PANTOMIME: IMITATING CAMERA CHANGE BY IMAGE ANALOGY

Anonymous authors
Paper under double-blind review

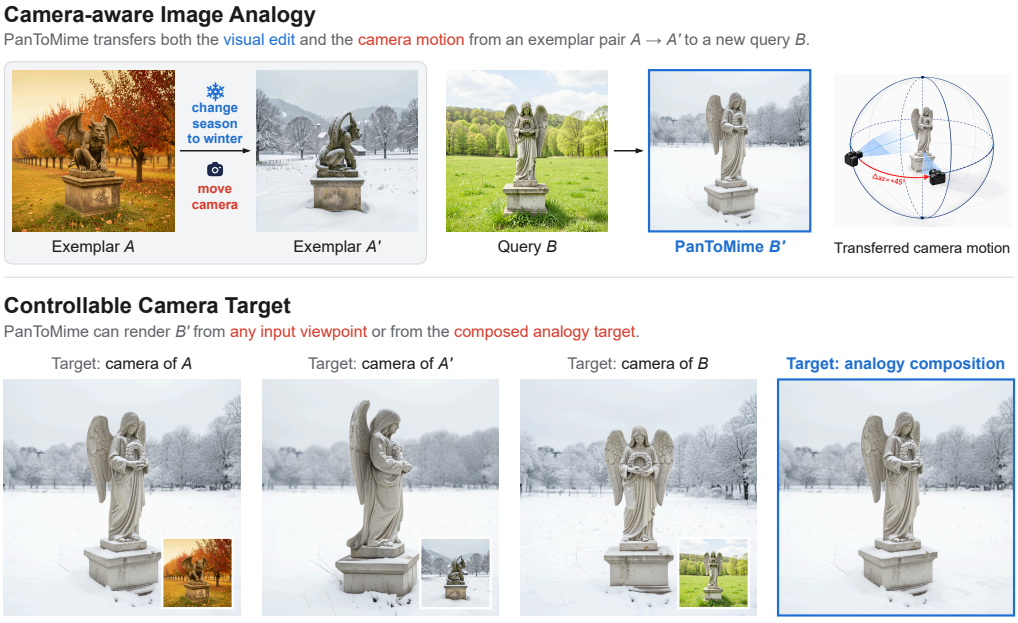

Figure 1: Standard image analogy methods cannot transfer the camera transition demonstrated by the exemplar pair $(A \rightarrow A')$ to the query image (Top). PanToMime captures this transition natively while also allowing users to specify any desired target viewpoint (Bottom).

ABSTRACT

Image analogy provides a natural way to edit images by example: instead of describing a transformation, the user simply demonstrates it. However, when the exemplar also contains a change in camera viewpoint, the desired transformation becomes ambiguous. A user may want to transfer the demonstrated camera movement, match the final exemplar view, or preserve the query view while applying only the visual edit. We introduce PanToMime, a camera-aware image analogy method that understands and controls camera changes while transferring the demonstrated visual transformation. To support this setting, we introduce **PanToMime-Bench**, containing paired visual edits under controlled camera changes, together with **Render2Real**, a geometry-grounded pipeline for generating realistic multi-view training data. Experiments show that existing image-analogy methods can reproduce visual edits but struggle with camera changes, particularly when the exemplar and query start from different viewpoints. PanToMime reduces camera orientation error from $68.20°$ for the second-best method to $30.69°$, while retaining competitive edit fidelity with an Edit score of 9.03. Our results also suggest that image analogy can provide an effective interface for camera understanding and pose transfer.

1 INTRODUCTION

Showing is often easier than telling. An image analogy $A : A' :: B : B'$ (Hertzmann et al., 2001a) asks a model to infer the transformation demonstrated by an exemplar pair (A, A') and apply it to a new image B. Recent visual in-context editors make this a natural interface for image editing (Gu et al., 2024; Chen et al., 2025; Gong et al., 2025b): instead of describing an edit through text, masks, or parameters, the user simply shows the desired transformation. A single exemplar can

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

jointly specify changes in appearance, material, object presence, subject pose, composition, and other visual properties that would otherwise require several separate instructions.

An important ambiguity arises when the exemplar also contains a change in camera viewpoint. Consider an exemplar in which a subject changes appearance while the camera moves from a frontal to a side view. There is no single correct way to transfer this transformation to B. The user may want B' to match the final viewpoint of A'. Alternatively, they may want to transfer the *camera movement* demonstrated from A to A' relative to the starting viewpoint of B. Or they may want to keep the original viewpoint of B and transfer only the visual edit, ignoring the camera change altogether. All three are meaningful image analogies, but they produce different results. In other words, to mimic (or *mime*) the full transformation, we may also need to *pan* the camera accordingly.

