Supplementary Material

A. Comparisons to Current Methods

In the main paper, we compared Vibe Space with CLIP Avg (our baseline), GPT [10], and Gemini [3]. In this section, we also compare with recent concept-level and pixel-level blending methods, including AID [6], Yu et al. [19], and DiffMorpher [20]. Our experiments demonstrate that CLIP Avg is stronger than these additional baselines.

We use image pairs from the Totally-Looks-Like dataset. Our user studies provide two annotations for each pair: (1) a Blend Difficulty level (low, medium, high) and (2) a short text description of the main shared attribute, or "vibe".

Qualitative Comparison. As illustrated in Figs. 3 to 5, pixel-level blending methods such as Yu et al. [19] and DiffMorpher [20] often produce blurred or low-quality intermediate images. CLIP Avg and AID [6] generate high-quality images but frequently fail to capture the main shared attribute (e.g., Fig. 4b: "Hair Style"; Fig. 3b: "Teeth and Eyes"). GPT and Gemini typically perform style transfer or part-level composition rather than attribute-level blending. In contrast, our **Vibe Space** consistently produces high-quality images while keeping the blending trajectory centered on the attribute identified by users.

Quantitative Comparison. We evaluate concept-level blending performance using our *Attribute-Masked DreamSim* metric. For each image pair (I_A, I_B), our user study provides a text description of the main shared attribute between the images, with examples in Fig. 5. We use this text to obtain a segmentation mask using an open-vocabulary segmentation model [8]. All image feature similarity computations are restricted to pixels inside the attribute mask. Specifically, given an image I, let $x_{\text{dreamsim}}(I)$ denote its dense DreamSim [5] features (prior to global pooling). For a masked region Mask, we compute the attribute-masked DreamSim embedding via mean pooling:

$$v(I; \text{Mask}) = \frac{1}{|\text{Mask}|} \sum_{p \in \text{Mask}} x_{\text{dreamsim}}(I)_p. \quad (1)$$

DreamSim similarity between two images is defined as cosine similarity between the attribute-masked embeddings:

$$\text{sim} = \frac{\langle v(I_A; \text{Mask}),\, v(I_B; \text{Mask}) \rangle}{\|v(I_A; \text{Mask})\|_2 \, \|v(I_B; \text{Mask})\|_2}. \quad (2)$$

For each method, we measure how well the midpoint blend I_{mid} preserves the target attribute region by computing: (1) $\text{sim}(I_{\text{mid}}, I_A)$, (2) $\text{sim}(I_{\text{mid}}, I_B)$, and (3) the average of these two scores.

As shown in Table 1, **Vibe Space** achieves the strongest performance among concept-level blending methods, while our CLIP Avg baseline remains the best-performing pixel-level blending method.

The *Attribute-Masked DreamSim* metric does not fully capture the differences between the methods. While the scores for Vibe Space and the CLIP Avg baseline are numerically similar in Table 1, Fig. 1 shows qualitative discrepancies. Specifically, CLIP Avg fails to accurately capture the primary attributes, such as the mouth in the first row, the hairstyle in the second row, and the hairstyle in the third row. Instead, CLIP Avg often blends all possible attributes simultaneously, whereas our Vibe Space approach effectively prioritizes and blends the main attributes before considering sub-attributes.

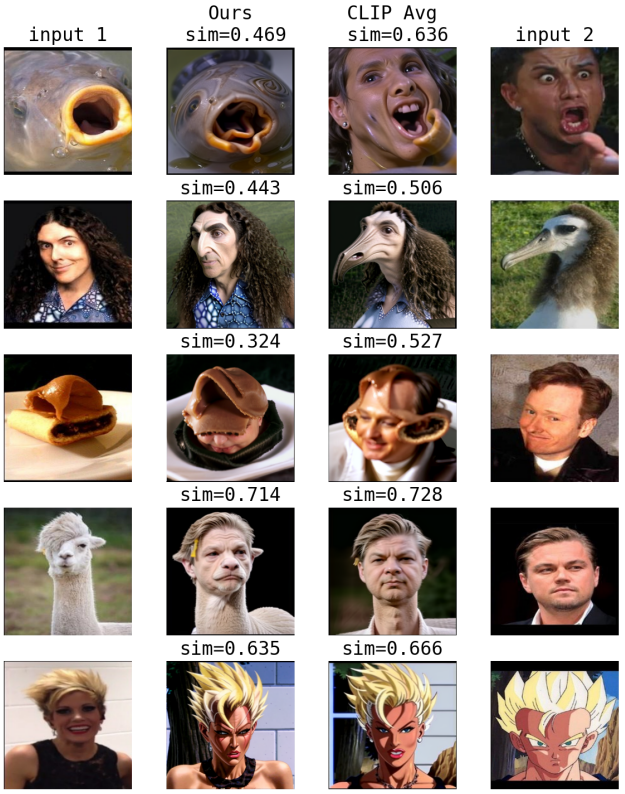
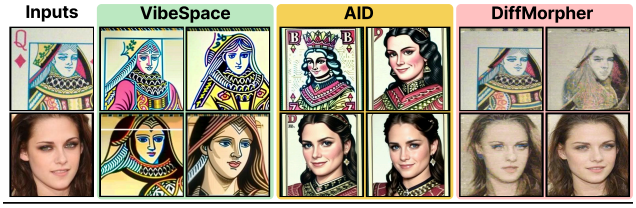


Figure 1. Limitation of quantitative evaluation. While CLIP Avg achieved a higher score on the first three pairs, it fails to capture the main attributes (mouth in the first row, hairstyle in the second row, hairstyle in the third row) and instead blends all possible attributes. Our Vibe Space, on the other hand, successfully blends the main attributes without blending distracting attributes.



Method	Totally-Looks-Like			Architecture		
	LPIPS Smooth. ↑	LPIPS Consis. ↓	FID ↓	LPIPS Smooth. ↑	LPIPS Consis. ↓	FID ↓
Vibe Space	**0.874** (0.04)	0.226 (0.05)	134.70	**0.876** (0.04)	0.223 (0.03)	64.81
AID	0.818 (0.05)	0.235 (0.03)	131.03	0.810 (0.05)	0.223 (0.03)	76.14
DiffMorpher	0.737 (0.05)	**0.148** (0.03)	**121.30**	0.743 (0.05)	**0.159** (0.03)	**51.35**

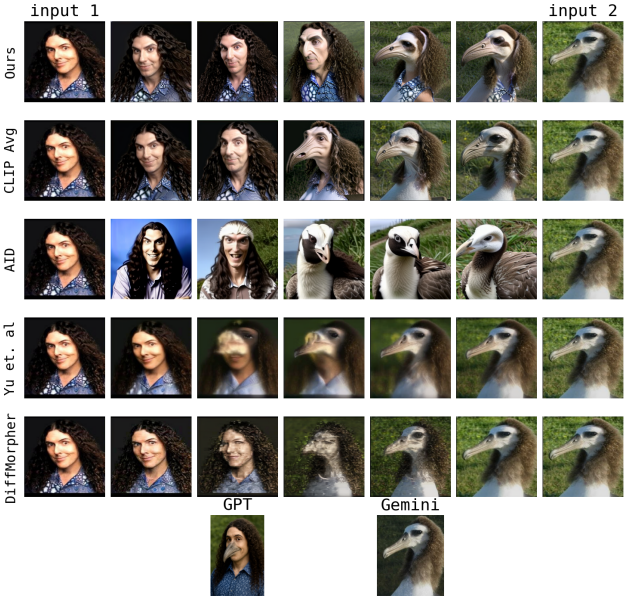
Figure 2. **Standard metrics can be misleading.** DiffMorpher scores well on LPIPS/FID but produces qualitatively poor blends.

We also compare against standard metrics (LPIPS smoothness, LPIPS consistency, FID) used by AID [6]. Results are in Fig. 2. Despite not being optimized for smoothness, our method achieves the best smoothness score on both the Totally Looks Like and Architecture datasets. Our lower performance on consistency and FID is expected; these metrics favor minimal pixel change, which is orthogonal to our goal of creative vibe blending.

	Attribute Masked DreamSim								
	High Difficulty			Medium Difficulty			Low Difficulty		
Method	Input 1	Input 2	Mean	Input 1	Input 2	Mean	Input 1	Input 2	Mean
Concept-level blending methods									
VibeSpace (ours)	0.632	0.540	**0.586**	0.642	0.562	**0.602**	0.708	0.596	**0.652**
AID [6]	0.483	0.531	<u>0.507</u>	0.570	0.468	<u>0.519</u>	0.599	0.531	<u>0.565</u>
GPT [10]	0.412	0.336	0.374	0.314	0.290	0.302	0.444	0.355	0.399
Gemini [3]	0.410	0.388	0.399	0.300	0.285	0.292	0.518	0.454	0.486
Pixel-level blending methods									
VibeSpace (pixel-level blending)	0.552	0.621	**0.586**	0.600	0.544	<u>0.572</u>	0.660	0.591	<u>0.626</u>
CLIP Avg (our baseline)	0.594	0.573	<u>0.584</u>	0.584	0.581	**0.583**	0.632	0.648	**0.640**
Yu et al. [19]	0.554	0.505	0.530	0.493	0.482	0.487	0.571	0.581	0.576
DiffMorpher [20]	0.390	0.483	0.437	0.440	0.586	0.513	0.547	0.591	0.569

Table 1. Comparison to current methods on the Totally Looks Like dataset [13]. We report *Attribute-Masked DreamSim* computed in three steps: (1) the main shared attribute for each image pair is obtained from our user study; (2) an open-vocabulary segmentation model [8] is used to generate a mask for this attribute; (3) DreamSim features are extracted and cosine similarity is computed only over the masked region. For each method, the blended midpoint image is compared to both input images: the "Input 1" and "Input 2" columns report DreamSim similarity between the midpoint and each input, respectively, and the "Mean" column reports their average. Bolded numbers denote the best-performing method among concept-level methods and among pixel-level methods within their respective groups, underlined numbers mark the second best.

Insights: (1) Vibe Space achieves the strongest performance among concept-level blending methods; (2) CLIP Avg—our baseline—is a strong pixel-level method and performs best within its category.



(a) User rated blend difficulty is: High.
User annotated main attribute: "Hair Style"

(b) User rated blend difficulty is: High.
User annotated main attribute: "Teeth and Eyes"

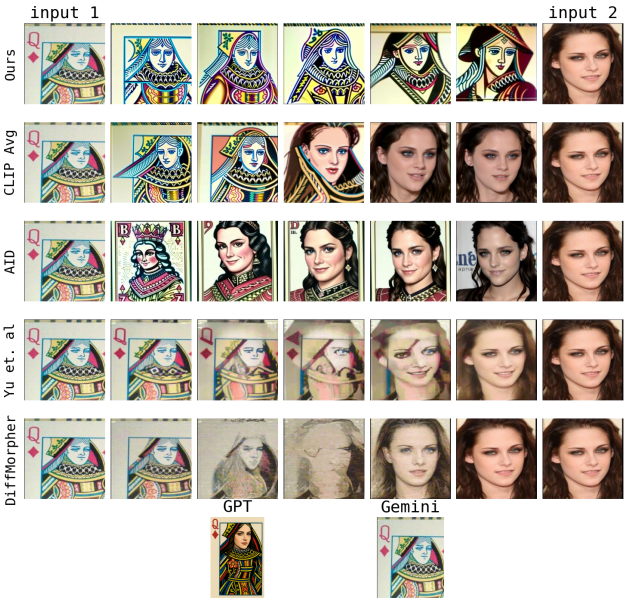
(c) User rated blend difficulty is: High.
User annotated main attribute: "Mouth Shape"

(d) User rated blend difficulty is: High.
User annotated main attribute: "Curly Hair"

Figure 3. Comparison to current methods. In this set of examples, blend difficulty is "High". **Insight:** Our method captures the main attribute, AID [6] failed to capture the main attribute and sometimes generate images unrelated to input 1 and input 2, Yu et. al [19] and DiffMorpher [20] produce low-quality images. Although the visual difference between ours vs CLIP Avg is high—CLIP Avg does not capture the main attribute while ours does—the quantitative measure in Table 1 shows small difference between Ours and CLIP Avg, this highlights the limitation of quantitative metrics in evaluating Vibe Blending.

(a) User rated blend difficulty is: Medium.
User annotated main attribute: "Facial Expression"

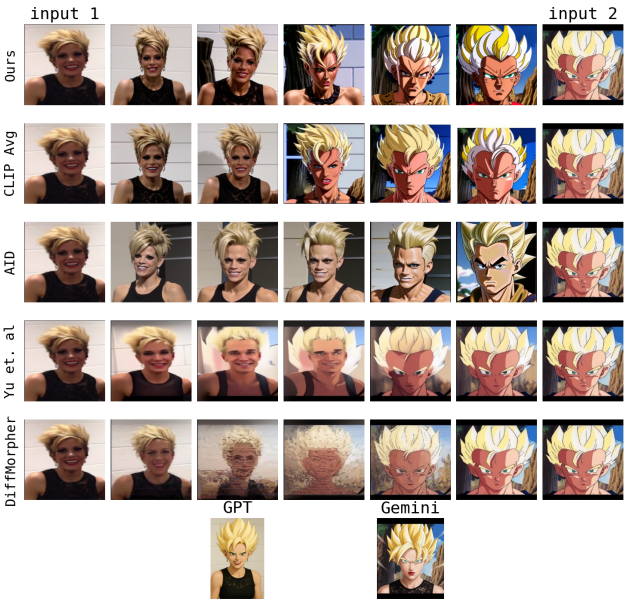
(b) User rated blend difficulty is: Medium.
User annotated main attribute: "Hair Style"



(c) User rated blend difficulty is: Medium.
User annotated main attribute: "Grin"

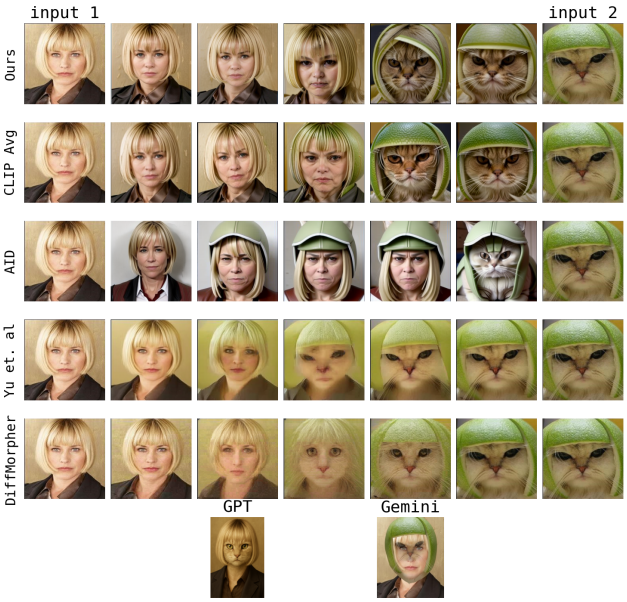
(d) User rated blend difficulty is: Medium.
User annotated main attribute: "Hair Style"

Figure 4. Comparison to current methods. In this set of examples, blend difficulty is "Medium". **Insight:** Our method captures the main attribute, AID [6] failed to capture the main attribute and sometimes generate images unrelated to input 1 and input 2, Yu et. al [19] and DiffMorpher [20] produce low-quality images.



