

A. Implementation Details

For all experiments, we used an NVIDIA GeForce RTX 3090 GPU and a DDIM sampler [50], setting the total sampling time step to $T = 50$. We empirically set the time steps to $t \in [4, 50)$ for performing both our appearance matching self-attention and semantic matching guidance. We converted all self-attention modules in every decoder layer $l \in [1, 4)$ to the proposed appearance matching self-attention. We chose $\lambda_c = 0.4$ and $\lambda_g = 75$ for evaluation on the ViCo [22] dataset, and $\lambda_c = 0.4$ and $\lambda_g = 50$ for evaluation on the proposed challenging prompt list.

B. Dataset

Prior works [17, 32, 44] in Text-to-Image (T2I) personalization have used different datasets for evaluation. To ensure a fair and unbiased evaluation, ViCo [22] collected an image dataset from these works [17, 32, 44], comprising 16 unique concepts, which include 6 toys, 5 live animals, 2 types of accessories, 2 types of containers, and 1 building. For the prompts, ViCo gathered 31 prompts for 11 non-live objects and another 31 prompts for 5 live objects. These were modified from the original DreamBooth [44] prompts to evaluate the expressiveness of the objects in more complex textual contexts. For a fair comparison, in this paper, we followed the ViCo dataset and its evaluation settings, producing 8 samples for each object and prompt, totaling 3,969 images.

Our goal is to achieve semantically-consistent T2I personalization in complex non-rigid scenarios. To assess the robustness of our method in intricate settings, we created a prompt dataset using ChatGPT [39], which is categorized into three parts: large displacements, occlusions, and novel-view synthesis. The dataset comprises 10 prompts each for large displacements and occlusions, and 4 for novel-view synthesis, separately for live and non-live objects. Specifically, we define the text-to-image diffusion personalization task, provide an example prompt list from ViCo, and highlight the necessity of a challenging prompt list aligned with the objectives of our research. We then asked ChatGPT to create distinct prompt lists for each category. The resulting prompt list, tailored for complex non-rigid personalization scenarios, is detailed in Figure A.12.

C. Baseline and Comparison

C.1. Baseline

DreamMatcher is designed to be compatible with any T2I personalized model. We implemented our method using three baselines: Textual Inversion [17], DreamBooth [44], and CustomDiffusion [32].

Textual Inversion [17] encapsulates a given subject into 768-dimensional textual embeddings derived from the special token $\langle S^* \rangle$. Using a few reference images, this is

achieved by training the textual embeddings while keeping the T2I diffusion model frozen. During inference, the model can generate novel renditions of the subject by manipulating the target prompt with $\langle S^* \rangle$. DreamBooth [44] extends this approach by further fine-tuning a T2I diffusion model with a unique identifier and the class name of the subject (e.g., $A [V] cat$). However, fine-tuning all parameters can lead to a language shift problem [33, 36]. To address this, DreamBooth proposes a class-specific prior preservation loss, which trains the model with diverse samples generated by pre-trained T2I models using the category name as a prompt (e.g., $A cat$). Lastly, CustomDiffusion [32] demonstrates that fine-tuning only a subset of parameters, specifically the cross-attention projection layers, is efficient for learning new concepts. Similar to DreamBooth, this is implemented by using a text prompt that combines a unique instance with a general category, and it also includes a regularization dataset from the large-scale open image-text dataset [46]. Despite promising results, the aforementioned approaches frequently struggle to accurately mimic the appearance of the subject, including colors, textures, and shapes. To address this, we propose a tuning-free plug-in method that significantly enhances the reference appearance while preserving the diverse structure from target prompts.

C.2. Comparison

We benchmarked DreamMatcher against previous tuning-free plug-in models, FreeU [49] and MagicFusion [68], and also against the optimization-based model, ViCo [22].

The key insight of FreeU [49] is that the main backbone of the denoising U-Net contributes to low-frequency semantics, while its skip connections focus on high-frequency details. Leveraging this observation, FreeU proposes a frequency-aware reweighting technique for these two distinct features, and demonstrates improved generation quality when integrated into DreamBooth [44]. MagicFusion [68] introduces a saliency-aware noise blending method, which involves combining the predicted noises from two distinct pre-trained diffusion models. MagicFusion demonstrates its effectiveness in T2I personalization when integrating a personalized model, DreamBooth [44], with a general T2I diffusion model. ViCo [22] optimizes an additional image adapter designed with the concept of key-value replacement.