This ambiguity makes camera understanding and control important for image analogy. To reproduce the camera motion, the model needs to understand how the camera changes from A to A', not just the final view in A'. This change should then be applied to the starting view of B. For example, if the camera moves by 45 degrees from A to A', the same 45-degree change should be applied to B, rather than simply copying the view of A'. At the same time, the camera change should be separate from the visual edit. A user may want to transfer the camera change, match the final view of A', or keep the original view of B while applying only the visual edit. Therefore, camera should be an explicit and controllable part of image analogy.

We introduce PanToMime, a camera-aware image analogy framework that separates *what changes* from *how the scene is viewed* (see Fig. 1). Our key intuition is simple: the exemplar pair should provide the visual transformation, while camera pose should be explicitly understood and controlled. Given $(A \rightarrow A')$ and B, our method estimates the camera states of the observed images and uses them to determine the desired camera for B' (Fig. 1 (Top)). A rectified-flow transformer learns the visual transformation from $(A \rightarrow A')$, while per-frame pose tokens provide explicit camera conditioning. This separation allows the same visual analogy to be realized under different camera interpretations without requiring camera parameters from the user (Fig. 1 (Bottom)).

Training such a model requires data where both the visual edit and the camera change are known and can be controlled independently. Existing datasets do not provide this supervision. Multi-view datasets provide the same subject from controlled viewpoints but do not contain paired visual edits, while image-editing datasets provide paired edits but lack controlled views. We therefore construct **PanToMime-Bench**, a dataset designed specifically for camera-aware image analogy. Since directly generating multiple edited views with existing image editors leads to identity drift, inconsistent edits, and background changes, we instead use a geometry-grounded pipeline. We reconstruct each subject as a textured 3D proxy and render it from controlled viewpoints, then introduce **Render2Real** to convert these renders into realistic images using a photographic reference. This pipeline produces 5,803 curated views across 448 subjects and 15,169 training analogy quadruples.

Our experiments show that existing image-analogy methods can often reproduce the visual edit, but struggle to capture the demonstrated camera change, especially when A and B start from different viewpoints. We evaluate PanToMime on 500 held-out analogies against eight existing methods. PanToMime reduces camera orientation error to $30.7°$, compared with 68.2–$78.0°$ for all baselines, and achieves the highest fidelity to the target subject and viewpoint, measured by DINOv3 object similarity (0.780) and aligned mask IoU (0.763). It also remains competitive in transferring the demonstrated visual edit, scoring 9.03 versus 9.35 for the strongest commercial model.

Our contributions are fourfold. First, we formulate camera-aware image analogy, making viewpoint an explicit and controllable part of the transformation and showing its *potential* for *camera understanding* and *pose transfer*. Second, we introduce **PanToMime-Bench** and **Render2Real** for constructing paired edits with controlled camera changes. Third, we introduce PanToMime, which separates visual transformation from camera conditioning and transfers the demonstrated camera relation to the query. Finally, we show that existing image-analogy methods struggle with camera changes, while our method improves both analogy quality and camera accuracy.

2 Related Work

Image analogy and in-context editing. Image analogy (Hertzmann et al., 2001b) asks for B' such that $A : A' :: B : B'$. Early learned approaches operated in embedding space (Reed et al.,

2015), while later diffusion methods recovered the exemplar transition through inversion or shared attention (Šubrtová et al., 2023; Alaluf et al., 2024). Visual prompting cast analogy as completing an image grid (Bar et al., 2022; Wang et al., 2023; Sun et al., 2023; Gu et al., 2024), leading to recent in-context editors such as Edit Transfer (Chen et al., 2025), VisualCloze (Li et al., 2025), RelationAdapter (Gong et al., 2025a), LoRWeB (Manor et al., 2026), VIRAL (Li et al., 2026), and SIA (Alimohammadi et al., 2026). General-purpose multi-reference editors can also be prompted similarly (Labs et al., 2025; Batifol et al., 2025; Wu et al., 2025; Google DeepMind, 2026; OpenAI, 2026). These methods primarily transfer changes directly demonstrated in the exemplar, such as appearance, material, or object edits. Reed et al. (Reed et al., 2015) also considered out-of-plane rotation, but on synthetic car renderings sampled from known camera angles. Our setting instead asks an analogy model to recover and transfer camera change from uncalibrated images.

Camera-controlled generation. A large body of work generates images from a specified camera. Object-centric models synthesize novel views from a given relative pose (Liu et al., 2023), while multi-view models generate images at known viewpoints (Tang et al., 2023; Shi et al., 2024). Camera-controlled video models similarly condition generation on prescribed camera trajectories (Wang et al., 2024c; He et al., 2025). Across these settings, the target camera is provided as input. Image analogy presents the opposite problem: only (A, A', B) are observed, and the target camera of B' must itself be inferred. It depends on both the camera change demonstrated by $A \to A'$ and the starting viewpoint of B, rather than on the viewpoint of A' alone.