(a) User rated blend difficulty is: Low.
User annotated main attribute: "Spiky Hair"

(b) User rated blend difficulty is: Low.
User annotated main attribute: "Grimace"



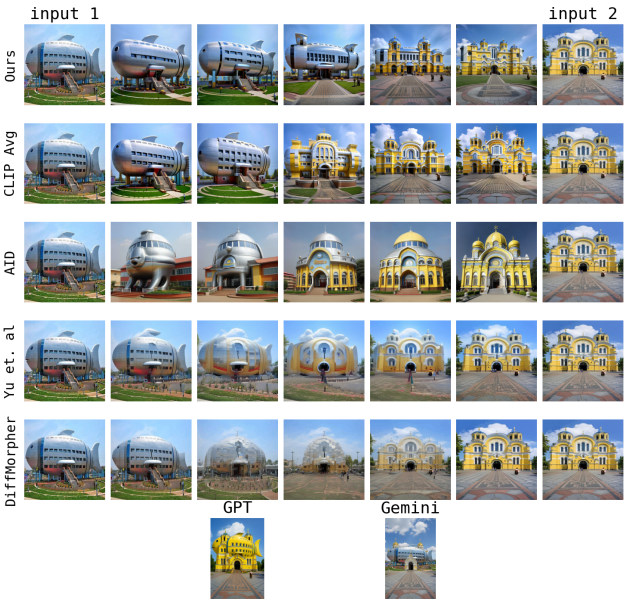
(c) User rated blend difficulty is: Low.
User annotated main attribute: "Hair Style"

(d) User rated blend difficulty is: Low.
User annotated main attribute: "Clothing"

Figure 5. Comparison to current methods. In this set of examples, blend difficulty is "Low". **Insight:** Our method captures the main attribute, AID [6] failed to capture the main attribute and sometimes generate images unrelated to input 1 and input 2, Yu et. al [19] and DiffMorpher [20] produce low-quality images. The visual difference between Ours vs CLIP Avg is small—both methods can generate coherent blend on the main attributes.

(a) Architecture dataset.

(b) Architecture dataset.



(c) Architecture dataset.

(d) Architecture dataset.

Figure 6. Comparison to current methods. **Insight:** Our method captures the main attribute, AID [6] failed to capture the main attribute and sometimes generate images unrelated to input 1 and input 2, Yu et. al [19] and DiffMorpher [20] produce low-quality images.

B. Ablation Studies

We conduct a series of ablations to understand the contribution of each component in Vibe Space. Experiments are performed on the Totally-Looks-Like dataset and evaluated using Attribute-Masked DreamSim (see Section A). Across all studies, we report performance separately for high-, medium-, and low-difficulty image pairs.

Flag Loss. The flag loss aligns the learned Vibe Space with the multiscale geometry of the underlying diffusion map by matching the Gram matrix of Vibe features to the flag-space kernel (see Equation (4)). This loss depends on a set of scales $\mathcal{M}$, where each $m \in \mathcal{M}$ corresponds to a prefix of low-frequency Laplacian eigenvectors.

We ablate the following setting:

- *multi-scale coarse-to-fine* ($\mathcal{M} = \{4, 8, \ldots, 64\}$)
- *single-scale fine-only* ($\mathcal{M} = \{64\}$)
- *single-scale coarse-only* ($\mathcal{M} = \{4\}$)
- *no flag loss* ($\mathcal{M} = \varnothing$)

Results are shown in Table 2 and Fig. 8. Multi-scale supervision provides the most robust performance across all difficulty levels. Single-scale variants work well only in specific regimes—fine-grained-only excels on low-difficulty cases where attributes are small and localized; coarse-only works best on high-difficulty pairs where fine-grained attribute connections are hard to discover. Removing flag loss significantly degrades performance, confirming that multiscale geometric alignment is essential for coherent Vibe Blending.

Correspondence. Vibe Blending relies on region-level correspondence between DINO token clusters in I_A and I_B (see Section D.1). We ablate both the number of clusters and the complete removal of correspondence.

In Table 3, using correspondence consistently outperforms removing it on medium- and low-difficulty cases. When difficulty is high, using fewer clusters (e.g., 10) is beneficial because high-difficulty pairs have less spatial alignment, making fine-grained clustering unstable. Without correspondence (i.e., pixel-level blending), performance is competitive only for high-difficulty pairs but fails on medium/low cases and produces incoherent blends when inputs are spatially misaligned (see qualitative examples in Section D.1). Overall, correspondence is crucial for concept-level blending.

DINO–CLIP Feature Mixing. Vibe Space is trained by mapping dense DINO features into Vibe Space and decoding back into CLIP space to leverage both fine-grained semantics (DINO) and generative compatibility (CLIP). We ablate this design by replacing DINO features with CLIP or MAE features as the encoder input.

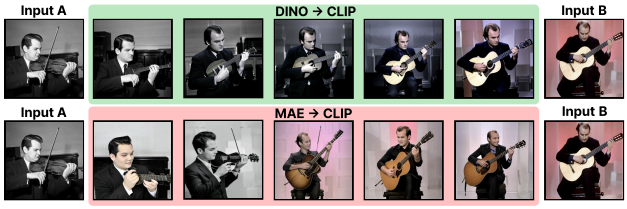


Figure 7. Vibe Blending with DINO→CLIP vs. MAE→CLIP. MAE features failed to embed semantics thus produce poor blend.

We show qualitative comparison in Fig. 7, MAE-CLIP fails to preserve instrument semantics and cannot produce a coherent blend between the violin and guitar. The MAE-trained Vibe path exhibits a semantic jump, which is captured as a localized region of high curvature along the Vibe path. Across our Totally Looks Alike subset, the DINO-trained Vibe path yields substantially lower max curvature (23.16) than MAE (31.27).

We show quantitative comparison in Table 4, mixing DINO and CLIP features achieves the best performance on medium- and low-difficulty pairs. For the most challenging pairs, CLIP-only encoding sometimes performs slightly better, likely because CLIP already captures high-level semantics that dominate these pairs. However, DINO–CLIP mixing remains the more stable choice across all difficulty levels.

	Attribute Masked DreamSim								
	High Difficulty			Medium Difficulty			Low Difficulty		
Method	Input 1	Input 2	Mean	Input 1	Input 2	Mean	Input 1	Input 2	Mean
Multi-scale (coarse-to-fine) $\mathcal{M} = \{4, 8, \ldots, 64\}$	0.626	0.420	0.523	0.669	0.490	**0.580**	0.668	0.618	0.643
Single-scale (fine) $\mathcal{M} = \{64\}$	0.650	0.459	0.554	0.621	0.490	0.555	0.731	0.573	**0.652**
Single-scale (coarse) $\mathcal{M} = \{4\}$	0.625	0.525	**0.575**	0.651	0.496	0.573	0.749	0.545	0.647
No flag loss $\mathcal{M} = \varnothing$	0.450	0.579	0.514	0.568	0.545	0.556	0.658	0.546	0.602

Table 2. Ablation of our methods on the Totally-Looks-Like dataset. We report *Attribute-Masked DreamSim* computed in three steps: (1) the main shared attribute for each image pair is obtained from our user study; (2) an open-vocabulary segmentation model [8] is used to generate a mask for this attribute; (3) DreamSim features are extracted and cosine similarity is computed only over the masked region. For each method, the blended midpoint image is compared to both input images: the "Input 1" and "Input 2" columns report DreamSim similarity between the midpoint and each input, respectively, and the "Mean" column reports their average. Bold numbers denote the best-performing method.
Insights: (1) With flag loss works better than no flag loss across all low-to-high difficulty cases. (2) Single-scale fine-grained-only works best on low difficulty cases; coarse-only works best on high difficulty cases. Multi-scale works well on all cases, and multi-scale is more robust than single-scale.

	Attribute Masked DreamSim								
	High Difficulty			Medium Difficulty			Low Difficulty		
Method	Input 1	Input 2	Mean	Input 1	Input 2	Mean	Input 1	Input 2	Mean
w/ correspondence $n_{\text{clusters}} = 10$	0.632	0.540	**0.586**	0.592	0.572	0.582	0.714	0.557	0.636
w/ correspondence $n_{\text{clusters}} = 20$	0.681	0.451	0.566	0.617	0.554	0.585	0.724	0.555	0.639
w/ correspondence $n_{\text{clusters}} = 30$	0.658	0.466	0.562	0.642	0.562	**0.602**	0.710	0.552	0.631
w/ correspondence $n_{\text{clusters}} = 40$	0.581	0.545	0.563	0.625	0.517	0.571	0.708	0.596	**0.652**
w/ correspondence $n_{\text{clusters}} = 50$	0.630	0.459	0.544	0.640	0.532	0.586	0.720	0.540	0.630
w/o correspondence (pixel-level blending)	0.552	0.621	**0.586**	0.600	0.544	0.572	0.660	0.591	0.626

Table 3. Ablation of our methods on the Totally-Looks-Like dataset. We report *Attribute-Masked DreamSim* computed in three steps: (1) the main shared attribute for each image pair is obtained from our user study; (2) an open-vocabulary segmentation model [8] is used to generate a mask for this attribute; (3) DreamSim features are extracted and cosine similarity is computed only over the masked region. For each method, the blended midpoint image is compared to both input images: the "Input 1" and "Input 2" columns report DreamSim similarity between the midpoint and each input, respectively, and the "Mean" column reports their average. Bold numbers denote the best-performing method.
Insights: (1) When increasing blend difficulty from low to high, computing correspondence with less clusters works better, this is because correspondence is harder to compute for large number of clusters on high difficulty cases (see Section D.1) (2) Although w/o correspondence works great on high difficulty examples, it doesn't work well on medium and low difficulty cases.

	Attribute Masked DreamSim								
	High Difficulty			Medium Difficulty			Low Difficulty		
Method	Input 1	Input 2	Mean	Input 1	Input 2	Mean	Input 1	Input 2	Mean
f : DINO $\rightarrow$ Vibe, g : Vibe $\rightarrow$ CLIP	0.626	0.420	0.523	0.669	0.490	**0.580**	0.668	0.618	**0.643**
f : CLIP $\rightarrow$ Vibe, g : Vibe $\rightarrow$ CLIP	0.648	0.444	**0.546**	0.731	0.422	0.577	0.665	0.492	0.579

Table 4. Ablation of our methods on the Totally-Looks-Like dataset. We report *Attribute-Masked DreamSim* computed in three steps: (1) the main shared attribute for each image pair is obtained from our user study; (2) an open-vocabulary segmentation model [8] is used to generate a mask for this attribute; (3) DreamSim features are extracted and cosine similarity is computed only over the masked region. For each method, the blended midpoint image is compared to both input images: the "Input 1" and "Input 2" columns report DreamSim similarity between the midpoint and each input, respectively, and the "Mean" column reports their average. Bold numbers denote the best-performing method.

Insights: Mixing DINO and CLIP features works best on medium difficulty and low difficulty cases, however on high difficulty cases, using CLIP without mixing DINO works better.

Figure 8. Ablation study on Flag Loss. **Insights:** without flag loss, the blending failed to capture the main attribute.

C. Path Finding Details

C.1. Proof of Vibe Space Analytical Minimum

This section provides the mathematical justification for the claim made in the main text: *linear interpolation in Vibe Space decodes to an approximate geodesic in the original feature manifold.* We show this by starting from the formulation of the multiscale inverse diffusion map, rewriting it in terms of the flag-space kernel, and then substituting the geometric alignment constraint learned during Vibe Space training. This results in a latent-space surrogate objective whose unique minimizer is exactly the linear interpolation between endpoints.

Notation Recap.

- $\mathbf{x} \in \mathbf{R}^{(HW)\times D}$: DINO feature tokens.
- f: encoder mapping tokens to Vibe Space.
- $\mathbf{z} \in \mathbf{R}^{(HW)\times d}$: Vibe Space embeddings, $\mathbf{z} = f(\mathbf{x})$.
- g: decoder mapping Vibe Space to CLIP feature space.
- $\mathbf{\Psi}^{1:m_k}(\mathbf{x}) \in \mathbf{R}^{(HW)\times m_k}$: first m_k diffusion-map eigenvectors.
- $\mathcal{M} = \{m_1, \ldots, m_k\}$: set of multiscale truncation levels.
- $S(\mathbf{\Psi}(\mathbf{x}))$: flag-space kernel

$$S_{ij}(\mathbf{\Psi}(\mathbf{x})) = \frac{1}{|\mathcal{M}|} \sum_{m_k \in \mathcal{M}} \mathbf{\Psi}^{1:m_k}(\mathbf{x}_i)\, \mathbf{\Psi}^{1:m_k}(\mathbf{x}_j)^\top.$$

- $\mathbf{z}_A, \mathbf{z}_B$: latent endpoints.
- $\alpha \in [0, 1]$: interpolation weights.
- $\gamma(\alpha)$: path in original manifold.
- $\mathbf{z}_\alpha$: latent interpolation: $\mathbf{z}_{\alpha,i} = (1-\alpha)\mathbf{z}_A + \alpha\, \mathbf{z}_B$.

Inverse Diffusion Map in Flag-Space Kernel Form Given interpolated diffusion map coordinates

$$\mathbf{\Psi}_t(\mathbf{x}_\alpha) = (1-\alpha)\mathbf{\Psi}_t(\mathbf{x}_A) + \alpha\, \mathbf{\Psi}_t(\mathbf{x}_B),$$

the multiscale inverse diffusion map seeks

$$\gamma(\alpha) = \arg\min_{\mathbf{x}^*} \frac{1}{|\mathcal{M}|} \sum_{m_k \in \mathcal{M}} \left\| \mathbf{\Psi}_t^{1:m_k}(\mathbf{x}^*) - \mathbf{\Psi}_t^{1:m_k}(\mathbf{x}_\alpha) \right\|_2^2. \tag{A.1}$$

Expand the squared norm

$$\begin{aligned} \left\| \mathbf{\Psi}_t^{1:m_k}(\mathbf{x}^*) - \mathbf{\Psi}_t^{1:m_k}(\mathbf{x}_\alpha) \right\|_2^2 &= \left\| \mathbf{\Psi}_t^{1:m_k}(\mathbf{x}^*) \right\|_2^2 \\ &\quad + \left\| \mathbf{\Psi}_t^{1:m_k}(\mathbf{x}_\alpha) \right\|_2^2 \\ &\quad - 2 \left\langle \mathbf{\Psi}_t^{1:m_k}(\mathbf{x}^*), \mathbf{\Psi}_t^{1:m_k}(\mathbf{x}_\alpha) \right\rangle. \end{aligned} \tag{3}$$

(A.2)

Summing (A.2) over all $m_k \in \mathcal{M}$ produces three corresponding sums:

- **Term 1:** $\frac{1}{|\mathcal{M}|} \sum_{m_k} \|\mathbf{\Psi}_t^{1:m_k}(\mathbf{x}^*)\|_2^2$ collects all inner products $\langle \mathbf{\Psi}_t^{1:m_k}(\mathbf{x}_i^*), \mathbf{\Psi}_t^{1:m_k}(\mathbf{x}_i^*) \rangle$ across scales. Across tokens, these are exactly the *diagonal entries* of the flag-space kernel $S(\mathbf{\Psi}(\mathbf{x}^*))$.
- **Term 2:** $\frac{1}{|\mathcal{M}|} \sum_{m_k} \|\mathbf{\Psi}_t^{1:m_k}(\mathbf{x}_\alpha)\|_2^2$ does not depend on $\mathbf{x}^*$ and is absorbed into a constant.
- **Term 3 (cross term):**

$$\frac{2}{|\mathcal{M}|} \sum_{m_k} \left\langle \mathbf{\Psi}_t^{1:m_k}(\mathbf{x}^*), \mathbf{\Psi}_t^{1:m_k}(\mathbf{x}_\alpha) \right\rangle.$$

Element-wise, each pair of tokens (i, j) contributes

$$\frac{1}{|\mathcal{M}|} \sum_{m_k} \mathbf{\Psi}^{1:m_k}(\mathbf{x}_i^*)\, \mathbf{\Psi}^{1:m_k}(\mathbf{x}_{\alpha,j})^\top,$$

which is precisely the (i, j) entry of the kernel $S(\mathbf{\Psi}(\mathbf{x}^*))$ matched to $S(\mathbf{\Psi}(\mathbf{x}_\alpha))$.

