge from different modalities, which received the huge attention these days. Language and vision is one of the most important directions for many tasks including image captioning [22–24], video captioning [10], language-based image or video retrieval [5], representation learning [19] and text-guide image generation [20, 25, 30].

Training representations from scratch cost the large amount of time and computing resource. Learning with the pretrained language or vision models become the new trend. Following the better performance of transformer architecture [35] in both vision and language tasks, a transformer-based pretrained language model, BERT [7] achieves success in various language tasks. As for the image representation learning, some popular CNN based pretrained models contains ResNet, VGGNet and AlexNet [32, 33]. While the Transformer architecture has become the standard for natural language processing tasks, its applications to computer vision remain limited. ViT [8, 21] utilizes vision transformer to encode the image and attains excellent results compared to state-of-the-art convolutional networks while requiring substantially fewer computational resources to train.

More recently, a novel vision and language pretrained model called CLIP [30], which was trained on 400 million real-world text-image pairs by using contrastive learning algorithm. The semantic similarity representation learning by CLIP could be used for zero-shot applications and instructions, especially for the language-guide image editing or construction [13, 29]. Different from the aforementioned work StyleCLIP [29] combines the high-quality images generated by StyleGAN [13], with the rich multi-domain semantics learned by CLIP, our task is to edit videos frame by frame with the help of latent codes for images by the GAN inversion and the extra temporal photometric loss.

3. Approach

3.1. Overview

Direct editing. Given an input video \( V_{input} = \{I_1, \cdots, I_T\} \) of \( T \) frames, our goal is to semantically edit
all the video frames while preserving the temporal coherence of the final edited video. To edit the input video $V_{input}$, we first align its frames and use existing GAN inversion techniques (e.g., [3,31]) to invert the frames $\{I_1^{inv}, I_2^{inv}, \ldots, I_T^{inv}\}$ and their corresponding latent code vectors in the $W$-space of StyleGAN $\{W_1^{inv}, W_2^{inv}, \ldots, W_T^{inv}\}$. We then independently perform semantic editing on these inverted frames to obtain $\{I_1^{edit}, I_2^{edit}, \ldots, I_T^{edit}\}$. To achieve temporal coherence, we hierarchically sample two frames $I_i^{edit}$ and $I_j^{edit}$ from $V_{edit}$ at a time. For each sampled frame pair $I_i^{edit}$ and $I_j^{edit}$, we compute the forward and backward flows $F_{i\rightarrow j}$ and $F_{j\rightarrow i}$ using RAFT [34]. We then use these two flow fields to compute the visibility masks by performing a forward-backward and backward-forward flow consistency error check. We also use the flows to warp the directly edited frames (i.e. $warp(I_i^{edit}, F_{j\rightarrow i})$ and $warp(I_j^{edit}, F_{i\rightarrow j})$). We finally compute the photometric loss (Eqn. 4) for all possible pairs $i$ and $j$.

**Direct editing on a video.** When applying StyleCLIP mapper [29] to a video independently for each frame, we obtain an edited video $V_{edit} = \{I_1^{edit}, I_2^{edit}, \ldots, I_T^{edit}\}$. For each directly edited frame $I_t^{edit}$, there is a corresponding latent code $W_t^{edit}$ such that $I_t^{edit} = G(W_t^{edit}, \theta^{edit})$. Due to the per-frame, independent process, the edited video $V_{edit}$ often suffers from temporal inconsistency.

**Overview of our approach.** We propose a flow-based approach to enforce the edited video to be temporally consistent to address this issue. Our goal is to ensure that the edited video remains temporally consistent while preserving the details from the direct editing. This is inherently a trade-off. On the one hand, preserving all the details from semantic editing leads to temporal flickering artifacts. On the other hand, minimizing the flow-based photometric loss leads to blurry frames (because any blurry frames induce low photometric losses). To address this, we present a test-time optimization approach to balance these two conflicting objectives. Figure 1 shows an overview of our approach. Our core idea is to update the latent code so that the generated frames are temporally consistent (minimizing photometric errors across frames). To do so, we use a flow network to compute the dense bi-directional dense flow map and use them to warp the frames. As all the computations, including the GAN generator, flow estimation network, spatial warping, and photometric losses, are differentiable, we can backpropagate the errors all the way back to optimize the latent code. In the following, we describe the detailed procedure and the losses for our approach.