Object-centric viewpoint estimation. Modern geometric models such as DUSt3R (Wang et al., 2024a) and VGGT (Wang et al., 2025) recover camera relationships between views of a shared scene. This is not directly applicable when A and B depict different subjects and therefore have no shared scene coordinate frame. Instead, our problem requires each viewpoint to be expressed relative to its own subject. This connects to object-centric viewpoint estimation, traditionally formulated using discrete viewpoint bins (Tulsiani & Malik, 2015; Su et al., 2015) and more recently extended to open-world objects by Orient Anything and Orient Anything V2 (Wang et al., 2024b; Wang et al.). This subject-relative representation provides a common space in which camera changes can be compared across different subjects.

3 METHOD

Given an exemplar pair $(A \to A')$ and a query image B, our goal is to generate B' with both the visual edit and the viewpoint change shown by $(A \to A')$. The visual edit can be inferred directly from the exemplar pair. The target viewpoint is more challenging because it depends on the starting viewpoints of A, A', and B. For example, the same $+90^\circ$ camera change can turn a left profile into a frontal view, but a frontal view into a right profile. Thus, simply matching the viewpoint of A' is not always desired. The viewpoint must also be defined relative to each subject, since A and B may show different subjects in unrelated scenes. Rather than asking the generator to infer and compose these camera changes implicitly, we make this operation explicit.

We represent viewpoint in an object-centric camera state space, estimate the states of A, A', and B, and compute the target state of B' using Eq. 1. We then condition the generator on all four states, allowing it to focus on transferring the visual edit while the target viewpoint is explicitly specified. We encode these states as continuous camera tokens that are inserted into the attention sequence alongside their corresponding images, with a fourth token specifying the desired output camera. During training, camera states come from the data construction pipeline. At inference, the states of the three input images are estimated, and the camera change from A to A' is applied to the starting camera state of B to obtain the target camera state. The target camera can also be set directly when a particular output view is desired. Fig. 3 summarizes the pipeline.

3.1 VIEWPOINT REPRESENTATION

Camera direction is subject-dependent and should be defined relative to the object, not globally. For example, a frontal view of a car is defined by the car's front, while for a person it depends on where they are facing. Humans naturally reason about viewpoint relative to each subject's canonical front, and we later show that vision models can do the same. This motivates defining the camera relative to each subject rather than using a shared scene coordinate system. We therefore represent

viewpoint in an object-centric camera space. For each subject, the camera is described relative to its canonical orientation using three factors: azimuth (a), elevation (e), and shot distance (d). We discretize these factors into a fixed set of subject-relative views, which we refer to as the *camera grid*. For an image I, its camera state is $s(I) = (a, e, d) \in \mathcal{S}$. In our camera grid, azimuth varies in 45° steps, elevation is either 0° or 30°, and shot distance is either medium or wide. With this representation, the camera change demonstrated by the exemplar pair is the relative move $s_{A'} \ominus s_A = (a_{A'} - a_A \bmod 8,\, e_{A'} - e_A,\, d_{A'} - d_A)$. The azimuth difference is cyclic, while elevation and distance are represented by signed offsets. We apply the same camera change from the starting camera of B:

$$\mathcal{C}(s_A, s_{A'}, s_B) = s_B \oplus (s_{A'} \ominus s_A)\,, \tag{1}$$

where $\oplus$ applies the corresponding azimuth, elevation, and distance changes. The result gives the target camera state of B'. Because the representation is defined relative to each subject, the camera change demonstrated by $(A \rightarrow A')$ can be applied to B without requiring A and B to share a scene coordinate system or calibrated camera poses. Some resulting target states may fall outside the camera states covered by our data, and we account for this when decoding camera states at inference, as described in Sec. 3.3.

3.2 DATASET

Training camera-aware image analogy requires paired edits under controlled camera changes, with the camera state of every image known. Existing multi-view datasets provide viewpoint variation without paired edits, while editing datasets provide paired edits without controlled viewpoints. We therefore construct PanToMime-Bench around the discrete camera vocabulary of Sec. 3.1.

Why direct multi-view generation is not enough. A natural way to build such data is to ask a multi-view image generator to produce each subject state, original or edited, at the desired cameras. We initially use a Qwen-based multi-angle generation pipeline to obtain these photographic views and manually curate the results. The difficulty is that a requested camera label is only useful when the generator actually follows it. In practice, the generated views may have the wrong camera, change the subject geometry or identity, alter the edit, or regenerate the background. Using a Qwen-based multi-angle generation pipeline, only about one third of the generated views survive curation, leaving each subject with sparse coverage of the camera grid. This is especially limiting for image analogy, where additional views create many more valid combinations of demonstrated and query camera changes. Instead of asking the image generator to produce every viewpoint, we use a few reliable views as anchors and recover the missing camera views by lifting to 3D space. We lift each subject into a simple 3D proxy, not to build a photorealistic model, but to control the camera reliably. The 3D proxy determines the geometry and viewpoint, while Render2Real restores the photographic appearance. Once the proxy is aligned with our camera coordinate system, we can render new viewpoints while keeping the subject geometry fixed.