Putting the three terms together, the entire objective in (A.1) is the Frobenius norm between the two corresponding flag-space kernel matrices:

$$\gamma(\alpha) = \arg\min_{\mathbf{x}^*} \left\| S(\mathbf{\Psi}(\mathbf{x}^*)) - S(\mathbf{\Psi}(\mathbf{x}_\alpha)) \right\|_F^2. \tag{A.3}$$

Thus the inverse diffusion map reduces to matching multiscale flag-space kernels.

Vibe Space Approximation. Vibe Space training enforces the alignment between Gram matrix $\mathbf{z}\mathbf{z}^\top$ and flag-space kernel $S(\mathbf{\Psi}(\mathbf{x}))$

$$\mathbf{z}\mathbf{z}^\top \approx S(\mathbf{\Psi}(\mathbf{x})), \qquad \mathbf{z} = f(\mathbf{x}). \tag{A.4}$$

Let $\mathbf{z}^* = f(\mathbf{x}^*)$ and $\mathbf{z}_\alpha = (1-\alpha)\, \mathbf{z}_A + \alpha\, \mathbf{z}_B$. Substituting (A.4) into (A.3) gives the latent surrogate:

$$\boxed{\mathbf{z}^* = \arg\min_{\mathbf{z}} \left\| \mathbf{z}\mathbf{z}^\top - \mathbf{z}_\alpha \mathbf{z}_\alpha^\top \right\|_F^2}. \tag{A.5}$$

Because the Gram matrix $\mathbf{z}\mathbf{z}^\top$ matches $\mathbf{z}_\alpha \mathbf{z}_\alpha^\top$ uniquely when $\mathbf{z} = \mathbf{z}_\alpha$, the unique minimal is

$$\boxed{\mathbf{z}^* = \mathbf{z}_\alpha}. \tag{A.6}$$

Thus $\gamma(\alpha) \approx g(\mathbf{z}_\alpha)$, i.e., *linear interpolation in Vibe Space provides a closed-form approximation to the multiscale inverse-diffusion geodesic.*

C.2. Graph Laplacian and Diffusion Map

Affinity graph and Laplacian. Let $\{\mathbf{x}_i\}_{i=1}^N$ denote the DINO feature tokens used as graph nodes. We construct a weighted affinity graph with

$$\mathbf{W}_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{\sigma^2} \right), \qquad \mathbf{D}_{ii} = \sum_{j=1}^N \mathbf{W}_{ij}, \tag{A.7}$$

where $\sigma > 0$ is a bandwidth parameter. We set $\sigma^2 = \sum_d \mathrm{Var}(x_i^{(d)})$ to match the global feature variance, ensuring that the affinity kernel adapts to the scale of the DINO feature distribution.

The graph Laplacian is

$$\mathbf{L} = \mathbf{D} - \mathbf{W}. \tag{A.8}$$

The corresponding random-walk diffusion operator is the row-stochastic matrix

$$\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}, \tag{A.9}$$

where $\mathbf{P}_{ij}$ gives the one-step transition probability from node i to node j.

Nyström approximation. Constructing the full affinity matrix $\mathbf{W} \in \mathbb{R}^{N\times N}$ is $\mathcal{O}(N^2)$ in memory and compute, which is prohibitive for tens of thousands of DINO tokens per image. We therefore employ the Nyström approximation [16]:

- Sample a subset of $S \ll N$ anchor tokens.
- Form the subgraph affinities $\mathbf{W}_{SS}$ exactly.
- Compute cross-affinities $\mathbf{W}_{NS}$ between all nodes and anchors.
- Approximate the full kernel by

$$\widetilde{\mathbf{W}} = \mathbf{W}_{NS}\,\mathbf{W}_{SS}^{-1}\,\mathbf{W}_{NS}^{\top}. \tag{A.10}$$

This reduces the eigen-decomposition cost from $\mathcal{O}(N^3)$ to $\mathcal{O}(S^3)$ and the kernel construction cost from $\mathcal{O}(N^2)$ to $\mathcal{O}(NS)$. In practice, we use $S = 500$; the graph eigenvectors can be computed in milliseconds.

Generalized eigenproblem. The diffusion-map coordinates are obtained by solving

$$\mathbf{L}\,\mathbf{\Psi} = \lambda\,\mathbf{D}\,\mathbf{\Psi}, \tag{A.11}$$

which is equivalent to diagonalizing the diffusion operator:

$$\mathbf{P}\,\mathbf{\Psi} = (1-\lambda)\,\mathbf{\Psi}. \tag{A.12}$$

D. Vibe Space Details

D.1. Correspondence Matching

To ensure that the "vibe" transition preserves semantic structure (e.g., blending a head to another head rather than to a background element), we establish a semantic correspondence π between the input images. This corresponds to the 'Match' function in Algorithms 1 and 2. Since pixel-wise matching is computationally expensive and noisy, we operate on semantic regions derived from DINO features.

As shown in Fig. 9, we first segment each image into k distinct semantic regions using NCut [16]. Given the DINO feature tokens $\mathbf{x}^{\text{dino}}$, we construct a token-wise affinity graph and compute the leading eigenvectors of the graph Laplacian. We then discretize these eigenvectors into segmentation masks using a k-way clustering approach. This results in a set of masks $\{\text{Mask}_1, \ldots, \text{Mask}_k\}$ for each image, grouping visually and semantically similar patches.

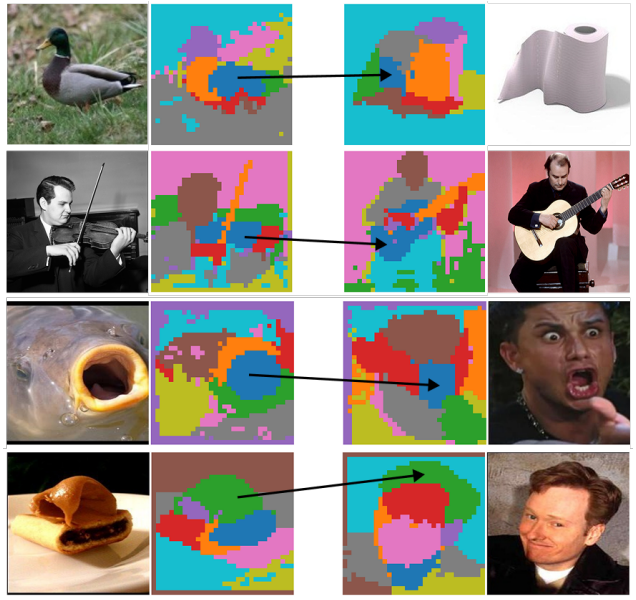


Figure 9. Examples of correspondence matching. We use NCut clustering [16] and Hungarian matching to compute correspondence between segments.

For each identified segment i, we compute a representative feature centroid $\mathbf{c}_i$. This is calculated by averaging the DINO feature embeddings of all patch tokens belonging to that segment:

$$\mathbf{c}_i = \frac{1}{|\text{Mask}_i|} \sum_{p\in\text{Mask}_i} \mathbf{x}_p^{\text{dino}}, \tag{4}$$

where p represents the token indices within the mask Mask_i.

To define the correspondence mapping π, we formulate the problem as a linear assignment task. We compute a cost matrix $C \in \mathbb{R}^{k\times k}$ representing the pairwise feature distances between the cluster centroids of the source image I_A and the target image I_B:

$$C_{ij} = \|\mathbf{c}_i^{(A)} - \mathbf{c}_j^{(B)}\|_2. \tag{5}$$

We apply the Hungarian algorithm to find the optimal bijection that minimizes the total feature distance between matched segments. This mapping π allows us to compute the vibe displacement $\Delta_{A\to B}$ relative to the corresponding semantic structures in the Vibe Space.

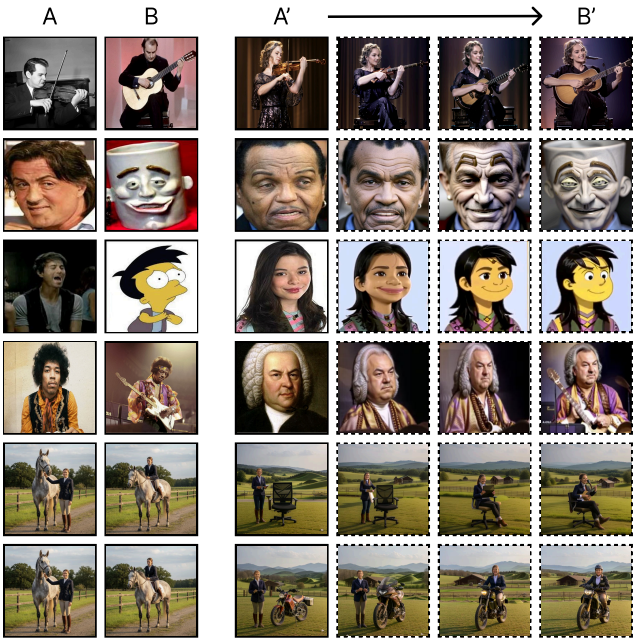


Figure 10. Vibe Analogy examples. Vibe Analogy can transfer the vibe of images $A \to B$ to another non-trivial but related image A' to generate B'. Examples of transferred vibes are human-object interaction and art styles between similar facial expressions.

Algorithm 1 Vibe Analogy

Input: Images $(I_A, I_B, I_{A'})$, image features $(\mathbf{x}^{\text{dino}}, \mathbf{x}^{\text{clip}})$
Output: Generated intermediate images $\{I_\alpha\}_{\alpha \in [0,1]}$

1: $\mathbf{W}_{ij} = \exp(-\frac{\|\mathbf{x}_i^{\text{dino}} - \mathbf{x}_j^{\text{dino}}\|^2}{\sigma^2})$, $\mathbf{D}_{ii} = \sum_j \mathbf{W}_{ij}$ ▷ Graph
2: $(\mathbf{D} - \mathbf{W})\mathbf{\Psi}(\mathbf{x}^{\text{dino}}) = \lambda \mathbf{D}\mathbf{\Psi}(\mathbf{x}^{\text{dino}})$ ▷ Graph Diffusion Map
3: $f, g \leftarrow \text{Train}(\mathbf{x}^{\text{clip}}, \mathbf{x}^{\text{dino}}, \mathbf{\Psi}(\mathbf{x}^{\text{dino}}))$ ▷ Train Vibe Space
4: $\mathbf{z}_A = f(\mathbf{x}_A^{\text{dino}})$; $\mathbf{z}_B = f(\mathbf{x}_B^{\text{dino}})$;
5: $\mathbf{z}_{A'} = f(\mathbf{x}_{A'}^{\text{dino}})$ ▷ Encode vibe
6: $\pi_{B \leftrightarrow A} \leftarrow \text{Match}(\mathbf{x}_A^{\text{dino}}, \mathbf{x}_B^{\text{dino}})$
7: $\pi_{A \leftrightarrow A'} \leftarrow \text{Match}(\mathbf{x}_A^{\text{dino}}, \mathbf{x}_{A'}^{\text{dino}})$ ▷ Cluster correspondence
8: $\mathbf{\Delta}_{A \to B} = \pi_{B \leftrightarrow A}(\mathbf{z}_B) - \mathbf{z}_A$
9: $\mathbf{\Delta}_{A' \to B'} = \pi_{A \leftrightarrow A'}(\mathbf{\Delta}_{A \to B})$ ▷ Path direction
10: **for** $\alpha \in [0, 1]$ **do**
11: $\quad \mathbf{z}_\alpha = \mathbf{z}_{A'} + \alpha\, \mathbf{\Delta}_{A' \to B'}$ ▷ Path interpolation
12: $\quad \mathbf{x}_\alpha^{\text{clip}} = g(\mathbf{z}_\alpha)$ ▷ Decode vibe
13: $\quad I_\alpha \leftarrow \text{IPAdapter}(\mathbf{x}_\alpha^{\text{clip}})$ ▷ Generate image
14: **end for**

D.2. Vibe Analogy

Vibe Analogy applies the semantic transformation observed between a reference pair (I_A, I_B) to a new subject $I_{A'}$, as shown in Fig. 10. The procedure is detailed in Algorithm 1.

We first compute the Vibe Space for all three images $I_A, I_B, I_{A'}$ simultaneously (Lines 1-5). We extract DINO features $\mathbf{x}^{\text{dino}}$, construct the joint affinity graph, and train the encoder-decoder pair (f, g) to map between the shared spectral manifold and CLIP space. We then encode all three inputs into the Vibe Space to obtain $\mathbf{z}_A, \mathbf{z}_B$, and $\mathbf{z}_{A'}$.

To transfer the "vibe" from the reference pair to the new subject, we must establish a correspondence chain (Lines 6-7). Using the matching procedure defined in Section D.1, we compute two distinct mappings: (1) $\pi_{A \leftrightarrow B}$ matches semantic segments in the reference source I_A to the reference target I_B. (2) $\pi_{A \leftrightarrow A'}$ matches semantic segments in the new subject $I_{A'}$ to the reference source I_A.

We define the "vibe" as the displacement field $\mathbf{\Delta}_{A \to B}$ between semantically corresponding regions in the reference pair. For a cluster centroid $\mathbf{c}_i^{(A)}$ and its match $\mathbf{c}_j^{(B)}$ (where $j = \pi_{A \leftrightarrow B}(i)$), the local displacement is $\mathbf{v}_i = \mathbf{c}_j^{(B)} - \mathbf{c}_i^{(A)}$.

To transfer this vibe to $I_{A'}$, we use the mapping $\pi_{A \leftrightarrow A'}$ to pull the appropriate displacement vector for each segment in A' (Lines 8-9). For a segment k in $I_{A'}$, if it maps to segment i in I_A (i.e., $i = \pi_{A \leftrightarrow A'}(k)$), we assign it the displacement $\mathbf{v}_i$. This constructs the target displacement field $\mathbf{\Delta}_{A' \to B'}$, ensuring that the specific semantic transformation (e.g., "turning a face into a flower") is applied to the correct anatomical regions of the new subject.

Finally, we generate the output image $I_{B'}$ by applying the transferred displacement field to the latent code of the new subject (Lines 10-13): $\mathbf{z}_\alpha = \mathbf{z}_{A'} + \alpha \mathbf{\Delta}_{A' \to B'}$. This latent path is decoded by g into CLIP space and rendered via IP-Adapter [17].

