DaCapo: Score Distillation as Stacked Bridge for Fast and High-quality 3D Editing

Supplementary Material

8. Rationale behind *DaCapo*

Previous SDS-based methods suffer from lengthy optimization with slow inference and degenerated results. We attribute this limitation to their internal conflicts from source preservation versus editing or potential editing directions that may interfere with each other. Inspired by the Mask operation in 2D editing, which we interpret as spatial separation between preservation and edit, we propose to seek a temporary separation between them for 3D scene editing, as the multi-view masks are expensive to obtain and update when the 3D scene gets updated in editing process. Given the semantic similarities between source and target scenes, we also want to seek a short editing path directly bridging from the source and target. Since the similarities mean they are relatively *close* in data manifold and direct bridging could be a short editing path. This short path would contribute to the speed-up of editing process. Next we will show how to achieve the two goals with stacked bridge.

8.1. Motivation

Inspired by the Mask operation in 2D editing, we propose separating preservation and editing temporarily for 3D scenes, as multi-view masks are costly to obtain and update. Given the semantic similarities between source and target scenes, we aim to find a short editing path directly connecting them. This short path, due to their proximity in the data manifold, helps speed up the editing process. Next, we explain how the stacked bridge achieves these goals.

The nature of temporary (spatial) separation is to assign different tasks to different time stages (regions). Diffusion sampling naturally does this: early high-noise stages focus on image contours, while later low-noise stages add fine details. **This inspired us to mimic diffusion sampling**, especially since SDS-based methods already add and remove noise at each step. We transform 3D editing optimization into a diffusion-like sampling process with coarse-to-fine stages. Early high-noise stages prioritize preservation to shape structure and guide editing, while later low-noise stages add texture details. We start with sampling-version DDS, i.e, DDS with diffusion time schedule or decreasing time schedule.

8.2. Stacked Bridge as approximation of DDIB

We further analyzed the theoretical properties of the sampling-version DDS and found that it can be seen as an approximation of DDIB. Unlike DDIB, where inversion and editing occur sequentially, in the sampling-version DDS, they happen simultaneously (i.e., using the score difference instead of treating the two scores as separate update directions). Thus, we refer to it as the Stacked Bridge.

Since the sampling-version DDS retains the advantage of optimization-based methods for directly updating 3D parameters and the stacked form offers a significantly shorter path compared to the concatenated bridge, we chose to approximate DDIB using the Stacked Bridge. Instead of directly extending DDIB to multi-view rendering (which we include as a baseline), Stacked Bridge proves more efficient.

8.2.1. Approximation Error

However, the simple Stacked Bridge framework instantiated with the sampling-version DDS fails to approximate DDIB effectively. We analyze three reasons for this in the main text and propose targeted enhancements: Source-Target Coupling, improved score estimation, and Manifold Correction Gradient. Sampling-version DDS with the three enhancements becomes our proposed *DaCapo* framework, which is a better implementation of the Stacked Bridge framework. *DaCapo* provides a better estimation of DDIB and thus serves as a direct transport path from source to target 3D parameters.

9. More details of *DaCapo*

We summary our Algorithm in Alg. 1.

9.1. Classifier Guidance in *DaCapo*

Recall that the overall editing ODE of *DaCapo* is:

$$\mathrm{d}\boldsymbol{x} = \begin{bmatrix} w_{edit}(t)\left(\boldsymbol{\epsilon}_\phi^{tgt} - \boldsymbol{\epsilon}_\phi^{src}\right) + w_{align}(t)\left(\mathbf{x}_t^{tgt} - \mathbf{x}_t^{src}\right) \\ + w_{correct}(t)\left(\epsilon_{uncond}^{tgt} - \epsilon_{uncond}^{src}\right) \end{bmatrix} \mathrm{d}t. \tag{19}$$

We can simplify the equation above as $\mathrm{d}\boldsymbol{x} = v_\phi(t, \boldsymbol{x}_t)\mathrm{d}t$. The score estimations $\boldsymbol{\epsilon}_\phi^{tgt}$ and $\boldsymbol{\epsilon}_\phi^{src}$ are the implicit classifier score terms in Classifier-Free Guidance (CFG) [13], including the image guidance and text guidance as in Intruct-Pix2Pix [1]. The implicit classifier scores are calculated as follows:

$$\begin{aligned} \boldsymbol{\epsilon}_\phi^{tgt} &= \omega_{\mathrm{y}}(\boldsymbol{\epsilon}_\phi(\boldsymbol{x}_t^{\mathrm{tgt}}, y^{\mathrm{tgt}}, \boldsymbol{x}^{\mathrm{src}}, t) - \boldsymbol{\epsilon}_\phi(\boldsymbol{x}_t^{\mathrm{tgt}}, y_\varnothing, \boldsymbol{x}^{\mathrm{src}}, t)) \\ &\quad + \omega_{\mathrm{I}}(\boldsymbol{\epsilon}_\phi(\boldsymbol{x}_t^{\mathrm{tgt}}, y_\varnothing, \boldsymbol{x}^{\mathrm{src}}, t) - \boldsymbol{\epsilon}_\phi(\boldsymbol{x}_t^{\mathrm{tgt}}, y_\varnothing, \boldsymbol{x}_\varnothing, t)), \\ \boldsymbol{\epsilon}_\phi^{src} &= \omega_{\mathrm{y}}(\boldsymbol{\epsilon}_\phi(\boldsymbol{x}_t^{\mathrm{src}}, y^{\mathrm{src}}, \boldsymbol{x}^{\mathrm{src}}, t) - \boldsymbol{\epsilon}_\phi(\boldsymbol{x}_t^{\mathrm{src}}, y_\varnothing, \boldsymbol{x}^{\mathrm{src}}, t)) \\ &\quad + \omega_{\mathrm{I}}(\boldsymbol{\epsilon}_\phi(\boldsymbol{x}_t^{\mathrm{src}}, y_\varnothing, \boldsymbol{x}^{\mathrm{src}}, t) - \boldsymbol{\epsilon}_\phi(\boldsymbol{x}_t^{\mathrm{src}}, y_\varnothing, \boldsymbol{x}_\varnothing, t)) \end{aligned} \tag{20}$$