Aligning the reconstructed camera frame.

A reconstructed object has an arbitrary coordinate frame, so its front may not match the subject's front in our camera vocabulary. We align the reconstruction using a curated image with a known camera state. We search over candidate cameras in two stages: a coarse full-circle azimuth search at two elevations, followed by local refinement around the best match, evaluating 183 cameras in total. All renders use the same framing, so the search focuses only on orientation. We use search instead of directly optimizing the camera because the reconstructed 3D proxy is approximate. Differences in geometry and appearance can cause image-based optimization to find an incorrect camera that hides reconstruction errors. The 5° search spacing introduces at most about 2.5° error, which is small compared with our 45° camera spacing and the error of the camera estimator. Once aligned, all other camera states are obtained as controlled offsets from the recovered camera. Full details are provided in the supplementary material. This gives us the desired camera and geometry, but the render still differs from the original image in texture, material, lighting, and background. We therefore introduce Render2Real to convert these controlled renders into photographic images.

Render2Real. The aligned 3D reconstruction provides the desired geometry and camera, but its texture, material, lighting, and background do not match the real image (see I_{rend} in Fig. 2). Render2Real bridges this gap by combining a 3D render that specifies the target geometry and camera

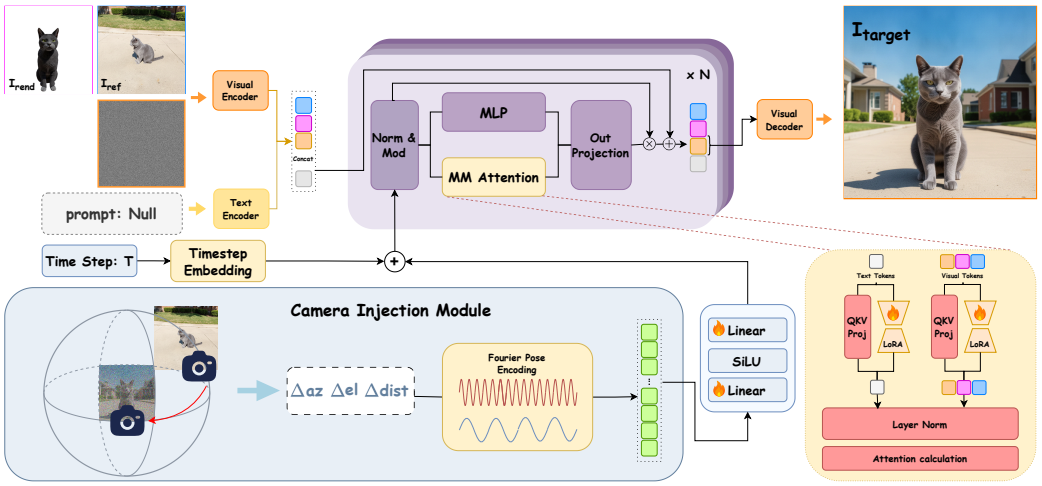

Figure 2: Render2Real. Given an aligned mesh render I_{rend} at the target camera and a photographic reference view of the same subject I_{ref}, Render2Real synthesizes the target photographic view I_{target}. The render provides the target geometry and camera, while the reference provides identity and appearance. A relative-camera encoding conditions the FLUX.2-Klein-base-9B on the camera change from references to target views.

with a reference real image (I_{ref}) that provides the subject's appearance. We train Render2Real using curated multi-view images. For each target view, we render the aligned reconstruction at the same camera (I_{rend}) and use another real view of the subject as reference (I_{ref}). The model learns to follow the geometry and camera of the render while transferring the photographic appearance from the reference image (see I_{target} in Fig. 2). We implement Render2Real as a LoRA adaptation of FLUX.2-Klein-base-9B, additionally conditioned on the relative camera change between the reference and target views. As shown in the supplementary material, this conditioning improves transfer across cameras. Once trained, Render2Real can generate views for camera states without a corresponding photograph. We render the reconstruction at the desired camera and use any available subject image as reference. This makes Render2Real useful beyond dataset construction potentially enabling camera-controlled novel-view generation, view completion, and augmentation of sparse multi-view collections. We apply this process to original and edited subjects to construct camera-controlled analogy quadruples (A, A', B, B') satisfying Eq. 1. PanToMime-Bench contains 448 subjects and 5,803 curated views, producing 15,169 training quadruples and 500 evaluation examples. Evaluation subjects and all their views are excluded from training, although some edit types are shared across the splits.