D. Evaluation

D.1. Evaluation Metrics

For evaluation, we focused on two primary aspects: subject fidelity and prompt fidelity. For subject fidelity, following prior studies [17, 22, 32, 44], we adopted the CLIP [42] and DINO [5] image similarity metrics, denoted as I_{CLIP} and I_{DINO} , respectively. Note that I_{DINO} is our preferred

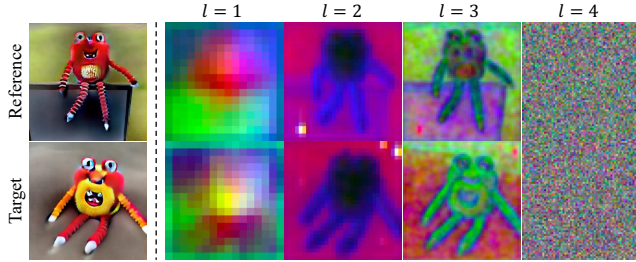


Figure A.1. **Diffusion feature visualization at different decoder layers:** The left side displays intermediate estimated reference and target images at 50% of the reverse diffusion process. The target is generated by DreamBooth [44] using the prompt $A \langle S^* \rangle$ on the beach. The right side visualizes the top three principal components of diffusion feature descriptors from different decoder layers l . Semantically similar regions share similar colors.

metric for evaluating subject expressivity. As mentioned in [22, 44], DINO is trained in a self-supervised manner to distinguish objects within the same category, so that it is more suitable for evaluating different methods that aim to mimic the visual attributes of the same subject. For prompt fidelity, following [17, 22, 32, 44], we adopted the image-text similarity metric T_{CLIP} , comparing CLIP visual features of the generated images to CLIP textual features of the corresponding text prompts, excluding placeholders. Following previous works [17, 22, 32, 44], we used ViT-B/32 [15] and ViT-S/16 [15] for CLIP and DINO, respectively.

D.2. User study

An example question of the user study is provided in Figure A.13. We conducted a paired human preference study about subject and prompt fidelity, comparing Dream-Matcher to previous works [22, 49, 68]. The results are summarized in Figure 10 in the main paper. For subject fidelity, participants were presented with a reference image and generated images from different methods, and were asked which better represents the subject in the reference. For prompt fidelity, they were shown the generated images from different works alongside the corresponding text prompt, and were asked which aligns more with the given prompt. 45 users responded to 32 comparative questions, resulting in a total of 1440 responses. We distributed two different questionnaires, with 23 users responding to one and 22 users to the other. Note that samples were chosen randomly from a large, unbiased pool.

E. Analysis

E.1. Appearance Matching Self-Attention

Feature extraction. Figure A.1 visualizes PCA [41] results on feature descriptors extracted from different decoder layers. Note that, for this analysis, we do not apply any of our proposed techniques. PCA is applied to the interme-

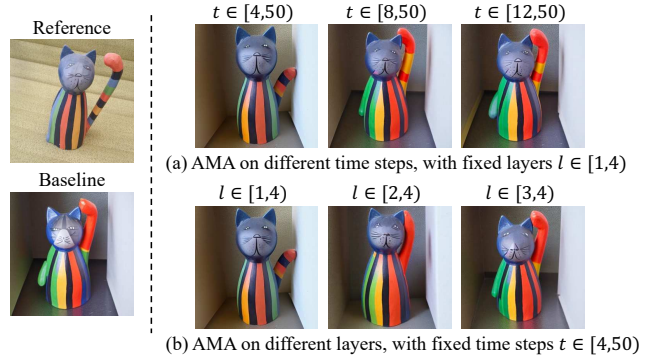


Figure A.2. **Ablating AMA on different time steps and layers:** The left section shows a reference image and a target image generated by the baseline [44]. The right section displays the improved target image generated by appearance matching self-attention on (a) different time steps and (b) different decoder layers. For this ablation study, we do not use semantic matching guidance.

diated feature descriptors of the estimated reference image and target image from DreamBooth [44], at 50% of the reverse diffusion process. Our primary insight is that earlier layers capture high-level semantics, while later layers focus on finer details of the generated images. Specifically, $l = 1$ captures overly high-level and low-resolution semantics, failing to provide sufficient semantics for finding correspondence. Conversely, $l = 4$ focuses on too fine-grained details, making it difficult to find semantically-consistent regions between features. In contrast, $l = 2$ and $l = 3$ strike a balance, focusing on sufficient semantical and structural information to facilitate semantic matching. Based on this analysis, we use concatenated feature descriptors from decoder layers $l \in [2, 3]$, resulting in $\psi_t \in \mathbb{R}^{H \times W \times 1920}$. We then apply PCA to these feature descriptors, which results in $\psi_t \in \mathbb{R}^{H \times W \times 256}$ to enhance matching accuracy and reduce memory consumption. The diffusion feature visualization across different time steps is presented in Figure 6 in our main paper.

Note that our approach differs from prior works [37, 54, 67], which select a specific time step and inject the corresponding noise into clean RGB images before passing them through the pre-trained diffusion model. In contrast, we utilize diffusion features from each time step of the reverse diffusion process to find semantic matching during each step of the personalization procedure.

AMA on different time steps and layers. We ablate starting time steps and decoder layers in relation to the proposed appearance matching self-attention (AMA) module. Figure A.2 summarizes the results. Interestingly, we observe that applying AMA at earlier time steps and decoder layers effectively corrects the overall appearance of the subject, including shapes, textures, and colors. In contrast, AMA applied at later time steps and layers tends to more closely preserve the appearance of the subject as in the baseline. Note that injecting AMA at every time step yields sub-optimal re-

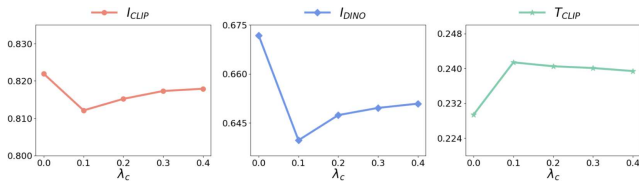


Figure A.3. **Relation between λ_c and personalization fidelity.** In this ablation study, we evaluate our method using the proposed challenging prompt list.