where $\omega_I = 1.5$ and $\omega_y = 7.5$ are image and text guidance

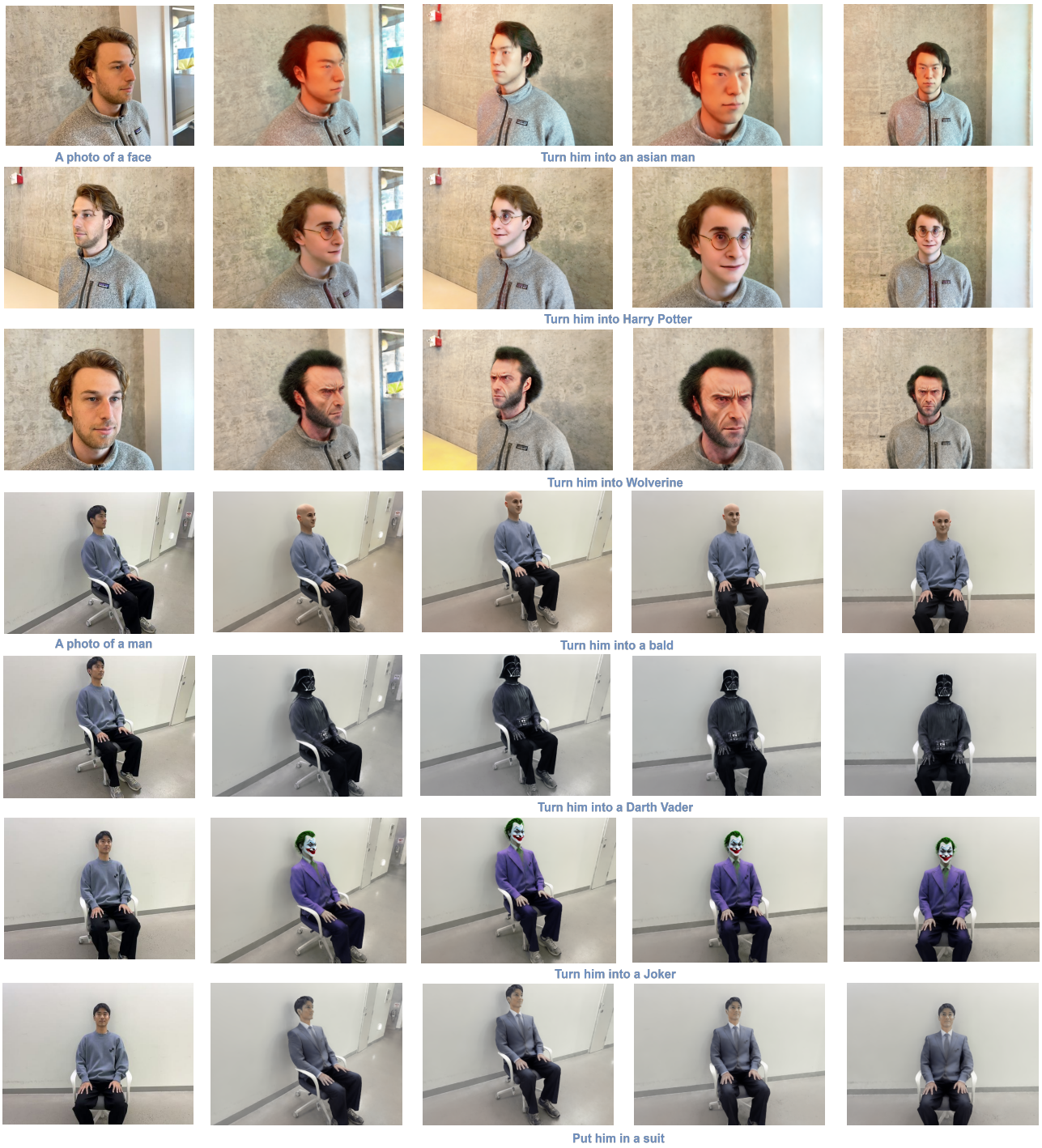

Figure 7. More editing results of *DaCapo* (Part 1).

scale respectively, $y_\varnothing$ and $\boldsymbol{x}_\varnothing$ are null text and image condition. $\boldsymbol{x}_t^{src}$ and $\boldsymbol{x}_t^{tgt}$ are noisy samples of source renderings and target renderings. The unconditional score ϵ_{uncond}^{tgt} and ϵ_{uncond}^{src} are obtained with null conditions:

$$\begin{aligned}\epsilon_{uncond}^{tgt} &= \boldsymbol{\epsilon}_\phi(\boldsymbol{x}_t^{\text{tgt}}, y_\varnothing, \boldsymbol{x}_\varnothing, t),\\ \epsilon_{uncond}^{src} &= \boldsymbol{\epsilon}_\phi(\boldsymbol{x}_t^{\text{src}}, y_\varnothing, \boldsymbol{x}_\varnothing, t).\end{aligned} \tag{21}$$

9.2. The weighting functions

As defined in equation 9, 16, and 17, the three terms all have a time-dependent weighting function. The weighting functions initially come from the score calculation or sample diffusion as in Equation 10 and 2, but we can extend them to general time-dependent weighting functions empirically for improved 3D editing performance. As noted in Section 4.2, the coupling weighting function controls the