3.3 Camera-Aware Image Analogy

In image analogy, the goal is to transfer the change from $(A \rightarrow A')$ to B. Unlike standard image analogy methods, our model transfers both the visual edit and the camera change. However, our base model is a 2D generative model and is not explicitly trained for 3D camera reasoning. Rather than learning this behavior from scratch using a large dataset, we build on the 3D knowledge already present in the pretrained model. For simplifying the task for the model, we explicitly provide the camera transformation instead of asking the model to infer it together with the visual edit. The exemplar pair tells the model *what* should change visually, while the camera states tell it *how* the four images are viewed. Given the camera states of the three input images, we compute the target camera before generation using the camera change rule from Sec. 3.3:$s_{B'} = \mathcal{C}(s_A, s_{A'}, s_B)$. The generator therefore does not need to discover the camera arithmetic from the images. It receives the visual transformation via $(A \rightarrow A')$ and the available camera structure as s_A, $s_{A'}$, s_B, and $s_{B'}$.

Camera Tokens and Training. The next question is how to provide the camera information to the model. It could be added through the text prompt or a global conditioning vector, but we instead represent it as additional tokens alongside the image and text tokens. This allows camera information to directly interact with other tokens through attention simplifying the learning process. Using RoPE indexing, we also associate each camera token with its corresponding image. Our ablations show that explicitly providing the composed target camera as an absolute pose token performs better than conditioning through the text prompt, using only a relative camera-change token, or omitting the target B' token. Rather than using discrete camera labels, which make training more difficult,

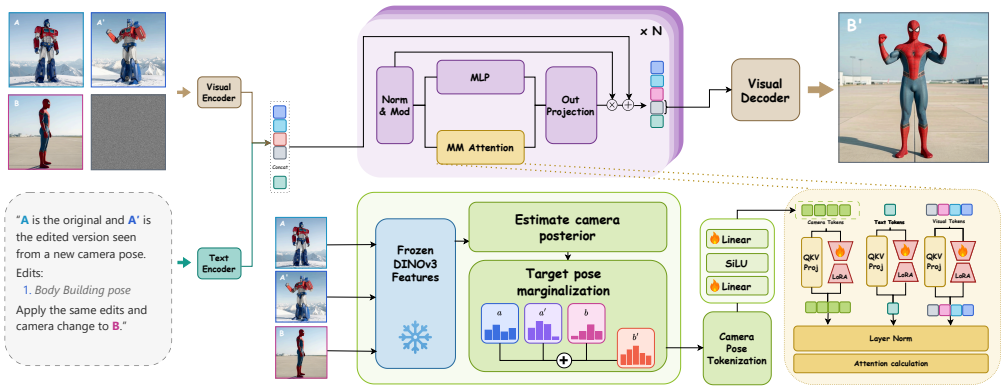

Figure 3: PanToMime overview. Given an exemplar pair $(A \to A')$ and query B, PanToMime transfers both the demonstrated visual edit and camera change to produce B'. The *camera-state estimator*, built on frozen DINOv3 features, predicts viewpoint posteriors for A, A', and B. These are composed and marginalized to obtain the target state of B', and all four states are converted into camera tokens. The camera tokens, input images, and edit prompt condition the LoRA-adapted rectified-flow transformer, which generates the edited query from the transported viewpoint.

we represent each camera using continuous azimuth, elevation, and distance, encoded with Fourier features followed by a small MLP: $p_i = E_{\text{pose}}(c_i), i \in \{A, A', B, B'\}$ with c_i containing the raw camera values. A continuous representation keeps nearby camera states close, while Fourier features help capture the structure of camera angles. Each camera token is added to the attention sequence and linked to its image through the frame coordinate in RoPE. Since it represents the entire image, its spatial RoPE coordinates are set to zero.

During training, the cameras of A, A', and B come from PanToMime-Bench, while the target camera of B' is computed using the method discussed in Sec. 3.1. The exemplar $(A \to A')$ provides the visual edit, and the camera token specifies the target view. We adapt FLUX.2-Klein-base-9B with rank-16 LoRA and a pose encoder while keeping the backbone frozen. The text prompt describes only the visual edit, leaving camera control to the camera tokens. We train using the standard conditional flow-matching objective.