**3.2. Flow-based temporal consistency.**

We propose a flow-based approach to enforce the directly edited video $V_{edit}$ to be temporally consistent.

**Hierarchical pair-based optimization.** As we cannot fit an entire video into the GPU memory, we choose to optimize the latent code from *pair of frames* at a time. At each iteration, we sample two latent code, $W_t^{edit}$ and $W_j^{edit}$, corresponding to two frames from $V_{edit}$, $I_t^{edit}$ and $I_j^{edit}$. An
intuitive way is to use all possible pair combinations in a video, but this brings high computation costs. Instead, we use a hierarchical sampling strategy, similar to [16, 28], to reduce the computational costs. The sampled pairs include short-term and long-term pairs:

\[ P = \{(I_i^{edit}, I_j^{edit}) \big| |i-j| = k, k = 1, 2, 4, 8, \ldots\} \tag{1} \]

**Flow-based consistency.** Two sampled frames \(I_i^{edit}\) and \(I_j^{edit}\) are used to compute the forward and backward flows \(F_{i\rightarrow j}\) and \(F_{j\rightarrow i}\) using RAFT [34]. We then use these two flows to warp the directly edited frames (i.e., \(\text{warp}(I_i^{edit}, F_{i\rightarrow j})\) and \(\text{warp}(I_j^{edit}, F_{i\rightarrow j})\)), and to compute the visibility masks \(M_{j\rightarrow i}\) and \(M_{i\rightarrow j}\), Visibility mask \(M \in [0, 1]\) is used to penalize the occluded parts. It shows lower weights for occluded pixels and higher weights for the non-occluded pixels. (Shown in Fig 1). To compute the visibility masks, we first compute forward-backward, and forward-backward flow consistency error maps \(\epsilon_{i\rightarrow j}\) and \(\epsilon_{j\rightarrow i}\). The error map is computed by \(\epsilon_{i\rightarrow j}(p) = \|p - F_{j\rightarrow i}(p + F_{i\rightarrow j}(p))\|_2\), where \(p\) is a pixel in the flow field. The error map reveals the occlusion degree. Then these resultant error maps are mapped to \([0, 1]\) using an exponential function such that \(M_{j\rightarrow i} = \exp(-\epsilon_{j\rightarrow i})\) and \(M_{i\rightarrow j} = \exp(-\epsilon_{i\rightarrow j})\). Finally, we compute the loss based on the warped frames \(\text{warp}(I_i^{edit}, F_{i\rightarrow j}), \text{warp}(I_j^{edit}, F_{j\rightarrow i})\) with the visibility masks.

### 3.3. Two-stage optimization strategy

We split our optimization into two stages. In the first stage, only the latent codes \(\{W_t^{edit}\}\) are updated. While in the second stage, we update generator weights \(\theta^{edit}\).

**Motivation.** Our motivation for using the two-stage optimization is that we observe that changing \(\lambda\) with a small learning rate reflects a local appearance change. On the other hand, updating the generator introduces a global change, e.g., the style. We thus split the optimization into two stages to pay special attention when it comes to different editing types.

For the first stage (latent code update), our goal is to update the latent code to minimize the photometric loss:

\[
\arg\min_{\{W_t^{edit}\}} L_{(W_t^{edit})} = L_{\text{photo}} \tag{2}
\]

For the second stage (generator update), our goal is to fine-tune the generator to minimize:

\[
\arg\min_{\theta^{edit}} L_G = \lambda_r L_r + L_{\text{photo}} \tag{3}
\]

where \(L_{\text{photo}}\) again is the bi-directional photometric loss, \(L_r\) is the regularization loss, and \(\lambda_r\) is the strength of regularization. Below we present the detailed loss formulation.