D.3. Negative Vibe Control

The Principle of Vibe Subtraction. Conceptually, our goal is to "subtract" the negative vibe from the positive vibe. In the geometric framework of Vibe Space, both positive and negative vibes can be represented as subspaces spanned by the eigenvectors of their respective graph Laplacians. The positive subspace, spanned by $\mathbf{\Psi}_{\text{pos}}$, contains the desired attributes, while the negative subspace, spanned by $\mathbf{\Psi}_{\text{neg}}$, contains the unwanted ones.

It is highly probable that these subspaces are not orthogonal; they share common components. For instance, if the positive vibe is "rotation" and "style change" (**??**), and the negative vibe is "style change," both subspaces will contain eigenvectors related to texture, color palette, and artistic rendering. To isolate "rotation," we must remove the "style change" components from the positive vibe's basis.

We achieve this by orthogonalizing the positive basis vectors against the negative basis vectors. This process effectively projects $\mathbf{\Psi}_{\text{pos}}$ onto a new basis, $\mathbf{\Psi}_{\text{filtered}}$, that is orthogonal to the subspace of $\mathbf{\Psi}_{\text{neg}}$. The mathematical formulation for this operation in the flag-space hierarchy is:

$$\mathbf{\Psi}_{\text{filtered}} = \mathbf{\Psi}_{\text{pos}} - \beta \cdot \mathbf{\Psi}_{\text{neg}} \left((\mathbf{\Psi}_{\text{neg}})^\top \mathbf{\Psi}_{\text{pos}} \right). \quad (6)$$

Here, we assume the eigenvectors forming the columns of $\mathbf{\Psi}_{\text{neg}}$ are orthonormal. The term $\mathbf{\Psi}_{\text{neg}}((\mathbf{\Psi}_{\text{neg}})^\top \mathbf{\Psi}_{\text{pos}})$ thus

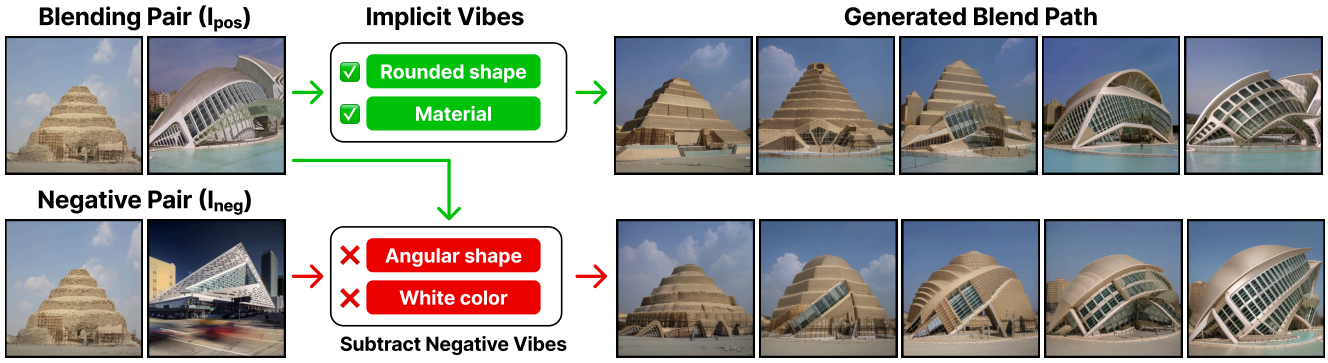


Figure 11. Negative vibe control. Vibe attributes are implicitly extracted by Vibe Space. The blending pair defines desired vibes (rounded shape and material). The negative pair defines vibes to suppress (angular shape and white color). Blending without negative examples may transfer more attributes than desired. Subtracting the negative vibes, we better preserve the rounded shape and tan color of the building.

represents the orthogonal projection of the positive eigenvectors onto the subspace spanned by the negative eigenvectors. By subtracting this projection, we are left with the component of the positive vibe that is orthogonal to the negative vibe. The hyperparameter β controls the strength of this subtraction: $\beta = 1$ performs full orthogonalization, while $\beta > 1$ can be used to actively push the resulting vibe away from the negative attributes.

Learning a Filtered Vibe Space. To incorporate negative vibes into our framework, we modify the training objective of the Vibe Space autoencoder. Instead of training the encoder f to match the geometry of the positive flag-space kernel $S(\mathbf{\Psi}(\mathbf{x}_{\text{pos}}))$, we train it to match the kernel of the *filtered* basis, $S(\mathbf{\Psi}_{\text{filtered}})$.

The flag-space encoder loss from Eq. (4) becomes:

$$\mathcal{L}_{\text{flag\_enc}}(f) = \|\mathbf{z}\mathbf{z}^\top - S(\mathbf{\Psi}_{\text{filtered}})\|_F^2, \tag{7}$$

where $\mathbf{z} = f(\mathbf{x}_{\text{pos}})$. The encoder f is trained only on the positive exemplars $\mathbf{x}_{\text{pos}}$, but the target geometry $S(\mathbf{\Psi}_{\text{filtered}})$ is now computed from both the positive ($\mathbf{\Psi}_{\text{pos}}$) and negative ($\mathbf{\Psi}_{\text{neg}}$) eigenvector sets. Fig. 11 shows an example of a vibe blending path with and without negative exemplars.

D.4. Vibe Visualization by Gradient

To understand which specific visual attributes drive the geometric alignment in Vibe Space, we propose a gradient-based visualization method. Since our framework relies on DINO features for semantic correspondence and graph construction, we identify the specific feature channels within the high-dimensional DINO embedding that are most responsible for a specific semantic region or "vibe".

We leverage the differentiability of the spectral clustering process. Let $\mathbf{X} \in \mathbb{R}^{N \times C}$ denote the extracted DINO feature tokens for an images, where N is the number of patches and C is the feature dimension. We compute the generalized eigenvectors $\mathbf{\Psi}$ of the graph Laplacian $\mathbf{L}$.

Algorithm 2 Vibe Blending with Negative Vibe Control

Input: Pos images (I_A, I_B) with pos image features $(\mathbf{x}_{\text{pos}}^{\text{dino}}, \mathbf{x}_{\text{pos}}^{\text{clip}})$, Neg images (I_A, I_C) with neg image features $(\mathbf{x}_{\text{neg}}^{\text{dino}})$
Output: Generated intermediate images $\{I_\alpha\}_{\alpha \in [0,1]}$

$\mathbf{W}_{\text{pos}}, \mathbf{D}_{\text{pos}} \leftarrow \text{ComputeAffinity}(\mathbf{x}_{\text{pos}}^{\text{dino}})$ ▷ Pos graph
$\mathbf{W}_{\text{neg}}, \mathbf{D}_{\text{neg}} \leftarrow \text{ComputeAffinity}(\mathbf{x}_{\text{neg}}^{\text{dino}})$ ▷ Neg graph
$\mathbf{\Psi}_{\text{pos}} \leftarrow \text{GraphDiffusionMap}(\mathbf{W}_{\text{pos}}, \mathbf{D}_{\text{pos}})$ ▷ Pos eigvecs
$\mathbf{\Psi}_{\text{neg}} \leftarrow \text{GraphDiffusionMap}(\mathbf{W}_{\text{neg}}, \mathbf{D}_{\text{neg}})$ ▷ Neg eigvecs
$\mathbf{\Psi}_{\text{filtered}} \leftarrow \mathbf{\Psi}_{\text{pos}} - \beta \cdot \mathbf{\Psi}_{\text{neg}}(\mathbf{\Psi}_{\text{neg}}^\top \mathbf{\Psi}_{\text{pos}})$ ▷ Orthogonalize
$f, g \leftarrow \text{Train}(\mathbf{x}_{\text{pos}}^{\text{clip}}, \mathbf{x}_{\text{pos}}^{\text{dino}}, \mathbf{\Psi}_{\text{filtered}})$ ▷ Train Vibe Space
$\mathbf{z}_A \leftarrow f(\mathbf{x}_A^{\text{dino}})$; $\mathbf{z}_B \leftarrow f(\mathbf{x}_B^{\text{dino}})$ ▷ Encode pos vibes
$\pi \leftarrow \text{Match}(\mathbf{x}_A^{\text{dino}}, \mathbf{x}_B^{\text{dino}})$ ▷ Cluster correspondence
$\Delta_{A \to B} \leftarrow \pi(\mathbf{z}_B) - \mathbf{z}_A$ ▷ Path direction in filtered space
for $\alpha \in [0, 1]$ **do**
  $\mathbf{z}_\alpha \leftarrow \mathbf{z}_A + \alpha \cdot \Delta_{A \to B}$ ▷ Path interpolation
  $\mathbf{x}_\alpha^{\text{clip}} \leftarrow g(\mathbf{z}_\alpha)$ ▷ Decode vibe
  $I_\alpha \leftarrow \text{IPAdapter}(\mathbf{x}_\alpha^{\text{clip}})$ ▷ Generate image
end for

To visualize the features corresponding to a specific semantic cluster k (e.g., the foreground object), we first obtain the discretized cluster indicator $\mathbf{y}_k \in \{0, 1\}^N$ from the eigenvectors via k-way Ncut [18]. We the define a maximization objective $\mathcal{L}_{\text{vis}}$ to identify the feature channels contributing to this cluster:

$$\mathcal{L}_{\text{vis}} = -\frac{1}{|\text{Mask}_k|} \sum_{i \in \text{Mask}_k} |\mathbf{\Psi}_{i,k}|, \tag{8}$$

where Mask_k is the set of indices belonging to cluster k, and $\mathbf{\Psi}_{\cdot,k}$ is the eigenvector corresponding to that cluster. We treat the input features $\mathbf{X}$ as a learnable parameter with respect to this loss. By computing the gradient $\nabla_{\mathbf{X}} \mathcal{L}_{\text{vis}}$ via backpropagation through the eigendecomposition, we obtain a saliency map $\mathbf{G} \in \mathbb{R}^{N \times C}$:

$$\mathbf{G} = \nabla_{\mathbf{X}} \mathcal{L}_{\text{vis}}. \tag{9}$$

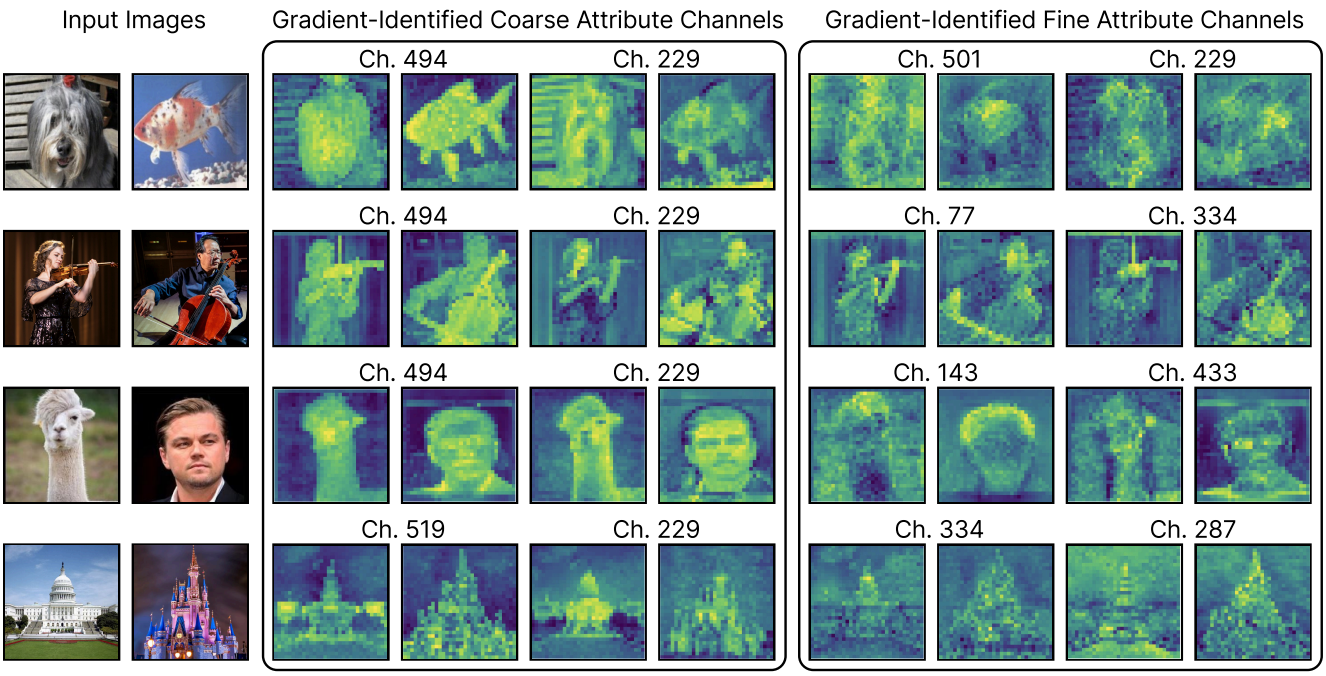


Figure 12. Heatmaps illustrating the top-ranked DINO channels corresponding to the coarse and fine attributes shared by the input images. For example, in the second row, the coarse attributes may include the body pose and object identity, while the fine-grained attributes include the hand-object interaction.

We then aggregate the gradients across the spatial dimensions of the cluster mask to score each feature channel c:

$$s_c = \frac{1}{|\text{Mask}_k|} \sum_{i \in \text{Mask}_k} \mathbf{G}_{i,c}. \tag{10}$$

The channels with the highest scores s_c represent the specific DINO feature dimensions encoding the attribute. We visualize these top-ranked channels as heatmaps as shown in Fig. 12.

D.5. Vibe Blending: Selecting the Optimal Blend Weight α via Dip in CLIP Consistency

Selecting a single representative frame from the generated continuous blend path is non-trivial. A naive choice, such as the midpoint $\alpha = 0.5$, often fails to capture the most compelling hybrid. This stems from the inherent asymmetry of our correspondence-based interpolation: we compute a semantic displacement field $\mathbf{\Delta}_{A \to B}$ and add it to the token clusters of the source image I_A. Since the transformation is anchored to the source image's semantic structure, the conceptual transition is not perfectly linear with respect to α. Consequently, the point of optimal blending may shift away from $\alpha = 0.5$, requiring an adaptive selection strategy.

To address this, we propose an automated procedure to select the optimal α by identifying the point of greatest conceptual transition along the blend path. Our intuition is that the most challenging point for the generative model to render often represents the most novel synthesis of attributes.