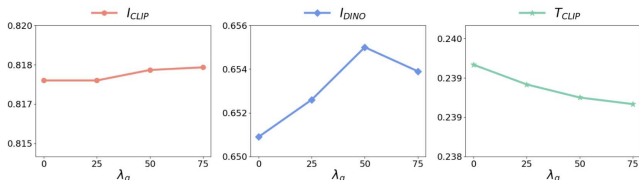


Figure A.4. **Relation between λ_g and personalization fidelity.** In this ablation study, we evaluate our method using the proposed challenging prompt list.

sults, as the baselines prior to time step 4 have not yet constructed the target image layout. Based on this analysis, we converted the self-attention module in the pre-trained U-Net into the appearance matching self-attention for $t \in [4, 50)$ and $l \in [1, 4)$ in all our evaluations.

E.2. Consistency Modeling

In Figure A.3, we show the quantitative relationship between the cycle-consistency hyperparameter λ_c and personalization fidelity. As we first introduce λ_c , prompt fidelity T_{CLIP} drastically improves, demonstrating that the confidence mask effectively filters out erroneous matches, allowing the model to preserve the detailed target structure. Subsequently, higher λ_c values inject more reference appearance into the target structure, increasing I_{DINO} and I_{CLIP} , but slightly sacrificing prompt fidelity T_{CLIP} . This indicates that users can control the extent of reference appearance and target structure preservation by adjusting λ_c . The pseudo code for overall AMA is available in Algorithm 1.

E.3. Semantic Matching Guidance

In Figure A.4, we display the quantitative relationship between semantic matching guidance λ_g and personalization fidelity. Increasing λ_g enhances subject fidelity I_{DINO} and I_{CLIP} , by directing the generated target $\hat{z}_{0,t}^Y$ closer to the clean reference latent z_0^X . However, excessively high λ_g can reduce subject fidelity due to discrepancies between the reference and target latents in early time steps. We carefully ablated the parameter λ_g and chose $\lambda_g = 75$ for the ViCo dataset and $\lambda_g = 50$ for the proposed challenging dataset. The pseudo code for overall semantic matching guidance is available in Algorithm 2.

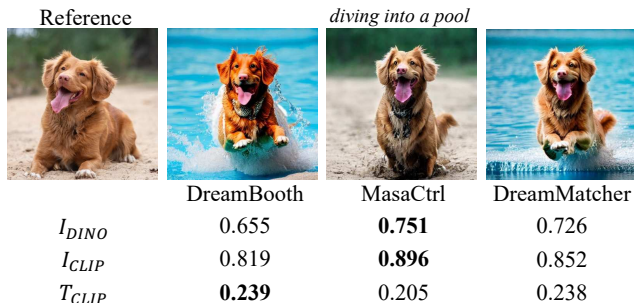


Figure A.5. **DreamBooth vs. MasaCtrl vs. DreamMatcher.**

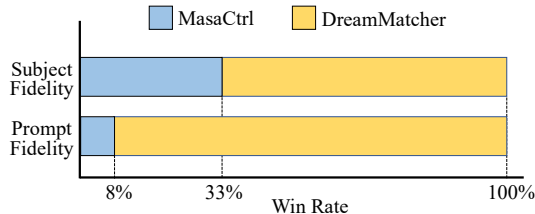


Figure A.6. **User study:** In this study, DreamBooth [44] is used as the baseline for both MasaCtrl and DreamMatcher.

E.4. Key-Value Replacement vs. DreamMatcher

MasaCtrl [4] introduced a key-value replacement technique for local editing tasks. Several subsequent works [4, 9, 11, 28, 31, 37] have adopted and further developed this framework. As shown in Figure A.5, which provides a qualitative comparison of DreamMatcher with MasaCtrl, the key-value replacement is prone to producing subject-centric images, often having poses similar to those of the subject in the reference image. This tendency arises because key-value replacement disrupts the target structure from the pre-trained self-attention module and relies on sub-optimal matching between target keys and reference queries. Furthermore, this technique does not consider the uncertainty of predicted matches, which leads to the injection of irrelevant elements from the reference image into the changed background or into newly emergent objects that are produced by the target prompts.

In contrast, DreamMatcher preserves the fixed target structure and accurately aligns the reference appearance by explicitly leveraging semantic matching. Our method also takes into account the uncertainty of predicted matches, thereby filtering out erroneous matches and maintaining newly introduced image elements by the target prompts. Note that the image similarity metrics I_{DINO} and I_{CLIP} do not simultaneously consider both the preservation of the target structure and the reflection of the reference appearance. They only calculate the similarities between the overall pixels of the reference and generated images. As a result, the key-value replacement, which generates subject-centric images and injects irrelevant elements from reference images into the target context, achieves better image similarities than DreamMatcher, as seen in Table 5 in the main paper. However, as shown in Figure A.5, DreamMatcher more ac-

Method	$I_{DINO} \uparrow$	$I_{CLIP} \uparrow$	$T_{CLIP} \uparrow$
Warping only reference values (DreamMatcher)	0.680	0.821	0.231
Warping both reference keys and values	0.654	0.809	0.235

Table A.1. Ablation study on key retention.