### 3.4. Loss functions

**Bi-directional photometric loss.** We minimize the bi-directional photometric loss for both stages to achieve a temporally consistent video. This loss measures the difference between two frames. We use it with the visibility masks to calculate the deviation in the non-occluded parts.

\[
L_{\text{photo}} = \sum_{I_i^{edit}} M_{i\rightarrow j}||I_j^{edit} - \text{warp}(I_i^{edit}, F_{j\rightarrow i})||_1 + M_{j\rightarrow i}||I_i^{edit} - \text{warp}(I_j^{edit}, F_{i\rightarrow j})||_1 \tag{4}
\]

Intuitively, bi-directional photometric loss ensures colors along the valid (forward-backward or backward-forward consistent) vectors across frames are as similar as possible. The use of a visibility map helps alleviate the negative impact from occlusion/disocclusion.

**Local generator regularization.** In the second stage (generator update), we do not wish to ruin the pretrained latent space. Therefore, we introduce a regularization loss to help prevent generator \(G\) from losing its latent space editability. Similar to [31], we use a local regularization to preserve the editing ability of our generator. More specifically, we first obtain a latent code \(W_t\) by linearly interpolating between the current latent code \(W_t^{edit}\) and a randomly sampled code \(W\) with an interpolation parameter \(\alpha_{\text{interp}}\). This gives us a new latent code in a local region around \(W_t^{edit}\).

To ensure that we do not lose the editing capability of the original generator, we add penalty on the distance between the generated image from new generator and old one as:

\[
L_r = L_{\text{L1PS}}(x_r, x'_r) + \lambda_{\ell_2} L_{\ell_2}(x_r, x'_r) \tag{5}
\]

where \(x_r = G(W_t; \theta^{edit}), x'_r = G(W_t; \theta^{out}), \lambda_{\ell_2}\) is the weight for \(\ell_2\) loss. This regularization can alleviate the side effect from updating \(G\), in a local area. Since for a video, the latent codes for the same identity tend to gather locally.

**Overall loss function.** Our final loss is

\[
\theta^{out}, \{W_t^{out}\} = \arg\min_{\theta^{out}, \{W_t^{edit}\}} \lambda_r L_r + L_{\text{photo}} \tag{6}
\]

where \(\theta^{out}\) is the output parameters of our output generator, and \(\{W_t^{out}\}\) are the latent codes of the output video. We use \(\lambda_{\ell_2} = 1.0, \lambda_r = 200.0, \alpha_{\text{interp}} = 30.0\) in all of our experiments.

### 4. Experiments

#### 4.1. Setup

We test our approach on two sub-sampled datasets, RAVDESS [27] and Voxceleb2 [6]. We subsampled 10 videos from RAVDESS and 40 videos from Voxceleb2, re-
Figure 2. Visual results on RA VDESS [27] and Voxceleb2 [6]. Our results maintain consistent changes with time preserving the temporal coherence.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>RA VDESS</th>
<th>Voxceleb2</th>
<th>RA VDESS</th>
<th>Voxceleb2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metrics</td>
<td>$E_{warp}$</td>
<td>LPIPS$\downarrow$</td>
<td>$E_{warp}$</td>
<td>LPIPS$\downarrow$</td>
</tr>
<tr>
<td>Direct editing</td>
<td>0.0198</td>
<td>0.0204</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>angry</td>
<td>0.0027</td>
<td>0.0058</td>
<td>0.2487</td>
<td>0.1248</td>
</tr>
<tr>
<td>beard</td>
<td>0.0051</td>
<td>0.0049</td>
<td>0.2914</td>
<td>0.1523</td>
</tr>
<tr>
<td>eyeglasses</td>
<td>0.0042</td>
<td>0.0064</td>
<td>0.1241</td>
<td>0.1742</td>
</tr>
<tr>
<td>Depp</td>
<td>0.0023</td>
<td>0.0052</td>
<td>0.2521</td>
<td>0.1602</td>
</tr>
<tr>
<td>surprised</td>
<td>0.0042</td>
<td>0.0059</td>
<td>0.2118</td>
<td>0.1461</td>
</tr>
<tr>
<td>Average performance</td>
<td>0.0033</td>
<td>0.0048</td>
<td>0.2236</td>
<td>0.1515</td>
</tr>
</tbody>
</table>

For the editing directions, we use StyleCLIP [29] and train five mappers, “eyeglasses”, “angry”, “surprised”, “Depp” and “beard”.