We formalize this procedure in Algorithm 3, which measures the consistency between an "ideal" path decoded from Vibe Space and a "realized" path obtained by re-encoding

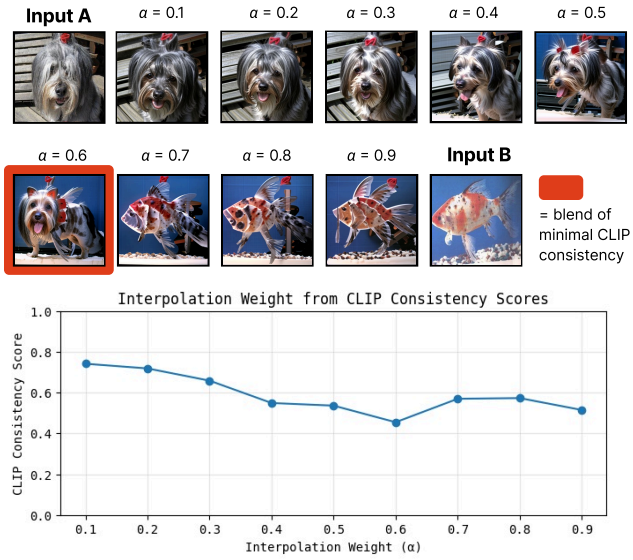


Figure 13. Our method generates a continuous path of blends for any interpolation weight $\alpha \in [0, 1]$. We propose an automated procedure to select the best α by identifying the point of greatest conceptual transition along the blend path. Specifically, we compute a *consistency score* to measure the difference between an "ideal" path decoded from Vibe Space and a "realized" path obtained by re-encoding the generated images. In this example, the optimal weight $\alpha^* = 0.6$ as determined by the "dip" in the score. Qualitatively, we observe that the α^* obtained via this algorithm achieves the best creative blend of the inputs, and we use this approach for our evaluation.

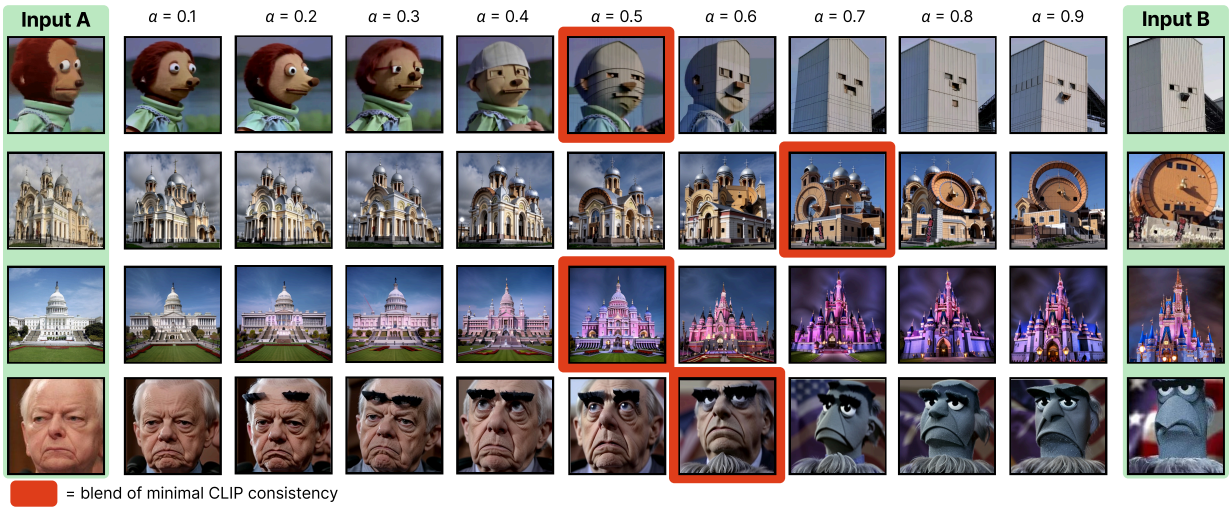


Figure 14. Examples of blended images selected via the optimal blend weight α^* determined by our "dip" in CLIP consistency score, described in Fig. 13. We observe that the "dip" in consistency score generally results in the best blend of the relevant attributes in both input images.

the generated images. The point of minimum consistency, or the "dip" in the score, identifies our target α^*.

Algorithm 3 Optimal Blend Selection (α^*)

Input: Vibe decoder g, vibe endpoints $\mathbf{z}_A, \mathbf{z}_B$, source image I_A, candidate steps $A_{\text{steps}} \subset [0, 1]$
Output: Optimal interpolation weight α^*

1: $S_{\text{list}} \leftarrow []$ ▷ Initialize list to store scores
2: **for** $\alpha \in A_{\text{steps}}$ **do**
3: $\mathbf{z}_\alpha \leftarrow (1-\alpha)\mathbf{z}_A + \alpha\mathbf{z}_B$ ▷ Interpolate in Vibe Space
4: $\mathbf{x}_\alpha^{\text{ideal}} \leftarrow g(\mathbf{z}_\alpha)$ ▷ **Ideal Path**: Decode to CLIP space
5: $I_\alpha \leftarrow \text{IPAdapter}(\mathbf{x}_\alpha^{\text{ideal}})$ ▷ Generate image
6: $\mathbf{x}_\alpha^{\text{realized}} \leftarrow \text{CLIP}(I_\alpha)$ ▷ **Realized Path**: Re-encode image
7: $\pi_\alpha, \mathcal{C}_A^{\text{down}}, \mathcal{C}_\alpha^{\text{down}} \leftarrow \text{ComputeCorrespondence}(I_A, I_\alpha)$ ▷ **Correspondence**: See Appendix D.1
8: $S_\alpha \leftarrow \text{ComputeConsistency}(\mathbf{x}_\alpha^{\text{ideal}}, \mathbf{x}_\alpha^{\text{realized}}, \pi_\alpha, \mathcal{C}_A^{\text{down}})$ ▷ **Score**: See Eq. 11
9: $S_{\text{list}}.append(S_\alpha)$
10: **end for**
11: $i^* \leftarrow \arg\min(S_{\text{list}})$ ▷ **Identify Dip**: Find index of min score
12: $\alpha^* \leftarrow A_{\text{steps}}[i^*]$
13: **return** α^*

The core of this procedure involves generating and comparing two conceptual paths for each candidate α. As shown in Algorithm 3 (lines 3-6), the **ideal path** is formed by decoding the Vibe Space interpolation directly into CLIP space, representing the intended manifold-respecting trajectory. The **realized path** is formed by generating a pixel-space image with the IP-Adapter and then re-encoding it with CLIP, representing what the model actually produces.

A direct token-wise comparison of these paths is brittle to minor spatial shifts. Therefore, we establish a robust semantic correspondence between the source image I_A and each generated image I_α (line 7). We use DINO features to perform spectral clustering on both images, segmenting them into meaningful regions. The Hungarian algorithm is then used to find an optimal matching π_α between the cluster centers. With the correspondence established, we compute the consistency score S_α (line 8). For each semantic region c in the source image, we calculate its mean CLIP feature vector along both the ideal path, μ_c^{ideal}, and the corresponding realized path, $\mu_c^{realized}$:

$$\mu_c^{ideal}(\alpha) = \text{mean}(\mathbf{x}_\alpha^{\text{ideal}}[c])$$
$$\mu_c^{realized}(\alpha) = \text{mean}(\mathbf{x}_\alpha^{\text{realized}}[\pi_\alpha(c)])$$

The overall score is the average cosine similarity over all matched cluster pairs:

$$S(\alpha) = \frac{1}{|\mathcal{C}_A|} \sum_{c \in \mathcal{C}_A} \cos\left(\mu_c^{ideal}(\alpha), \mu_c^{realized}(\alpha)\right) \quad (11)$$

where $\mathcal{C}_A$ is the set of source clusters and $\mathbf{x}[c]$ denotes the CLIP tokens belonging to cluster c.

Finally, the optimal interpolation weight α^* is selected as the one that minimizes this consistency score (lines 10-11). This "dip" in consistency marks the point of greatest semantic tension and, frequently, the most creative and compelling synthesis of the two concepts (Fig. 13). Fig. 14 shows examples of selecting α^* across a blend trajectory.

D.6. Vibe Blending: Training with Many Images

Vibe Space can be trained with more than two images, and using additional related images helps in identifying the main attributes. Algorithm 4 outlines the steps for training Vibe Space with multiple additional images. The key difference between **??** and Algorithm 4 is that the latter uses more images to solve the graph diffusion map $\mathbf{\Psi}$ and train the Vibe Space. However, the blending process is still executed using two images.

Fig. 15 presents qualitative examples comparing training with only two images to training with additional images. The middle blend, when additional images are included, successfully captures the "glass window" vibe when trained with extra images featuring glass windows.

Algorithm 4 Vibe Blending with extra images

Input: Multiple images $(I_A, I_B, \dots)$, image features for all images $(\mathbf{x}^{\text{dino}}, \mathbf{x}^{\text{clip}})$
Output: Generated intermediate images $\{I_\alpha\}_{\alpha\in[0,1]}$

1: $\mathbf{W}_{ij} = \exp(-\frac{\|\mathbf{x}_i^{\text{dino}} - \mathbf{x}_j^{\text{dino}}\|^2}{\sigma^2})$, $\mathbf{D}_{ii} = \sum_j \mathbf{W}_{ij}$ ▷ Graph
2: $(\mathbf{D} - \mathbf{W})\mathbf{\Psi}(\mathbf{x}^{\text{dino}}) = \lambda \mathbf{D}\mathbf{\Psi}(\mathbf{x}^{\text{dino}})$ ▷ Graph Diffusion Map
3: $f, g \leftarrow \text{Train}(\mathbf{x}^{\text{clip}}, \mathbf{x}^{\text{dino}}, \mathbf{\Psi}(\mathbf{x}^{\text{dino}}))$ ▷ Train Vibe Space (all images)
4: $\mathbf{z}_A = f(\mathbf{x}_A^{\text{dino}})$; $\mathbf{z}_B = f(\mathbf{x}_B^{\text{dino}})$ ▷ Encode vibe (two images)
5: $\pi \leftarrow \text{Match}(\mathbf{x}_A^{\text{dino}}, \mathbf{x}_B^{\text{dino}})$ ▷ Cluster correspondence
6: $\mathbf{\Delta}_{A\to B} = \pi(\mathbf{z}_B) - \mathbf{z}_A$ ▷ Path direction
7: **for** $\alpha \in [0, 1]$ **do**
8: $\quad \mathbf{z}_\alpha = \mathbf{z}_A + \alpha\, \mathbf{\Delta}_{A\to B}$ ▷ Path interpolation
9: $\quad \mathbf{x}_\alpha^{\text{clip}} = g(\mathbf{z}_\alpha)$ ▷ Decode vibe
10: $\quad I_\alpha \leftarrow \text{IPAdapter}(\mathbf{x}_\alpha^{\text{clip}})$ ▷ Generate image
11: **end for**

D.7. Extrapolating Vibe Blending Paths

Since our framework constructs a locally linear manifold, we are not limited to interpolation within the convex hull of the input images ($\alpha \in [0, 1]$). We can perform *extrapolation* by setting the interpolation weight $\alpha > 1$, effectively continuing along the geodesic path defined by the displacement vector $\mathbf{\Delta}_{A\to B}$. We observe that this can yield an exaggerating effect, where specific visual characteristics are amplified beyond their presence in the target image I_B (Fig. 16).

D.8. N-Image Blending

We extend our framework to blend N images $\{I_1, \dots, I_N\}$ simultaneously, enabling applications such as barycentric interpolation within a triangle of concepts (Fig. 17).

Unlike pairwise blending, establishing consistent multi-way correspondence is non-trivial. We designate one input as the base image (I_{base}), which serves as the structural anchor for the blend. We then compute $N - 1$ pairwise correspondence mappings, aligning the semantic clusters of every other image I_k to the clusters of the base image:

$$\pi_{k\to\text{base}} = \text{Match}(\mathbf{x}_{\text{base}}^{\text{dino}}, \mathbf{x}_k^{\text{dino}})$$

where $\pi_{k\to\text{base}}$ maps a cluster index in the base image to the corresponding cluster index in image k.

With semantic correspondences established across all inputs, we generalize the path interpolation logic to a multi-way vector summation. Let $\{\alpha_1, \dots, \alpha_N\}$ be the scalar interpolation weights for the N images. We generate the blended latent code $\mathbf{z}_{\text{blend}}$ by starting with the base embedding and accumulating the weighted semantic displacements from all other images.

For a specific semantic region (cluster) i in the base image, let $\mathbf{c}_i^{(\text{base})}$ be its centroid in Vibe Space. For every other image k, the centroid of the corresponding matched cluster is $\mathbf{c}_j^{(k)}$, where $j = \pi_{k\leftrightarrow\text{base}}(i)$. The blending equation for tokens belonging to region i is:

$$\mathbf{z}_{\text{blend}}[i] = \mathbf{z}_{\text{base}}[i] + \sum_{k=1}^{N} \alpha_k \left(\mathbf{c}^{(k)}_{\pi_{k\leftrightarrow\text{base}}(i)} - \mathbf{c}_i^{(\text{base})} \right)$$

This formulation computes a weighted average of the "vibe" displacements relative to the base structure. If $\alpha_k = 1$ (and all others 0), the term simplifies to shifting the base embedding by the exact vector difference between the base and target k, approximating a full morph to image k. The resulting $\mathbf{z}_{\text{blend}}$ is decoded into CLIP space and rendered via the IP-Adapter [17].

E. Implementation Details

E.1. Training Procedure

We train the encoder–decoder (f, g) for each image pair independently using a lightweight MLP architecture.

- **Optimizer:** Adam (learning rate = 0.001).
- **Batch size:** 2 images.
- **Total iterations:** 1000 steps.
- **MLP layers:** 4.
- **MLP hidden dimension:** 256.
- **Number of Parameters:** 0.72M.

Vibe Blending running time is under 30 seconds on a RTX4090 GPU. In Algorithm 4 lines 1-2, solving the graph diffusion map for 2-5 images (512 input image resolution, 1024 DINO tokens each image) only takes milliseconds with the help of Nyström approximation [16]. In Algorithm 4 line 3, training encoder-decoder MLPs for 1000 steps takes 15 seconds. In Algorithm 4 line 5, computing correspondence matching takes milliseconds. In Algorithm 4 line 10, image generation with Stable Diffusion takes 2 seconds per image.

Memory usage is under 1GB for graph diffusion map and training the Vibe Space. Memory usage is 12GB after loading the Stable Diffusion model.