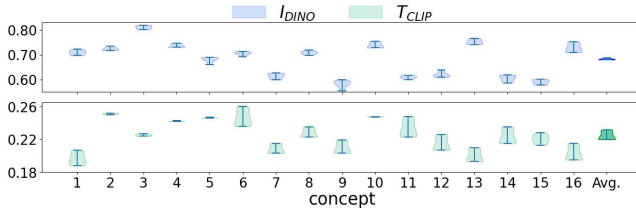


Figure A.7. Statistical results from 5 sets of randomly selected reference images.

curately aligns the reference appearance into the target context, even with large structural displacements. More qualitative comparisons are provided in Figures A.17 and A.18.

This is further demonstrated in a user study comparing MasaCtrl [4] and DreamMatcher, summarized in Figure A.6. A total of 39 users responded to 32 comparative questions, resulting in 1248 responses. These responses were divided between two different questionnaires, with 20 users responding to one and 19 to the other. Samples were chosen randomly from a large, unbiased pool. An example of this user study is shown in Figure A.14. DreamMatcher significantly surpasses MasaCtrl for both fidelity by a large margin, demonstrating the effectiveness of our proposed matching-aware value injection method.

E.5. Justification of Key Retention

DreamMatcher brings warped reference values to the target structure through semantic matching. This design choice is rational because we leverage the pre-trained U-Net, which has been trained with pairs of queries and keys sharing the same structure. This allows us to preserve the pre-trained target structure path by keeping target keys and queries unchanged. To validate this, Table A.1 shows a quantitative comparison between warping only reference values and warping both reference keys and values, indicating that anchoring the target structure path while only warping the reference appearance is crucial for overall performance. Concerns may arise regarding the misalignment between target keys and warped reference values. However, we emphasize that our appearance matching self-attention accurately aligns reference values with the target structure, ensuring that target keys and warped reference values are geometrically aligned as they were pre-trained.

E.6. Reference Selection

We evaluate the stability of DreamMatcher against variations in reference image selection by measuring the variance of all metrics across five sets of randomly selected reference images. Figure A.7 indicates that all metrics are closely distributed around the average. Specifically, the average

Method	$I_{DINO} \uparrow$	$I_{CLIP} \uparrow$	$T_{CLIP} \uparrow$
Textual Inversion [17]	0.529	0.762	0.220
Stable Diffusion (SD)	0.516	0.770	0.215
DreamMatcher	0.571 (+10.74%)	0.785 (+1.95%)	0.214 (-0.50%)

Table A.2. Quantitative results of DreamMatcher on Stable Diffusion.

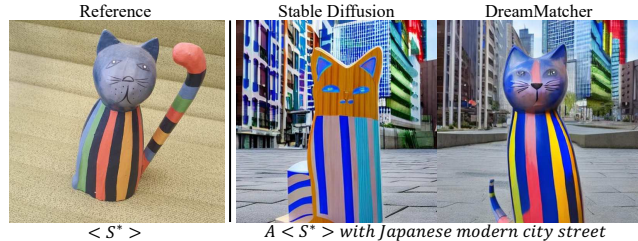


Figure A.8. Qualitative results of DreamMatcher on Stable Diffusion.

I_{DINO} is 0.683 with a variance of $6e-6$, and the average T_{CLIP} is 0.225 with a variance of $3e-5$. This highlights that our method is robust to reference selection and consistently generates reliable results. We further discuss the qualitative comparisons- with different reference images in Section G.

E.7. DreamMatcher on Stable Diffusion

DreamMatcher is a plug-in method dependent on the baseline, so we evaluated DreamMatcher on pre-trained personalized models [17, 32, 44] in the main paper. In this section, we also evaluated DreamMatcher using Stable Diffusion as a baseline. Table A.2 and Figure A.8 show that DreamMatcher enhances subject fidelity without any off-the-shelf pre-trained models, even surpassing I_{DINO} and I_{CLIP} of Textual Inversion which optimizes the 769-dimensional text embeddings.

E.8. Multiple Subjects Personalization

As shown in Figure A.9, we extend DreamMatcher for multiple subjects. For this experiments, we used CustomDiffusion [32] as a baseline. Note that a simple modification, which involves batching two different subjects as input, enables this functionality.

E.9. Computational Complexity

We investigate time and memory consumption on different configurations of our framework, as summarized in Table A.3. As seen, DreamMatcher significantly improves subject appearance with a reasonable increase in time and memory, compared to the baseline DreamBooth [44]. Additionally, we observe that reducing the PCA [41] dimension of feature descriptors before building the cost volume does not affect the overall performance, while dramatically reducing time consumption. Note that our method, unlike previous training-based [8, 10, 18, 29, 34, 48, 53, 63, 65] or



Figure A.9. **Qualitative results of DreamMatcher for multiple subject personalization.**

	Component	I_{DINO}	I_{CLIP}	T_{CLIP}	Time [s]	Mem. [GB]
(I)	Baseline (DreamBooth [44])	0.638	0.808	0.237	0.27	4.41
(II)	(I) + Appearance Matching Self-Att. (PCA 64)	0.675	0.818	0.233	0.38	4.49
(III)	(I) + Appearance Matching Self-Att. (PCA 256)	0.676	0.818	0.232	0.46	4.56
(IV)	(II) + Semantic Matching Guid.	0.680	0.820	0.231	0.57	4.85

Table A.3. **Computational complexity:** We used DreamBooth [44] as the baseline. For this analysis, we examine the time consumption for a single sampling time step.

optimization-based approaches [22], does not require any training or fine-tuning.