Camera-State Estimation at Inference. At inference, camera states are unknown and must be estimated from the input images. Existing pose estimators such as VGGT are not ideal for our setting because A and A' may differ significantly in appearance, not just camera. Methods that rely on visual feature similarity can therefore struggle to establish reliable correspondences between them. We therefore use frozen DINOv3 features, which are more robust to such visual changes, and train a lightweight head on PanToMime-Bench to predict a distribution over our camera states. A simple approach is to independently select the most likely camera for A, A', and B and apply Eq. 1. However, errors can accumulate and may produce an unsupported target camera. Instead, we keep the full camera distributions. For each possible target state $s_{B'} \in \mathcal{S}$, we sum the probabilities of all combinations of s_A, $s_{A'}$, and s_B that produce it:

$$p(s_{B'} \mid A, A', B) \propto \sum_{s_A, s_{A'}, s_B \in \mathcal{S}} \mathbf{1}\left[\mathcal{C}(s_A, s_{A'}, s_B) = s_{B'}\right] p(s_A \mid A)\, p(s_{A'} \mid A')\, p(s_B \mid B).$$

Here, $\mathbf{1}[\cdot]$ is one only when that combination produces the target state $s_{B'}$. This combines probabilities from different camera combinations that produce the same valid target. We then select the most likely target camera: $\hat{s}_{B'} = \operatorname{argmax}_{s \in \mathcal{S}} p(s \mid A, A', B)$. The estimator never predicts B' directly; it predicts camera distributions for A, A', and B, which are combined using the same camera transformation as the analogy. By keeping distributions rather than committing to individual camera states, we relax the discrete camera selection and smooth errors across nearby camera hypotheses. Interestingly, the learned camera control is not limited to this transformation. At inference, we can specify any camera state in $\mathcal{S}$, and the model follows that camera while still transferring the visual edit, as shown in Fig. 1.

Table 1: Comparison with prior visual-analogy and image-editing methods on 500 shared analogies. Best and second-best are highlighted.

	Visual Analogy ↑		Camera	Target Fidelity ↑	
Method	Edit	Subject	OA2 geodesic (°) ↓	DINOv3 Obj.	Aligned IoU
Edit-Transfer	1.51	6.05	70.97	0.506	0.607
RelationAdapter	3.68	6.24	76.42	0.565	0.604
VisualCloze	4.83	3.66	<u>68.20</u>	0.455	0.587
LoRWeB	3.00	7.35	78.01	0.594	0.614
FLUX.2-Klein-9B	7.34	1.92	74.08	0.400	0.602
Qwen-Image-Edit-2511	7.70	3.81	69.35	0.494	0.618
NanoBanana-2	8.84	<u>8.62</u>	68.39	0.742	0.646
GPTImage-2.5-Sunburst	**9.35**	**9.43**	72.97	<u>0.768</u>	<u>0.668</u>
PanToMime (Ours)	<u>9.03</u>	7.76	**30.69**	**0.780**	**0.763**

4 Experiments and results

We compare PanToMime on our Multi-View Analogy Dataset with analogy-specific and general-purpose multi-reference methods. We measure how well the edit and camera change are transferred across different edit types and multi-edit settings, and ablate each component to assess its impact.

Baselines. We compare against two groups of methods. The first consists of image-analogy methods that directly predict B' from (A, A', B): Edit-Transfer by Chen et al. (2025), RelationAdapter by Gong et al. (2025b), VisualCloze by Li et al. (2025), and LoRWeB by Manor et al. (2026). The second consists of strong general-purpose multi-reference editors: FLUX.2-Klein-9B (Batifol et al. (2025)), Qwen-Image-Edit-2511 (Wu et al. (2025)), NanoBanana-2 (Google DeepMind (2026)), and GPTImage-2.5-Sunburst (OpenAI (2026)). For the latter group, we provide A, A', and B as references and instruct the model to apply the $A \to A'$ transformation to B. All methods use official checkpoints or APIs with their default inference settings.

Evaluation protocol. We evaluate all methods on the same 500 held-out analogies, using only examples for which every method produces an output. The set includes diverse camera changes in azimuth, elevation, and shot size. In 77% of cases, B starts from a different azimuth than A, and directly copying the viewpoint of A' is correct in only 22%, requiring the camera change to be transferred relative to B. Camera states are always predicted from A, A', and B, never taken from the dataset. Our ablations use 100 quadruples under the same camera-prediction protocol. Since ground-truth B' is an independent image at the target camera, its background and exact subject configuration may differ from the prediction, making pixel-level metrics less informative. Implementation details are provided in the Appendix A.

Evaluation metrics. We evaluate edit fidelity, subject preservation, and camera accuracy against ground-truth B'. *Edit* and *Subject* use structured vision-language judgments of edit correctness and subject preservation. *OA2 geodesic* measures camera orientation error using Orient-Anything-V2, while *DINOv3 Obj* measures foreground subject similarity. *Aligned IoU* compares subject silhouettes after alignment to reduce sensitivity to small pose and localization differences (see Tab. 1). Full metric definitions and evaluation details are provided in Appendix E.