E.2. Loss Balancing

Our full training objective is:

$$\begin{aligned}\mathcal{L} = \lambda_{\text{flag\_enc}}\mathcal{L}_{\text{flag\_enc}} + \lambda_{\text{flag\_dec}}\mathcal{L}_{\text{flag\_dec}} \\ + \lambda_{\text{sample}}\mathcal{L}_{\text{sample}} + \lambda_{\text{recon}}\mathcal{L}_{\text{recon}}\end{aligned} \tag{12}$$

The loss terms include:

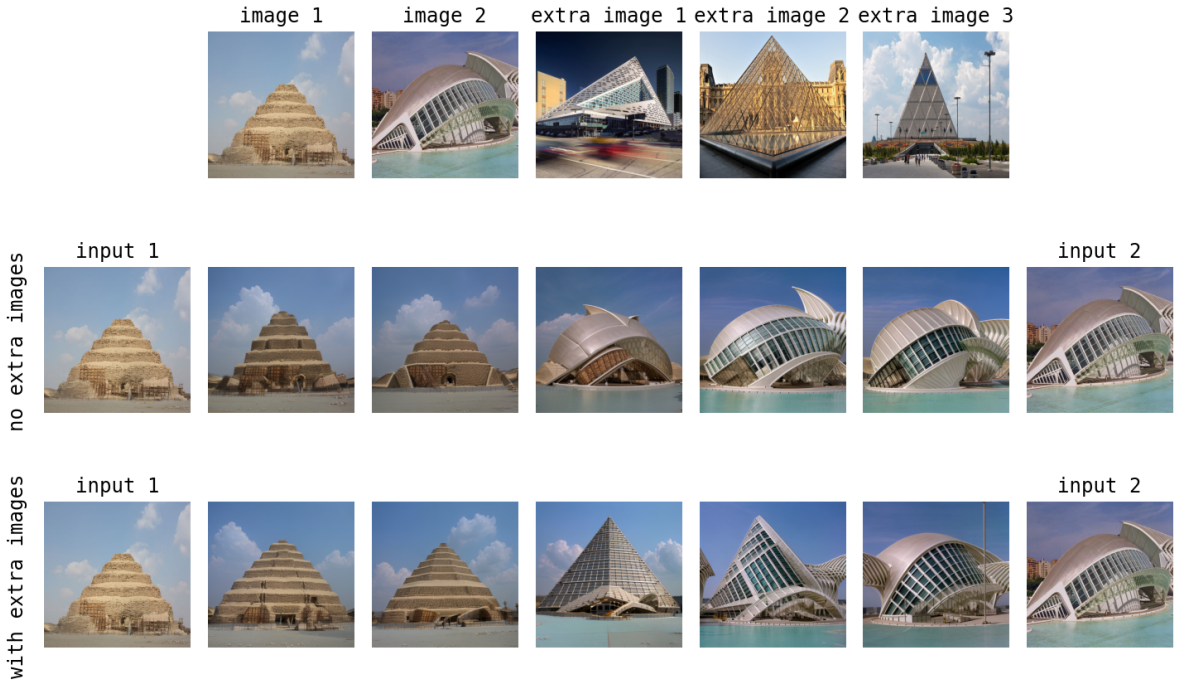


Figure 15. Vibe Blending with extra image training. We compare training Vibe Space with only two images (middle row) and with five images (bottom row), input images and extra images are in the first row. Training with extra glass window images helps capture the "glass window" vibe, resulting in a glass texture pyramid in the middle-blend image.

- $\mathcal{L}_{\text{flag\_enc}}$, $\mathcal{L}_{\text{flag\_dec}}$: multiscale flag-space geometry preservation for the encoder f and decoder g respectively.

$$\mathcal{L}_{\text{flag\_enc}}(f) = \left\|\mathbf{z}\mathbf{z}^\top - \mathbf{S}(\mathbf{\Psi}(\mathbf{x}))\right\|_2^2, \qquad \mathbf{z} = f(\mathbf{x}),$$
$$\mathcal{L}_{\text{flag\_dec}}(f, g) = \left\|\mathbf{z}\mathbf{z}^\top - \mathbf{S}(\mathbf{\Psi}(g(\mathbf{z})))\right\|_2^2,$$

- $\mathcal{L}_{\text{sample}}$: flag-space geometry consistency in extrapolated space, for the decoder.

$$\mathcal{L}_{\text{sample}}(g) = \left\|\mathbf{z}_{\text{sample}}\mathbf{z}_{\text{sample}}^\top - \mathbf{S}\big(\mathbf{\Psi}(g(\mathbf{z}_{\text{sample}}))\big)\right\|_2^2.$$

- $\mathcal{L}_{\text{recon}}$: Reconstruct CLIP features.

$$\mathcal{L}_{\text{recon}}(f, g) = \|\mathbf{x}^{\text{clip}} - g(f(\mathbf{x}^{\text{dino}}))\|_2^2.$$

Loss weight balancing:

$$\lambda_{\text{flag\_enc}} = 1, \quad \lambda_{\text{flag\_dec}} = 0.01, \quad \lambda_{\text{sample}} = 0.01, \quad \lambda_{\text{recon}} = 1,$$

We observe that $\mathcal{L}_{\text{flag\_dec}}$ and $\mathcal{L}_{\text{sample}}$ can overwhelm the CLIP feature reconstruction loss $\mathcal{L}_{\text{recon}}$, this leads to poor image generation quality. Thus, we down-weighted $\mathcal{L}_{\text{flag\_dec}}$ and $\mathcal{L}_{\text{sample}}$ by a factor of 0.01. Additionally, $\mathcal{L}_{\text{sample}}$ can cause training instability (NaN) during the warm-up periods, thus we only apply $\mathcal{L}_{\text{sample}}$ after first 500 steps of training iterations.

F. Additional Evaluation Details

F.1. LLMs for Generating and Evaluating Blends

Vibe Blending involves reasoning about the relevant visual attributes shared between images, and then coherently fusing these attributes. Accordingly, we compare our method with multimodal LLMs that exhibit strong visual reasoning capabilities, such as Gemini 2.5 Flash Image [3] and GPT Image 1 [10]. To promote their reasoning for this task, we use chain-of-thought prompting [14] inspired by Peng et al. [11]. Specifically, we ask Gemini to (1) identify the main objects in each input image, (2) recognize the main visual attributes that are similar between the images, and (3) generate an image that creatively blends these attributes, as shown in Fig. 18. Since the OpenAI API does not provide a public model that supports multi-turn conversations with both image and text inputs—and image and text outputs—we combine these steps into a single prompt for GPT [10]. The full prompts for generating blends are presented below. The parentheses indicate (inputs $\rightarrow$ outputs) where I_A and I_B are input images, I_{blend} is the blended output, T denotes text, and $\varnothing$ denotes no additional inputs besides the prompt.

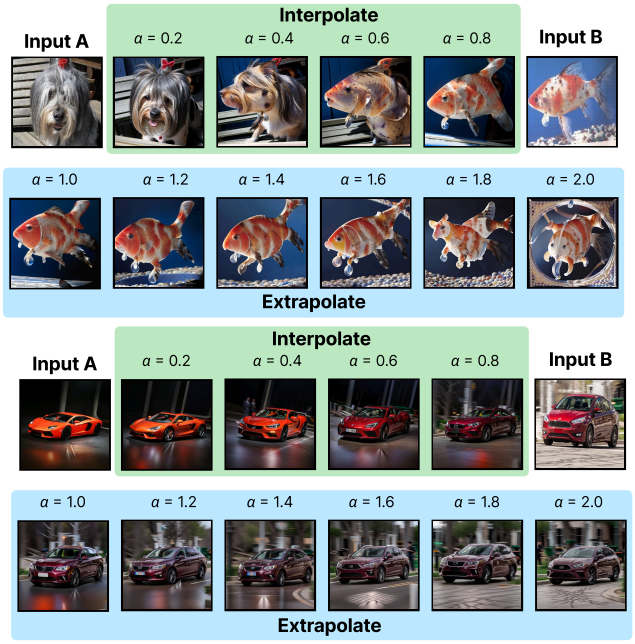


Figure 16. Paths computed by Vibe Blending can be extrapolated to exaggerate attributes. **Top:** Extrapolating the path from a dog to a fish emphasizes the body shape of the fish and some elements of the background. However, extrapolation behavior is not well controlled at higher weights. **Bottom:** Extrapolating from an orange sports car to a red sedan continues the color shift, resulting in further darkening.

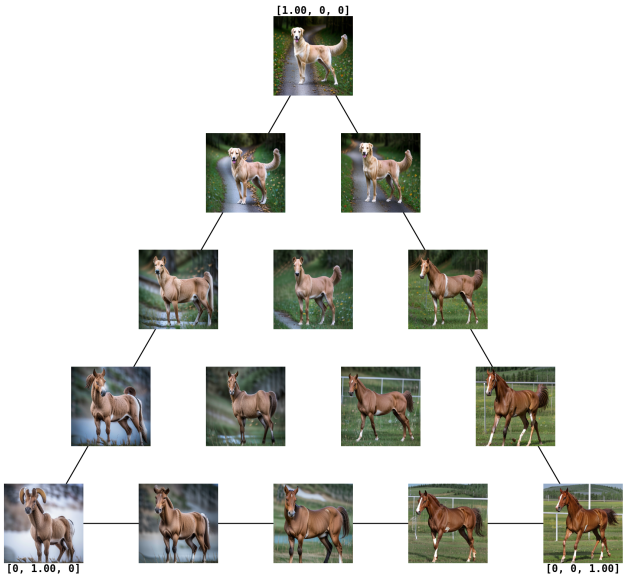


Figure 17. N-Image Blending extends Vibe Blending to an arbitrary number of images. In this visualization, we show 3-Image Blending between a dog, ram, and horse. The top image (dog) is used as the base image to anchor correspondence matching and blending.

Gemini 2.5 Flash Image Blending Prompt

($I_A \rightarrow$ T) Identify the object in this first input image.

($I_B \rightarrow$ T) Identify the object in this second input image.

($\varnothing \rightarrow$ T) I want to blend the objects in both images so their main visual attributes merge into a single, coherent hybrid. First, identify the main visual attributes that are shared by the input images, such as facial expression, hair style, or object shape. Additionally, interpret what the blend of objects might be intended to convey.

($\varnothing \rightarrow I_{blend}$) Based on all previous turns, generate an image that blends the main attributes in the input images. The output should not be a collage, overlay, or style transfer. You may modify any image region.

GPT Image 1 Blending Prompt

($I_A, I_B \rightarrow I_{blend}$) Generate an image that blends the objects in the input images so their main visual attributes merge into a single, coherent hybrid. First, identify the main visual attributes that are shared by the input images, such as facial expression, hair style, or object shape. Second, blend the main attributes in the images. The output should not be a collage, overlay, or style transfer. You may modify any image region.

We also employ an LLM to judge the output blends from different methods using a similar step-by-step reasoning approach. We ask GPT-5 [9] to (1) identify the main objects in each input image; (2) recognize the main visual attributes that are similar between the images; (3) for each output, assess how well it blends the main attributes; and (4) choose the best blended output and provide reasoning. An abbreviated example chat is shown in Fig. 19. The full prompt for evaluating blends is provided below.

LLM Judge Evaluation Prompt

($I_A \rightarrow$ T) Identify the object in this first input image.

($I_B \rightarrow$ T) Identify the object in this second input image.

($\varnothing \rightarrow$ T) I want to blend the objects in both images so their visual attributes merge into a single, coherent hybrid concept. First, identify the main visual attributes that are similar between the inputs, such as facial expression, hair style, or object shape. Additionally, interpret what the blend of objects might be intended to convey.

Inputs	Gemini Outputs
 "Identify the object in this first input image."	"The object in the first input image is Richard Nixon. He was the 37th President of the United States."
 "Identify the object in this second input image."	"The object in the second input image is an eggplant with an unusual growth, making it appear phallic."
"I want to blend the objects in both images so their main visual attributes merge into a single, coherent hybrid. First, identify the main visual attributes that are shared by the input images, such as facial expression, hair style, or object shape. Additionally, interpret what the blend of objects might be intended to convey."	Main visual attributes from the first image: • Strong, somewhat stern facial expression • Distinctive hair part and receding hairline • Formal attire • Overall human form Main visual attributes from the second image: • Purple, glossy skin • Elongated, somewhat pear-like main body shape • Distinct, protruding appendage • Green stem/calyx Shared attribute for blending: • Object Shape/Form: The most significant "shared" attribute, however, comes from the prominent protruding element on the eggplant. The blend of these objects might be intended to convey several things, often with a satirical or critical undertone: 1. Caricature or lampooning 2. Symbolism of absurdity or corruption 3. Visual Metaphor for "Nose" or "Lie" 4. Discomfort or Grotesqueness
"Based on all previous turns, generate an image that blends the main attributes in the input images. The output should not be a collage, overlay, or style transfer. You may modify any image region."	

Figure 18. Example chat with Gemini [3] to generate a creative blend.

LLM Judge Evaluation Prompt (continued)

($\varnothing \rightarrow$ T) Assess how well this first output image blends the main attributes in both input images.

($\varnothing \rightarrow$ T) Assess how well this second output image blends the main attributes in both input images.

($\varnothing \rightarrow$ T) Assess how well this third output image blends the main attributes in both input images.

($\varnothing \rightarrow$ T) Assess how well this fourth output image blends the main attributes in both input images.

($\varnothing \rightarrow$ T) Based on all previous turns, which output image (1, 2, 3, or 4) best blends the main attributes in both input images? Do not pick any output that does not perform blending, such as a collage, overlay, style transfer, or juxtaposition of the inputs. Do not pick any output that preserves the exact structure or identity of the inputs. Do not pick any output that is a part-level composition, where parts of one input are simply attached to parts of the other input. Provide your reasoning.

Fig. 20 illustrates examples where human raters and the LLM judge agree and disagree on the best blended output. We observe that common LLM failure cases include identifying extra irrelevant attributes such as body pose, color palette, and image composition.

F.2. User Study Details

Human-Rated Creative Potential vs. Blend Difficulty. We conducted a user study with ten human participants to compare 44 image pairs sourced from the Totally Looks Like dataset [13]. Fig. 22 shows the prompt to users and examples with high rater agreement. Participants were presented with two image pairs side-by-side and asked to select which of the two pairs exhibited a higher Creative Potential and which exhibited a higher Blend Difficulty. They performed a total of 110 side-by-side comparisons, ensuring each of the 44 image pairs was included in 5 unique comparisons. We then fit a Bradley-Terry model [2] to these comparison ratings to produce a robust ranking of Creative Potential and Blend Difficulty across image pairs. Finally, we used these continuous Bradley-Terry scores to categorize the image pairs into low, medium, and high bins, as depicted in Fig. 21.

Human Preference of Creative Blends. We conducted another user study to assess the quality of blended outputs generated by different methods using input image pairs from Totally Looks Like [13]. Fig. 23 (a) displays the prompt given to participants, who answered two subques-

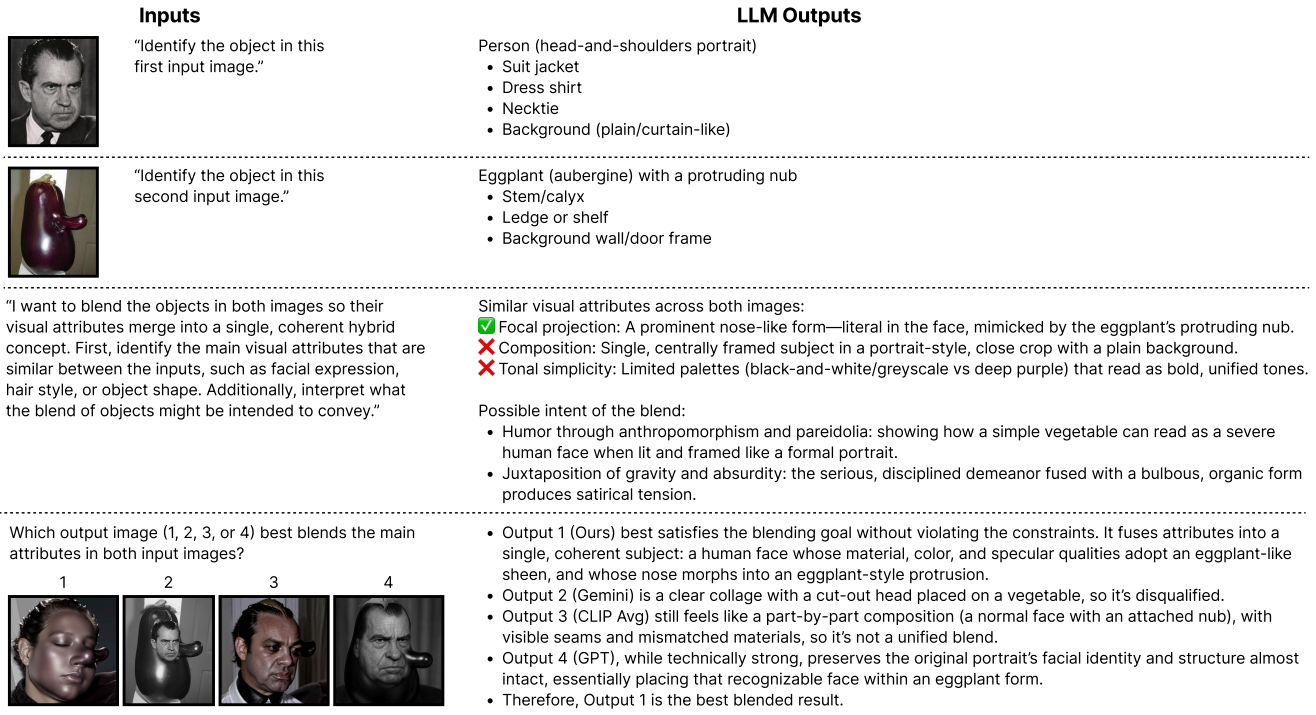


Figure 19. Example chat with the LLM judge [9] to select the best blended output.