F. More Results

F.1. Comparison with Baselines

We present more qualitative results comparing with baselines, Textual Inversion [17], DreamBooth [44], and CustomDiffusion [32] in Figure A.15 and A.16.

F.2. Comparison with Previous Works

We provide more qualitative results in Figure A.17 and A.18 by comparing DreamMatcher with the optimization-based method ViCo [22] and tuning-free methods MasaCtrl [4], FreeU [49], and MagicFusion [68].

G. Limitation

Stylization. DreamMatcher may ignore stylization prompts such as $A \text{ red } \langle S^* \rangle$ or $A \text{ shiny } \langle S^* \rangle$, which do not appear in the reference images, as the model is designed to accurately inject the appearance from the reference. However, as shown in Figure A.10, combining off-the-shelf editing techniques [3, 24, 60] with DreamMatcher is highly effective in scenarios requiring both stylization and place alteration, such as $A \text{ red } \langle S^* \rangle \text{ on the beach}$. Specifically,

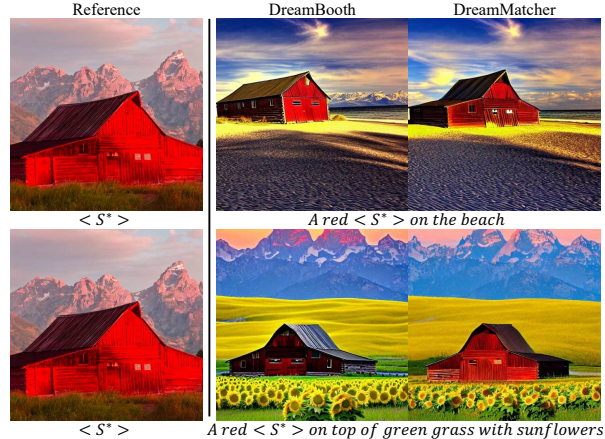
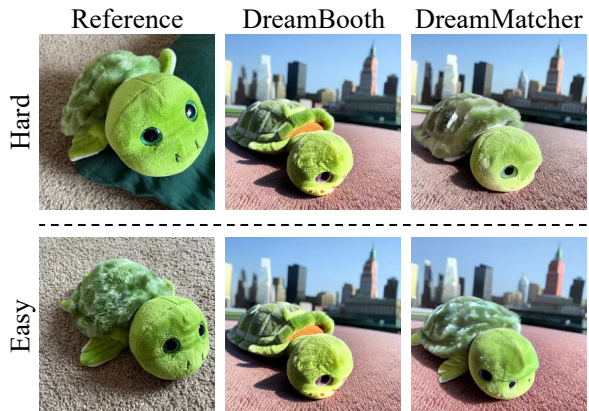


Figure A.10. **Integrating an image editing technique:** From left to right: the edited reference image by instructPix2Pix [3], the target image generated by the baseline, and the target image generated by DreamMatcher with the edited reference image. DreamMatcher can generate novel subject images by aligning the modified appearance with diverse target layouts.



$A \langle S^* \rangle \text{ in front of the Eiffel Tower}$

Figure A.11. **Impact of reference selection on personalization:** The top row presents results using a reference image that is difficult to match, while the bottom row shows results using a reference image that is relatively easier to match. The latter, containing sufficient visual attributes of the subject, leads to improved personalized results. This indicates that appropriate reference selection can enhance personalization fidelity.

we initially edit the reference image with existing editing methods to reflect the stylization prompt $red \langle S^* \rangle$, and then DreamMatcher generates novel scenes using this edited image. Our future work will focus on incorporating stylization techniques [3, 24, 60] into our framework directly, enabling the model to manipulate the reference appearance when the target prompt includes stylization.

Extreme Matching Case. In Appendix E.6, we demonstrate that our proposed method exhibits robust performance with randomly selected reference images. However, as depicted in Figure A.11, using a reference image that is relatively challenging to match may not significantly improve


the target image due to a lack of confidently matched appearances. This indicates even if our method is robust in reference selection, a better reference image which contains rich visual attributes of the subject will be beneficial for performance. Our future studies will focus on automating the selection of reference images or integrating multiple reference images jointly.