We first use Restyle encoder [3] as the GAN inversion method frame by frame, then we apply the directions learned by StyleCLIP frame by frame. We denote this frame-by-frame editing as “direct editing” results. Finally, we apply our approach to the direct editing results.

We evaluate the results with two metrics: 1) temporal consistency and 2) perceptual similarity with the semantically edited frames. To evaluate temporal consistency, we measure the Warping Error $E_{warp}$:

$$E_{warp}(I_t, I_{t+1}) = \frac{1}{\sum_{i=1}^{N} M_t(p_i)} \sum_{i=1}^{N} M_t(p_i) \| I_t(p_i) - \hat{I}_{t+1}(p_i) \|^2_2,$$  

(7)
where \( \hat{I}_{t+1} = warp(I_{t+1}, F_{t-t+1}) \), \( N \) is the number of pixels, \( p_i \) is the \( i \)-th pixel, and \( M_t \) is a binary non-occlusion mask, which shows non-occluded pixels. We compute the forward-backward consistency error and use the threshold in [17, 26] to get \( M_t \).

We also measure the perceptual similarity score [37] between the directly edited video \( V_{{\text{edit}}} = \{I_1^{{\text{edit}}}, I_2^{{\text{edit}}}, \ldots, I_{T}^{{\text{edit}}}\} \) and our output video \( V_{{\text{out}}} = \{I_1^{{\text{out}}}, I_2^{{\text{out}}}, \ldots, I_{T}^{{\text{out}}}\} \) by measuring the averaged perceptual similarity between the corresponding frames:

\[
LPIPS = \frac{1}{T} \sum_{t=1}^{T} LPIPS(I_t^{{\text{edit}}}, I_t^{{\text{out}}}), \quad (8)
\]

### 4.2. Results on RAVDESS

**Setup.** We perform our two-stage optimization approach on the directly edited video using Adam optimizer [15]. For the second stage, we set the learning rate of \( G \) to \( \alpha_G = 5 \times 10^{-6} \), and finetune \( G \) for ten epochs. We set the regularization weight \( \lambda_r \) to 200.

**Analysis.** Table 1 shows that our approach improves the temporal consistency over the directly edited video baseline by a large margin. The main challenging source of inconsistency comes from the details of the newly added attributes, e.g., glasses, and some background flickering. We show sample visual results in Figure 2, where the introduced changes are consistent among the different frames.

### 4.3. Results on VoxCeleb2

**Setup.** We perform our two-stage optimization approach on the directly edited video using Adam optimizer [15]. For the first stage of training, we set the learning rate of the latent codes to \( \alpha_w = 5 \times 10^{-5} \), and update the latent codes for 20 epochs. For the second stage, we set the learning rate of \( G \) to \( \alpha_G = 5 \times 10^{-6} \), and finetune \( G \) for ten epochs. We set the regularization weight \( \lambda_r \) to 200.

**Analysis.** We observe that our approach does not perform well on Voxceleb2 in Figure 2. This could come from two reasons: a) low resolution of the videos, and b) complicated background. Low-resolution brings worse results for the GAN inversion, while more complicated background brings more redundant pixels for the flow estimation and the optimization.

### 4.4. Qualitative comparison with DVP

We also compare our results with a blind video consistency approach, DVP [18] visually. The result is shown in Figure 3. We observe that DVP usually sacrifice the details to achieve a temporal consistency.

### 5. Conclusion

We have presented a novel method for video-based semantic editing by leveraging image-based GAN inversion and editing frame by frame. To deal with the challenges of the temporal constraint, we apply hierarchical pair-based optimization by minimizing the bi-directional photometric loss and the local generator regularization. From our experiments, we show that our method can achieve temporal consistency and preserve its similarity to the direct editing results.

### References


[26] Liang Liu, Jiangning Zhang, Ruifei He, Yong Liu, Yabiao Wang, Ying Tai, Donghao Luo, Chengjie Wang, Jilin Li, and Feiyue Huang. Learning by analogy: Reliable supervision from transformations for unsupervised optical flow estimation. In IEEE Conference on Computer Vision and Pattern Recognition(CVPR), 2020. 5