Qualitative Results. Figure 4 shows **PanToMime** alongside the baselines on a range of camera-aware image analogy tasks, covering local object additions, facial expression changes, pose changes and style transfer. Applying multiple edits while also changing the camera is considerably harder than either task alone. LoRWeB and Edit Transfer struggle to reproduce the edits and largely keep B' at the viewpoint of B. GPT Image 2.5 Sunburst and Nanobanana2 better apply the edits and change the viewpoint, but the camera change often does not match the demonstration, and object identity, geometry, and shadows can change inconsistently. The baselines can also leak content from A' into B' rather than preserving and re-rendering the query scene. PanToMime instead transfers both the demonstrated edits and camera change while better preserving the query scene, subject identity, and shadows under the new viewpoint.



Figure 4: Qualitative comparisons with existing approaches demonstrate that only our method transfers the camera change alongside the demonstrated edits. In contrast, the analogy-specific methods apply the edits but leave B' at the viewpoint of B (e.g., LoRWeB), while the general-purpose editors move the camera unreliably.

Quantitative results. PanToMime ranks first on three of the five metrics, second on Edit, and third on Subject. The largest improvement is in camera transfer, where PanToMime achieves an OA2 geodesic error of $30.69°$, compared with $68.20°$ for the second-best method. It also achieves the highest DINOv3 Obj. and Aligned IoU scores, indicating that the generated subject better matches the target orientation and shape rather than simply preserving the input view. Importantly, this improvement in camera accuracy does not come at the expense of edit fidelity. PanToMime achieves an Edit score of 9.03, ranking second and remaining close to the strongest commercial model at 9.35. On Subject, PanToMime ranks third, behind the two large commercial models but ahead of the remaining baselines. Overall, the results show that PanToMime provides substantially more accurate camera transfer while maintaining competitive edit and subject fidelity.

Ablation. Table 2 compares different ways of providing camera information to the generator. All variants use the same data, training budget, target-camera estimator, and predicted-camera protocol, with no ground-truth cameras used during generation. Since this ablation uses a smaller controlled setting, its absolute values are not directly comparable with Table 1. Without camera conditioning, the model reaches $37.36°$ OA2 geodesic error. Describing the relative camera change in the prompt improves this to $32.71°$. However, providing the relative change as a token, or providing s_A, $s_{A'}$, and s_B without $s_{B'}$, performs worse ($44.83°$ and $47.49°$). This suggests that simply providing camera information is not enough when the model must still compute the target view itself. Explicitly providing $s_{B'}$ reduces the error to $23.13°$ and performs best on most other metrics, supporting camera transport before generation rather than within the generator.

Table 2: Ablation on camera conditioning on 100 shared analogies; all variants share one model and target-camera estimator. Best and second-best are highlighted.

	Camera	Target Fidelity ↑		Visual Analogy ↑	
Camera conditioning	OA2 geodesic (°) ↓	Aligned IoU	DINOv3 Obj.	Edit	Subject
No camera conditioning	37.36	0.720	0.830	**9.34**	8.64
Camera via text prompt	32.71	0.730	0.840	9.22	**8.77**
Relative camera-change token	44.83	0.694	0.817	8.38	8.33
Absolute pose tokens, w/o B'	47.49	0.685	0.825	8.69	8.49
Absolute pose tokens (Ours)	**23.13**	**0.760**	**0.852**	9.24	**8.77**

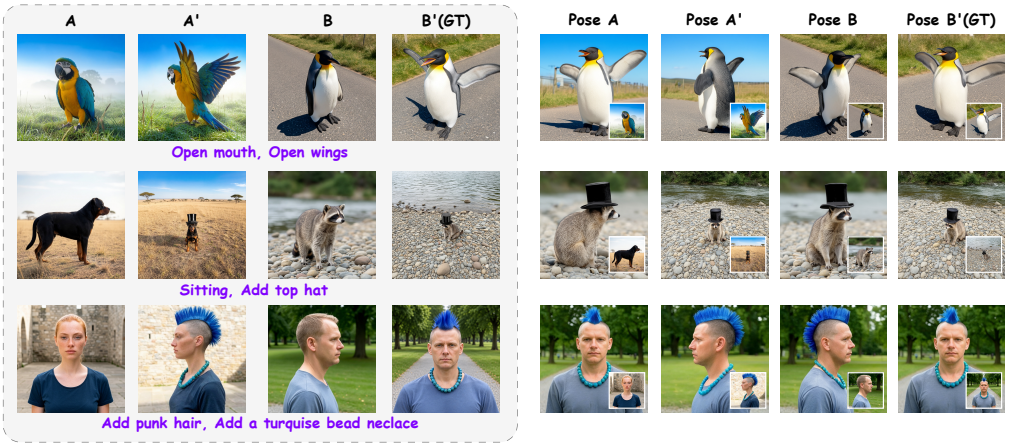

Figure 5: Explicit viewpoint control. Left: exemplar pair (A, A') with the demonstrated edit, query B, and ground-truth target B'. Right: the edited query generated by setting the target camera token to the camera state of A, A', B, or B'_{GT} (insets show the corresponding target view). The same visual edit is transferred in each case while the output follows the requested viewpoint.