	Challenging text prompts for non-live objects	Challenging text prompts for live objects
Large displacement	a {} floating in a pond	a {} jumping over a cascading waterfall
	a {} bouncing on a trampoline	a {} swinging from the trees in a rainforest
	a {} on a wooden dock by a lake	a {} jumping over a fence
	a {} in a grassy park with a bench	a {} climbing a tree
	a {} on a brick pathway in a garden	a {} swinging on a swing
	a {} at the edge of a swimming pool	a {} diving into a pool
	a {} on a stone wall in the countryside	a {} running across a meadow
	a {} in a schoolyard playground	a {} hopping on stepping stones
	a {} on a sandy beach near the dunes	a {} leaping across rooftops in a city
	a {} at a picnic spot with a checkered blanket	a {} swimming upstream in a river
Occlusion	a {} partially covered by sand in the desert	a {} submerged halfway in a crystal-clear lake
	a {} nestled among rocks	a {} looking out from a tent
	a {} inside a box	a {} camouflaged in tall grass
	a {} inside a closet	a {} behind a cloud of smoke
	a {} between two chairs	a {} inside a cave entrance
	a {} inside a basket	a {} wearing a top hat
	a {} wearing a top hat	a {} wearing a backpack
	a {} wearing a scarf	a {} wearing a crown
	a {} in an astronaut outfit	a {} in an astronaut outfit
	a {} wearing bowtie	a {} wearing a bowtie
Novel-view synthesis	a {} seen from the top	a {} seen from the top
	a {} seen from the bottom	a {} seen from the bottom
	a {} seen from the back	a {} seen from the back
	a {} seen from the side	a {} seen from the side





Figure A.12. **Challenging text prompt list:** Evaluation prompts in complex, non-rigid scenarios for both non-live and live subjects. ‘{ }’ represents $\langle S^* \rangle$ in Textual Inversion [17] and ‘[V] class’ in DreamBooth [44] and CustomDiffusion [32].

Please rank the methods for generating an object **most similar** to the one contained in the following **reference image**. *

Reference







A **B** **C** **D**

	1	2	3	4
A	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
B	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
C	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
D	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Please rank the methods for generating an **image** most similar to the **given prompt**. *

A **B** **C** **D**




Prompt: A <S*> on top of green grass with sunflowers around it

	1	2	3	4
A	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
B	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
C	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
D	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Figure A.13. **An example of a user study comparing DreamMatcher with previous methods:** For subject fidelity, we provide the reference image and generated images from different methods, ViCo [22], FreeU [49], MagicFusion [68] and DreamMatcher. For prompt fidelity, we provide the target prompt and the generated images from those methods. For a fair comparison, we randomly choose the image samples from a large, unbiased pool.

Please choose the method for generating an object most similar to the one contained in the following reference image. *

Reference **A** **B**




  

A

B

Please choose the method for generating an image most similar to the given prompt. *

Reference **A** **B**

Prompt: A <S*> on a stone wall in the countryside

A

B

Figure A.14. **An example of a user study comparing DreamMatcher with MasaCtrl [4]:** For subject fidelity, we provide the reference image and images generated from two different methods, MasaCtrl and DreamMatcher. For prompt fidelity, the target prompt and generated images from these two methods are provided. For a fair comparison, image samples are randomly chosen from a large, unbiased pool.

Algorithm 1 Pseudo-Code for Appearance Matching Self-Attention, PyTorch-like

```
def AMA(self, pca_feats, q_tgt, q_ref, k_tgt, k_ref, v_tgt, v_ref, mask_tgt, cc_thres, num_heads, **kwargs):  
    # Initialize dimensions and rearrange inputs.  
    B, H, W = init_dimensions(q_tgt, num_heads)  
    q_tgt, q_ref, k_tgt, k_ref, v_tgt, v_ref = rearrange_inputs(q_tgt, q_ref, k_tgt, k_ref, v_tgt, v_ref,  
        num_heads, H, W)  
  
    # Perform feature interpolation and rearrangement.  
    src_feat, trg_feat = interpolate_and_rearrange(pca_feats, H)  
    src_feat, trg_feat = l2_norm(src_feat, trg_feat)  
  
    # Compute similarity.  
    sim = compute_similarity(trg_feat, src_feat)  
  
    # Calculate forward and backward similarities and flows.  
    sim_backward = rearrange(sim, ``b (Ht Wt) (Hs Ws) -> b (Hs Ws) Ht Wt``)  
    sim_forward = rearrange(sim, ``b (Ht Wt) (Hs Ws) -> b (Ht Wt) Hs Ws``)  
  
    flow_tgt_to_ref, flow_ref_to_tgt = compute_flows_with_argmax(sim_backward, sim_forward)  
  
    # Compute cycle consistency error and confidence.  
    cc_error = compute_cycle_consistency(flow_tgt_to_ref, flow_ref_to_tgt)  
    fg_ratio = mask_tgt.sum() / (H * W)  
    confidence = (cc_error < cc_thres * H * fg_ratio)  
  
    # Warp value and apply semantic-consistent mask.  
    warped_v = warp(v_ref, flow_tgt_to_ref)  
    warped_v = warped_v * confidence + v_tgt * (1 - confidence)  
    warped_v = warped_v * mask_tgt + v_tgt * (1 - mask_tgt)  
  
    # Perform self-attention.  
    aff = compute_affinity(q_tgt, k_tgt)  
    attn = aff.softmax(-1)  
    out = compute_output(attn, warped_v)  
  
    return out
```