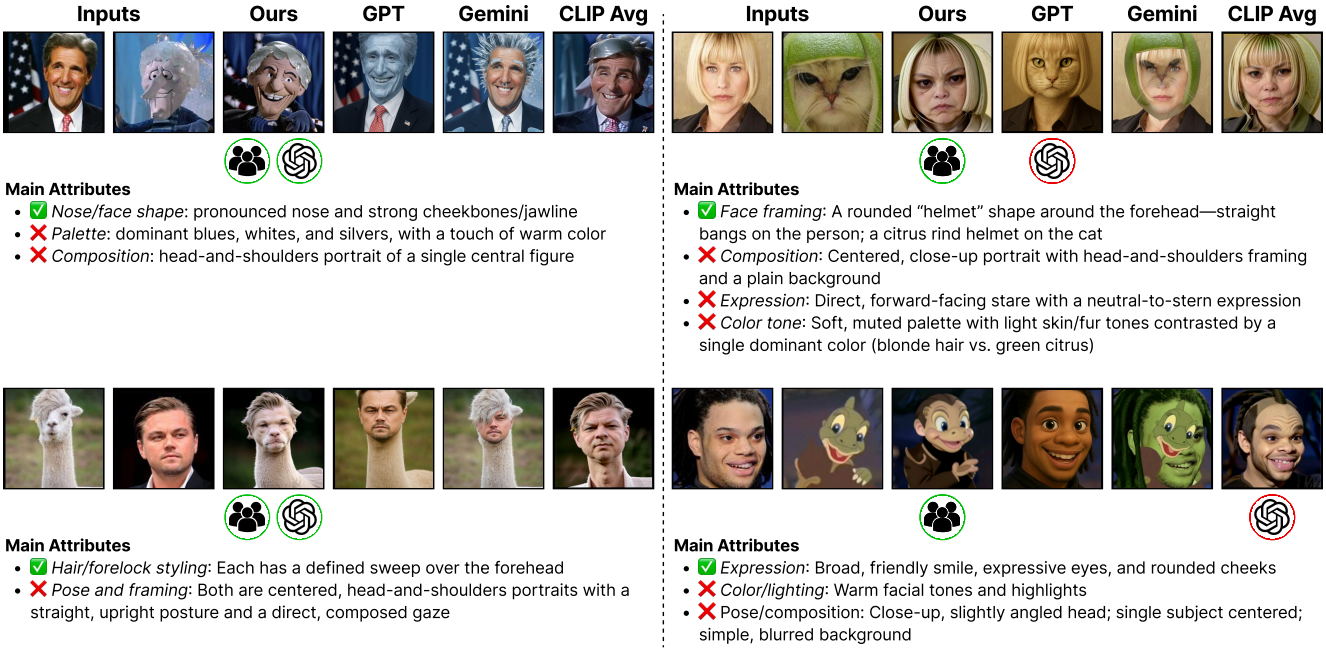


Figure 20. Examples where human raters and the LLM judge agree (**left**) and disagree (**right**) on the best blended output.

tions. First, users identified the main attributes shared by the images (Fig. 23 (b)). Then, users ranked the outputs based on how well they coherently and creatively blended those main attributes (Fig. 23 (c)). This two-step process encourages users to actively consider the relevant attributes for blending before making their selection. We determined the final ranking for each example via majority vote based on first-place votes, followed by second-place votes.

To encourage participants to focus on blending coherence and creativity, instead of other irrelevant characteristics like image quality, we utilized the following enhancement prompt with Gemini 2.5 Flash Image [3] to standard-

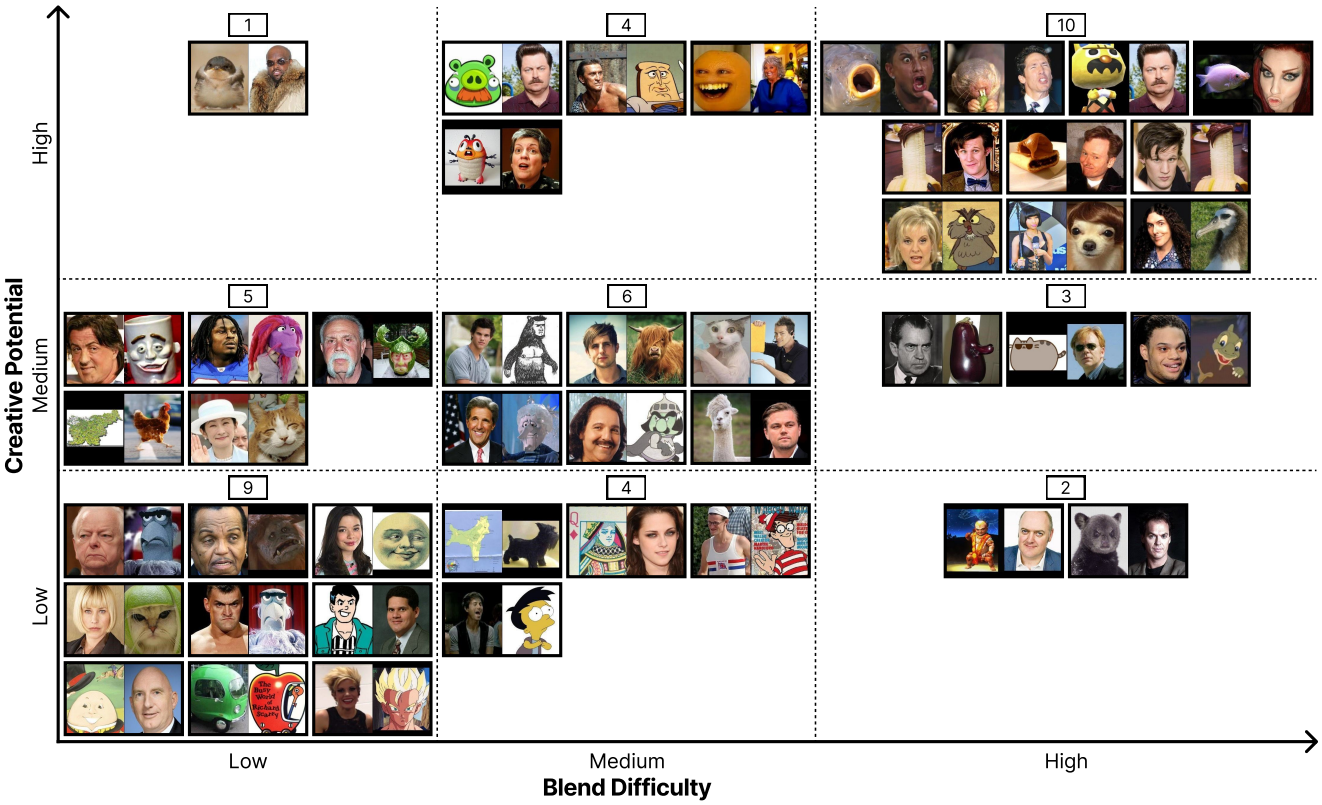


Figure 21. Extension of **??**. To gauge human perceptions of creativity, we ask raters to compare image pairs along two axes: *Creative Potential* refers to how interesting a blend might be, and *Blend Difficulty* indicates how challenging it is to form a coherent blend. Image pairs with higher Blend Difficulty tend to have higher Creative Potential and are often more conceptually different. Numbers in each cell indicate the number of examples.

ize the image quality of our blended output to match that of the outputs from models like Gemini and GPT [10].

Enhancement Prompt

Enhance this image to a high-resolution, professional quality while preserving all details and textures. Improve sharpness, smooth noise, and remove text and image artifacts for a clean, realistic, and well-balanced look.

F.3. How Does Creativity Relate to Diversity?

	Totally Looks Like		Architecture	
Method	CLIP	DreamSim	CLIP	DreamSim
CLIP Avg	0.079	0.096	0.067	0.112
Gemini [3]	0.189	0.305	0.129	0.257
GPT [10]	0.121	0.193	0.088	0.177
Ours	**0.223**	**0.339**	**0.150**	**0.291**

Table 5. We generate multiple blends for each input image pair and measure the mean pairwise diversity of the output images using CLIP [12] and DreamSim [5]. Higher diversity is better. Our method produces more diverse blends across multiple trials, in addition to more creative blends as shown in **??**.

While Creative Potential and Blend Difficulty describe how a blend could be formed, creativity can also be evaluated in the outputs. We examine one complementary metric, output diversity, echoing Hofstadter's insight that "making variations on a theme is the crux of creativity" [7]. We quantify output diversity by generating multiple blends per input image pair and computing pairwise perceptual distances using DreamSim [5], denoted by F. The diversity V for one example is:

$$V = \frac{1}{\binom{n}{2}} \sum_{i<j} \text{dist}(F(I_i), F(I_j)) \,. \tag{13}$$

where I_i and I_j are different outputs from the same input, $\text{dist}(\cdot,\cdot)$ denotes DreamSim distance, and $n = 3$. Higher V indicates greater output diversity, and we report the mean V across all examples. We also implement a comparable measure of output diversity using CLIP image similarity [12].

Shown in Table 5, our method generates the most varied blends, in addition to the best creative blends according to humans and the LLM in **??**. Interestingly, Gemini creates the second-most diverse outputs, even though it lags behind CLIP Avg and GPT in terms of blend quality.

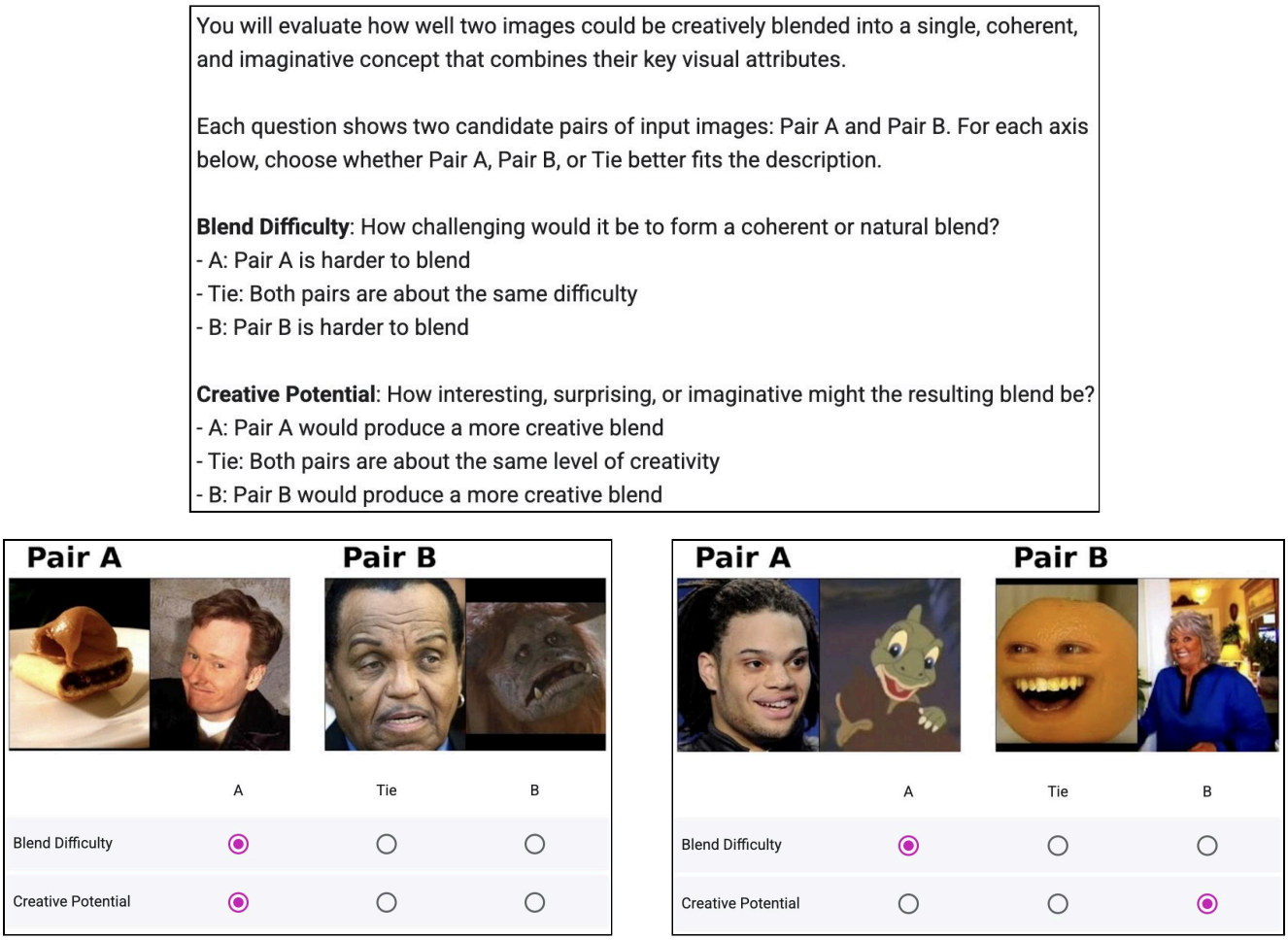


Figure 22. User study for evaluating the Creative Potential and Blend Difficulty of image pairs. **Top:** The prompt shown to users. **Bottom:** Two examples with high rater agreement. Participants select which image pair has higher Creative Potential and higher Blend Difficulty.

F.4. Curated Datasets

The Totally Looks Like dataset [13] contains image pairs that humans judge as visually similar. To curate a subset suitable for our evaluation, we focus on image pairs depicting distinct concepts while ensuring high image quality. We first automatically exclude pairs with overly similar content by removing those with both CLIP image similarity [12] above 0.65 and DreamSim distance [5] below 0.65. A human annotator then filters out any remaining pairs containing text or low-quality images. This process yields 44 high-quality and semantically distinct image pairs.

Additionally, we curate 300 pairs of architectural design images. We begin by collecting images spanning diverse architectural styles and architects from several existing datasets [1, 4, 15]. We then form random pairs such that the two images in each pair come from different architectural styles and different architects, and we recruit a human annotator to verify the quality of each image pair.