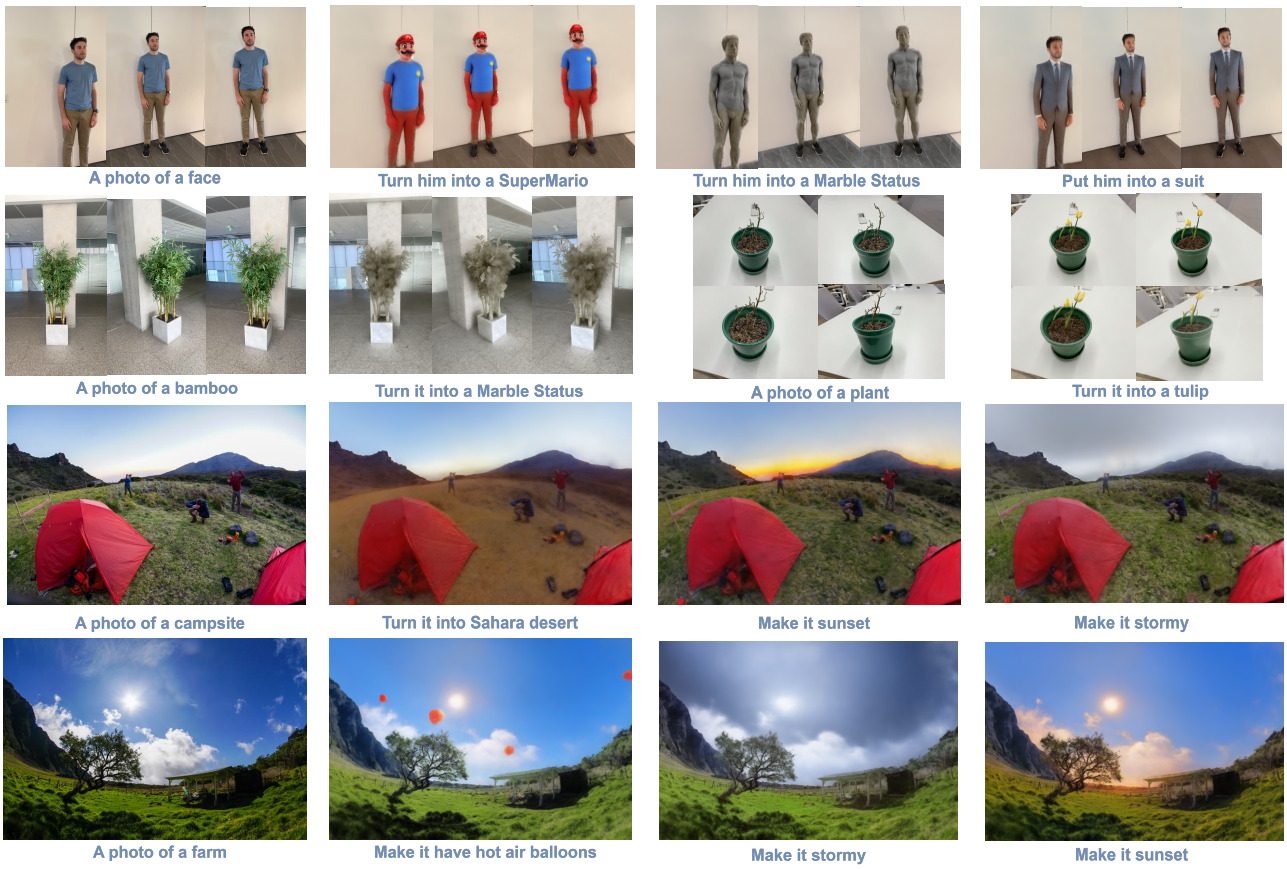

Figure 8. More editing results of *DaCapo* (Part 2).

Algorithm 1 The Stacked Bridge with *DaCapo*

Input: Max/min time step t_{max} and t_{min}, the total optimization step N the target/source prompt y^{tgt} and y^{src}, the source 3D scene model θ^{src}

1: Initialization: the 3D model to be updated $\theta = \theta^{src}$
2: **for** each step $i \in [1, N]$ of the 3D optimization **do**
3: Sample: Camera Pose c,
$\boldsymbol{x}_0^{tgt}, \boldsymbol{x}_0^{src} = g(\theta, c), g(\theta^{src}, c)$,
$t = t_{max} - (t_{max} - t_{min})\frac{i}{N}$.
4: Add noise:
$\boldsymbol{x}_t := \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t, \quad \epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Predict ϵ_ϕ^{tgt}, ϵ_ϕ^{src}, ϵ_{uncond}^{tgt} and ϵ_{uncond}^{src} via Equation 20 and 21
6: Compute the *DaCapo* vector field $v_\phi(t, \boldsymbol{x}_t)$ according to Equation 19.
7: Update the 3D model θ with $v_\phi(t, \boldsymbol{x}_t)$ via $v_\phi(t, \boldsymbol{x}_t)\frac{\partial \boldsymbol{x}_0^{tgt}}{\partial \theta}$
8: **end for**
9: **return** the edited 3D model parameter θ.

strength of source preservation, thus requiring a relatively strong preservation at the early stages of editing to constrain the editing process for the temporal separation of preservation and edit. Thus we define $w_{align}(t)$ as a decreasing function of time step t. The three weight functions used in *DaCapo* are as follows:

$$\begin{aligned} w_{edit}(t) &= 1 - 0.1 \cdot \sqrt{1-\bar{\alpha}_t} \\ w_{align}(t) &= 0.01 + 0.25 \cdot \sqrt{1-\bar{\alpha}_t} \\ w_{corret}(t) &= 1 - 0.5 \cdot \sqrt{1-\bar{\alpha}_t} \end{aligned} \tag{22}$$

where $\bar{\alpha} = \prod_{s=1}^{t} \alpha_s$ is the variance schedule variables.

9.3. Negative Classifier Scores

Inspired by NFSD, we could further include negative classifier scores (with negative prompt as text condition) as an additional component of MCG. We define the negative classifier scores as:

$$\epsilon_{neg} = w_{neg}(\boldsymbol{\epsilon}_\phi(\boldsymbol{x}_t^{\text{tgt}}, y_{neg}, \boldsymbol{x}^{\text{src}}, t) - \boldsymbol{\epsilon}_\phi(\boldsymbol{x}_t^{\text{tgt}}, y_\varnothing, \boldsymbol{x}_{src}, t))$$
$$w_2 = \begin{cases} 0.1 & \text{if } t > 0.2, \\ 0 & \text{otherwise.} \end{cases} \tag{23}$$

where y_{neg} is the negative prompts and w_{neg} is the coefficient for negative classifier score.