Algorithm 2 Pseudo-Code for Semantic Matching Guidance, PyTorch-like

```
for i, t in enumerate(tqdm(self.scheduler.timesteps)):

    # Define model inputs.
    latents = combine_latents(latents_ref, latents_tgt)

    # Enable gradient computation for matching guidance.
    enable_gradients(latents)

    # Sampling and feature extraction from the U-Net.
    noise_pred, feats = self.unet(latents, t, text_embeddings)

    # Interpolation and concatenating features from different layers.
    src_feat_uncond, tgt_feat_uncond, src_feat_cond, tgt_feat_cond = interpolate_and_concat(feats)

    # Perform PCA and normalize the features.
    pca_feats = perform_pca_and_normalize(src_feat_uncond, tgt_feat_uncond, src_feat_cond, tgt_feat_cond)

    # Apply semantic matching guidance if required.
    if matching_guidance and (i in self.mg_step_idx):
        _, pred_z0_tgt = self.step(noise_pred, t, latents)
        pred_z0_src = image_to_latent(src_img)
        uncond_grad, cond_grad = compute_gradients(pred_z0_tgt, pred_z0_src, t, pca_feats)

        alpha_prod_t = self.scheduler.alphas_cumprod[t]
        beta_prod_t = 1 - alpha_prod_t
        noise_pred[1] -= grad_weight * beta_prod_t**0.5 * uncond_grad
        noise_pred[3] -= grad_weight * beta_prod_t**0.5 * cond_grad

    # Apply classifier-free guidance for enhanced generation.
    if guidance_scale > 1.0:
        noise_pred = classifier_free_guidance(noise_pred, guidance_scale)

    # Step from z_t to z_{t-1}.
    latents = self.step(noise_pred, t, latents)
```



Figure A.15. **Qualitative comparison with baselines for live objects:** We compare DreamMatcher with three different baselines, Textual Inversion [17], DreamBooth [44], and CustomDiffusion [32].

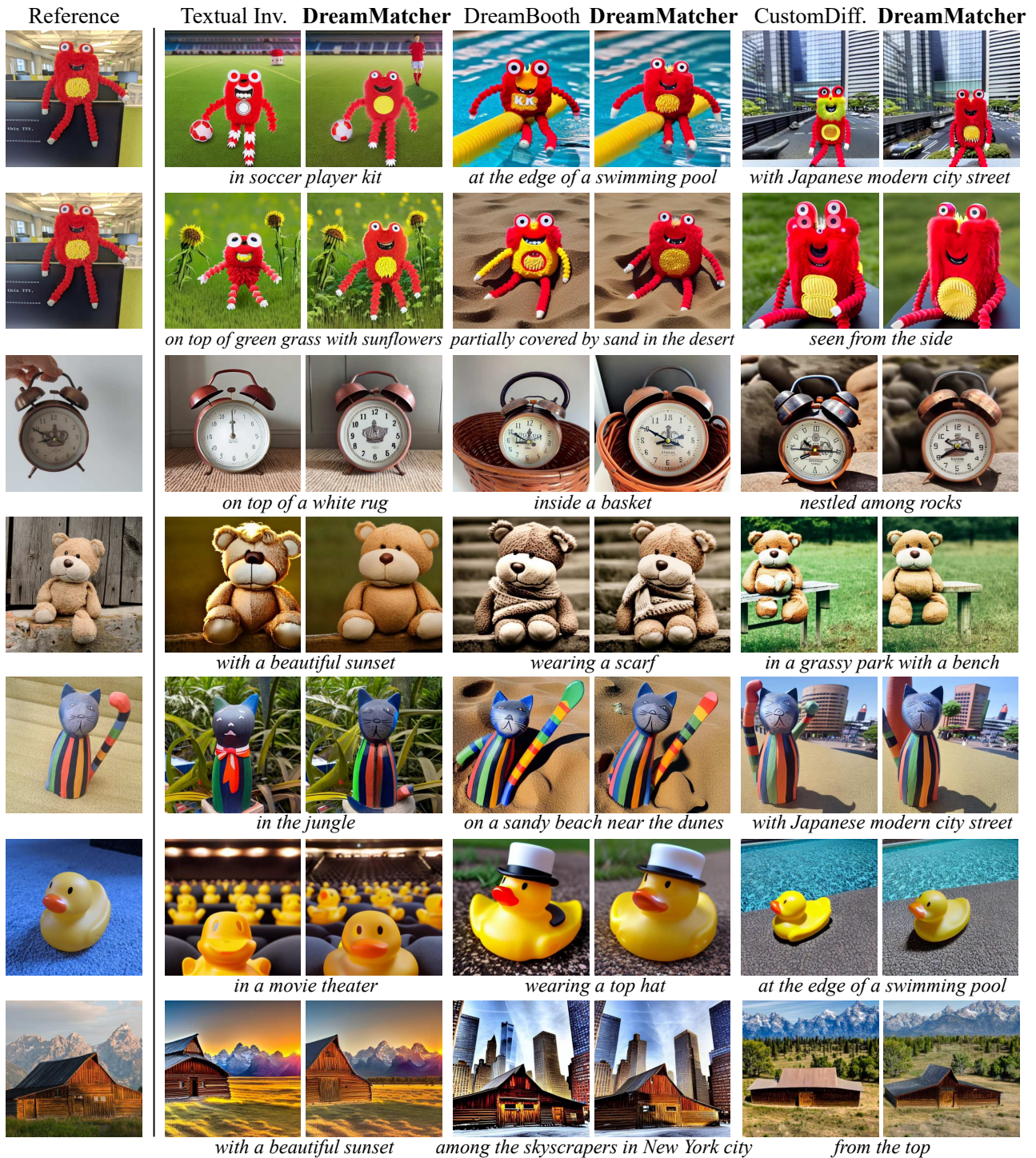


Figure A.16. **Qualitative comparison with baselines for non-live objects:** We compare DreamMatcher with three different baselines, Textual Inversion [17], DreamBooth [44], and CustomDiffusion [32].