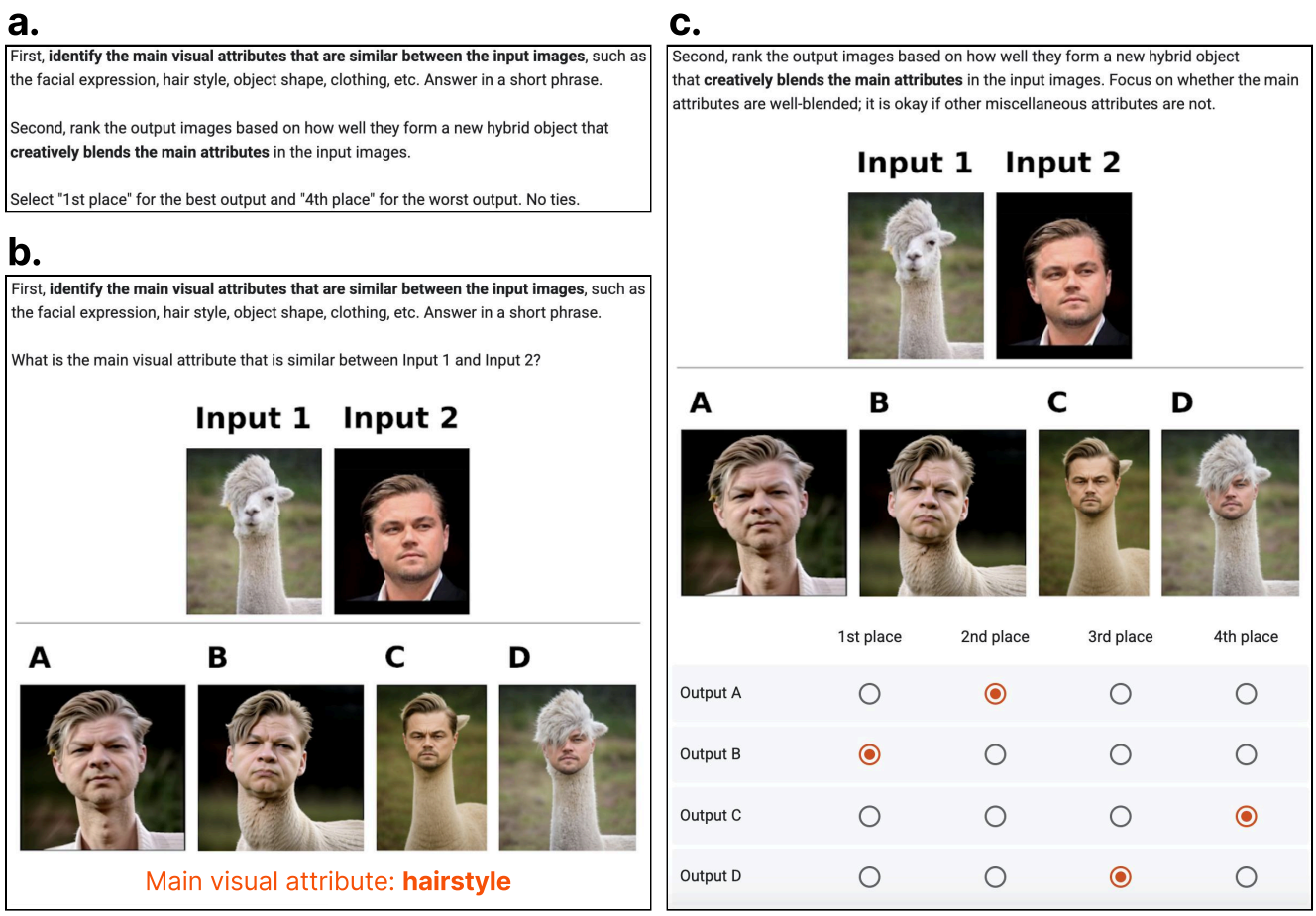


Figure 23. User study for assessing human preference of creative blends. **(a.)** The prompt shown to users. **(b.)** First, we ask participants to describe the main visual attributes shared by the two input images. **(c.)** Second, raters rank the blended output images from different methods based on how well they blend the main attributes.

G. Failure Cases

Unrelated Images Vibe Blending assumes that there is a shared attribute between two images. Our methods will fail to identify the vibe when the input images are not related. Examples in Fig. 24. Since two unrelated images are in disconnected manifolds, the spectral coordinates of the two images are not continuous and have gaps, linearly traverse the spectral coordinates will fall into gaps.

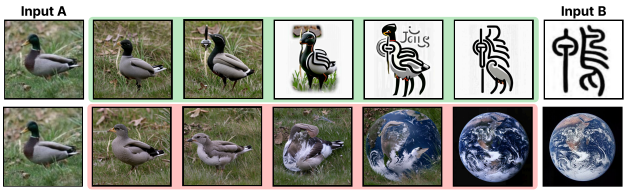


Figure 24. Vibe Blending for in-the-wild images. One failure case of our methods is that the two images does not share a clear vibe.

Entangled Vibes A limitation of negative vibe blending arises when desired and undesired attributes are entangled within the feature space. The orthogonalization process assumes that the positive and negative vibes can be separated into distinct, albeit non-orthogonal, subspaces. Fig. 25 illustrates a failure example where the vibe of "style change" is entangled with "color change", which prevents the entangled vibe from being filtered out.

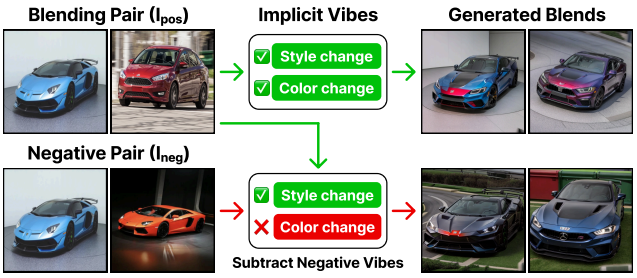


Figure 25. Negative vibe failure case. The positive inputs capture both a style change based on the types of car and a color change. The negative inputs intend to capture only color change. However, the attributes of style and color are entangled, making them difficult to separate with negative examples.

Extrapolation Uncertainty While effective in certain cases, extrapolating Vibe Blending does not always produce images with a meaningful exaggeration of relevant attributes (Fig. 26).

Correspondence Failure. Vibe Space relies on unsupervised region-level correspondence matching between DINO token clusters (see Section D.1). This correspondence is essential for concept-level blending, as it determines which semantic regions in I_A and I_B should be merged. However, because the matching is unsupervised, it is not always reliable. When correspondence is incorrect (vary by random seeds), the resulting blends can degrade significantly, often producing incoherent or mismatched object-part combinations (Fig. 27).

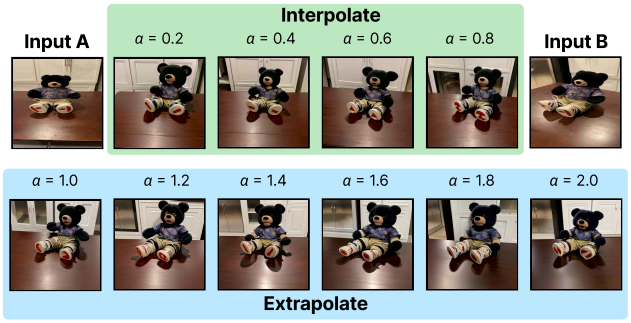


Figure 26. Failure case of Vibe Blending extrapolation. Extrapolating beyond $\alpha > 1$ does not produce further rotation of the object in the input images.

Reconstruction Failure. Our method depends on IP-Adapter [17] to generate images conditioned on dense CLIP features. However, IP-Adapter is not always reliable outside the training distribution of Stable Diffusion. To isolate this limitation, we perform the following diagnostic: (1) extract dense CLIP features from an input image; (2) perform no blending and do not train Vibe Space; (3) feed the extracted features directly into IP-Adapter to reconstruct the input image. As shown in Fig. 28, IP-Adapter can consistently reconstruct in-distribution images (e.g., human faces) across random seeds, but fails on out-of-distribution (OOD) inputs, producing unstable and inconsistent reconstructions.

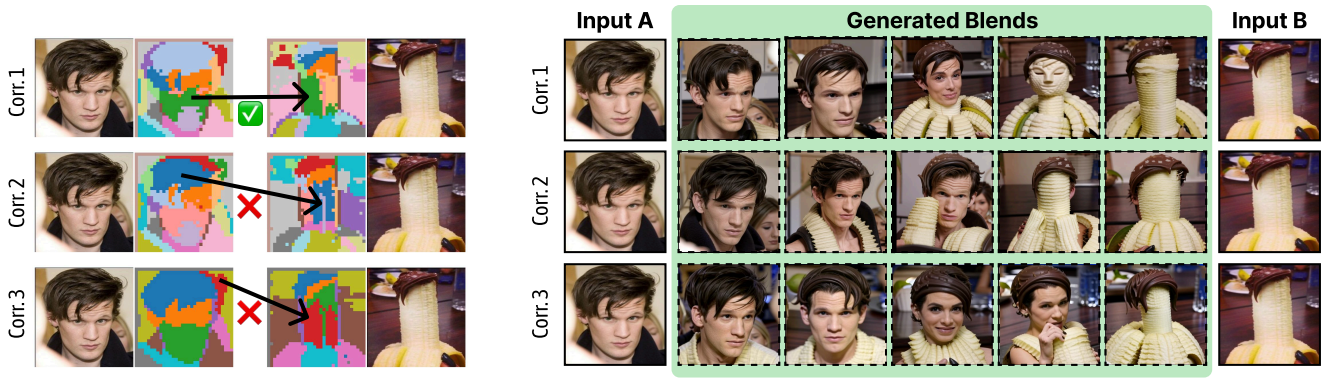


Figure 27. Failure cases of unsupervised correspondence. Our method depends on correspondence matching to identify which semantic regions should be blended. Different random seeds lead to different clusterings and therefore different correspondence maps. Good correspondence (top row) yields clean and semantically meaningful blends, whereas poor correspondence (middle and bottom rows) leads to low-quality blends with incorrect part-level mixing.

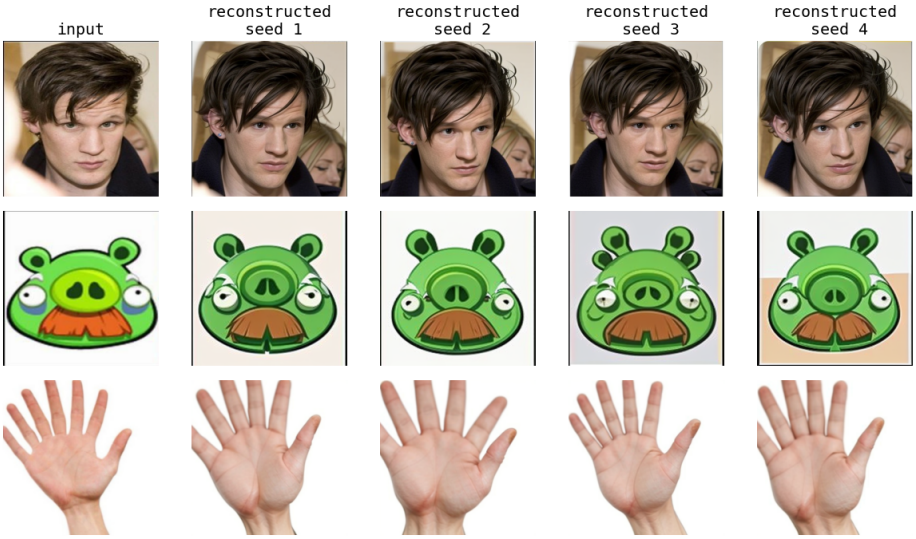


Figure 28. Failure cases of IP-Adapter reconstruction. Left column is input images; right four columns are reconstructed images from 4 random seeds. IP-Adapter reliably reconstructs images that lie within the Stable Diffusion training distribution (top row), producing consistent outputs across random seeds. In contrast, for out-of-distribution inputs (middle and bottom rows), IP-Adapter consistently fails to reproduce the original image, illustrating a fundamental limitation of SD-based decoders when applied to concept-level blending tasks.

References

[1] Galaxy Architects. Architects Dataset. https://www.kaggle.com/datasets/galaxyarchitects/architects-dataset, 2023. Accessed: 2025-02-11. 24

[2] Ralph A Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952. 21

[3] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025. 1, 3, 19, 21, 22, 23

[4] Dumitrux. Architectural Styles. https://www.kaggle.com/datasets/dumitrux/architectural-styles-dataset, 2020. Accessed: 2025-02-11. 24

[5] Stephanie Fu, Netanel Tamir, Shobhita Sundaram, Lucy Chai, Richard Zhang, Tali Dekel, and Phillip Isola. Dreamsim: Learning new dimensions of human visual similarity using synthetic data. *arXiv preprint arXiv:2306.09344*, 2023. 1, 23, 24

[6] Qiyuan He, Jinghao Wang, Ziwei Liu, and Angela Yao. Aid: Attention interpolation of text-to-image diffusion. *arXiv preprint arXiv:2403.17924*, 2024. 1, 2, 3, 4, 5, 6, 7

[7] Douglas R Hofstadter. *Metamagical themas: Questing for the essence of mind and pattern*. Basic books, 2008. 23

[8] Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. Lisa: Reasoning segmentation via large language model. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9579–9589, 2024. 1, 3, 9, 10

[9] OpenAI. GPT-5. https://platform.openai.com/docs/models/gpt-5, 2024. Accessed: 2025-02-11. 20, 22

[10] OpenAI. GPT Image 1. https://platform.openai.com/docs/models/gpt-image-1, 2024. Accessed: 2025-02-11. 1, 3, 19, 23

[11] Yongqian Peng, Yuxi Ma, Mengmeng Wang, Yuxuan Wang, Yizhou Wang, Chi Zhang, Yixin Zhu, and Zilong Zheng. Probing and inducing combinational creativity in vision-language models. *arXiv preprint arXiv:2504.13120*, 2025. 19

[12] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021. 23, 24

[13] Amir Rosenfeld, Markus D Solbach, and John K Tsotsos. Totally looks like-how humans compare, compared to machines. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1961–1964, 2018. 3, 21, 24

[14] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022. 19

[15] Zhe Xu, Dacheng Tao, Ya Zhang, Junjie Wu, and Ah Chung Tsoi. Architectural style classification using multinomial latent logistic regression. In *European conference on computer vision*, pages 600–615. Springer, 2014. 24

[16] Huzheng Yang. Nyström Normalized Cuts PyTorch (ncut-pytorch). Accessed: 2025-11-13. 13, 18

[17] Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. *arXiv preprint arXiv:2308.06721*, 2023. 14, 18, 26

[18] Yu and Shi. Multiclass spectral clustering. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 313–319 vol.1, 2003. 15

[19] Qingtao Yu, Jaskirat Singh, Zhaoyuan Yang, Peter Henry Tu, Jing Zhang, Hongdong Li, Richard Hartley, and Dylan Campbell. Probability density geodesics in image diffusion latent space. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 27989–27998, 2025. 1, 3, 4, 5, 6, 7

[20] Kaiwen Zhang, Yifan Zhou, Xudong Xu, Bo Dai, and Xingang Pan. Diffmorpher: Unleashing the capability of diffusion models for image morphing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7912–7921, 2024. 1, 3, 4, 5, 6, 7