10. More editing Results

We present additional editing results to demonstrate the editing efficiency and quality of *DaCapo*. Relevant results

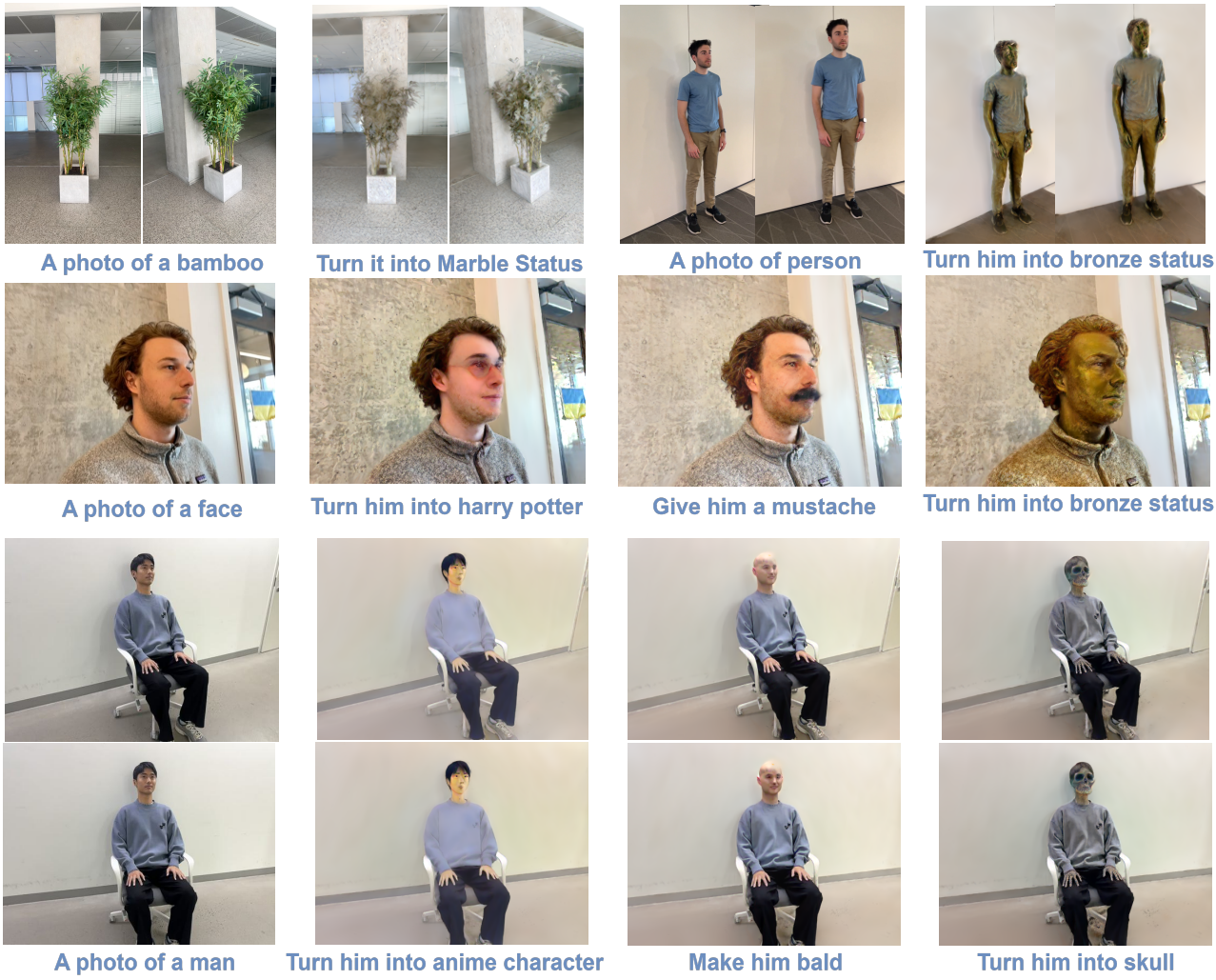

Figure 9. More editing results of *DaCapo* on 3DGS scenes.

Figure 10. Qualitative Comparsion of PDS and *DaCapo* at time-step 2500.

are shown in Figure 7 and Figure 8, **all of which were optimized within 2500 steps, with an average time of approximately 30 minutes**. Editing results on 3DGS are presented in Figure 9.

We also compare the editing results of PDS and *DaCapo* at 2500th step in Figure 10. The results demonstrate the effectiveness and efficiency of the proposed *DaCapo*.