Figure A.17. **Qualitative comparison with previous works [4, 22, 44, 49, 68] for live objects:** For this comparison, DreamBooth [44] is used as the baseline for MasaCtrl [4], FreeU [49], MagicFusion [68], and DreamMatcher.

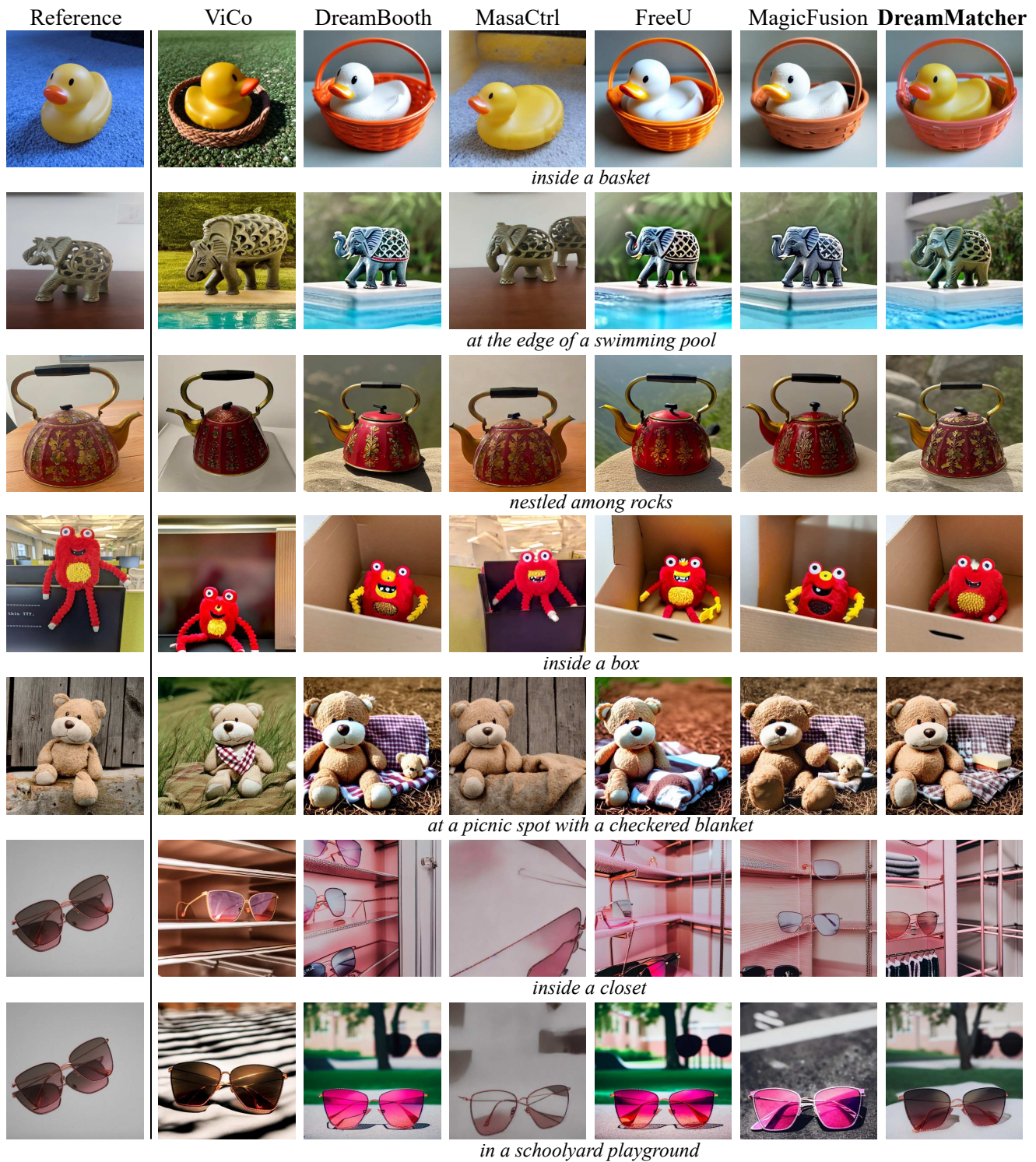


Figure A.18. **Qualitative comparison with previous works [4, 22, 44, 49, 68] for non-live objects:** For this comparison, Dream-Booth [44] is used as the baseline for MasaCtrl [4], FreeU [49], MagicFusion [68], and DreamMatcher.