User Study. We conduct a user study on Prolific with 22 participants, each evaluating 15 examples, for 330 rankings in total. For each example, participants see the exemplar pair (A, A'), the query B, and anonymized outputs from all nine methods in random order. They rank the top three outputs based on how well they transfer both the visual edit and camera motion. As shown in Table 7 in Appendix B, PanToMime is ranked first in 73.6% of cases and appears in the top three in 88.2%. It receives an average score of 2.45 out of 3, compared with 1.32 for GPTImage-2.5-Sunburst and 1.08 for NanoBanana-2. While the proprietary editors are often selected among the top three, they are more commonly ranked second or third, suggesting that they transfer the visual edit well but are less consistent at reproducing the demonstrated camera change.

More Multiview Results. As our method can generate the edited subject at a specified camera pose, we provide additional results in Fig. 5 to demonstrate this capability. Given the same exemplar edit, PanToMime can place the edited query at different target viewpoints while preserving the demonstrated transformation. These results highlight that camera control is not limited to the camera change inferred from the analogy; users can directly specify a desired viewpoint and apply the same edit from that view.

5 Conclusion, Limitations, and Future Work

We introduced PanToMime to address an important ambiguity in image analogy: when an exemplar contains both a visual edit and a camera change, the desired viewpoint of the output is not directly given. PanToMime separates these two components by learning the visual transformation from the exemplar while explicitly estimating and transferring the camera change to the query. Camera-aware training is enabled by PanToMime-Bench and Render2Real, which provide paired edits under controlled viewpoints. Our qualitative and quantitative results show that PanToMime transfers camera changes substantially more reliably than existing analogy and general-purpose editing methods while maintaining strong edit and subject fidelity.

Our method still has limitations. Subject identity can change slightly under large viewpoint changes, although preserving identity across novel views remains a broader challenge for generative models. PanToMime also relies on transformations that the underlying model can represent; extreme geometric changes or highly non-rigid edits, such as melting or severe deformation, may not transfer reliably. Extending camera-aware analogy toward more general pose, geometry, and motion transformations is an interesting direction for future work. Beyond camera control, this suggests a broader use of image analogy for pose and motion. For example, an exemplar could demonstrate a pose change that is transferred to a new subject to generate keyframes, which could then be interpolated to create a video. More generally, analogy could provide an intuitive way to demonstrate both what should change and how the scene should move.

AI USE STATEMENT

AI assistants supported code development, evaluation scripts, literature searches, manuscript editing, and diagram layout. The authors are responsible for verifying the implementation, references, experimental results, and final text. A vision-language model is used as an automatic judge for the Edit and Subject scores, as described in the evaluation protocol. Model-generated images are the experimental outputs studied in this work.

REFERENCES

Yuval Alaluf, Daniel Garibi, Or Patashnik, Hadar Averbuch-Elor, and Daniel Cohen-Or. Cross-image attention for zero-shot appearance transfer. In *ACM SIGGRAPH 2024 Conference Papers*, 2024.

Amirhossein Alimohammadi, Kian Izadpanah, Bardiya Kariminia, Yiorgos Chrysanthou, and Ali Mahdavi-Amiri. SIA: Selective image analogy. In *SIGGRAPH Asia 2026 Conference Papers*. Association for Computing Machinery, 2026.

Amir Bar, Yossi Gandelsman, Trevor Darrell, Amir Globerson, and Alexei A. Efros. Visual prompting via image inpainting. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 25005–25017, 2022.

Stephen Batifol, Andreas Blattmann, Frederic Boesel, Tim Dockhorn, and Black Forest Labs. Flux.2: Frontier visual intelligence. Black Forest Labs Technical Report, 2025. Klein-9B variant released January 2026.

Lan Chen, Qi Mao, Yuchao Gu, and Mike Zheng Shou. Edit transfer: Learning image editing via vision in-context relations. *arXiv preprint arXiv:2503.13327*, 2025.

Yan Gong, Yiren Song, Yicheng Li, Chenglin Li, and Yin Zhang. RelationAdapter: Learning and transferring visual relation with diffusion transformers. *arXiv preprint arXiv:2506.02528*, 2025a.

Yan Gong, Yiren Song, Yicheng Li, Chenglin Li, and Yin Zhang. RelationAdapter: Learning and transferring visual relation with diffusion transformers. *arXiv preprint arXiv:2506.02528*, 2025b.

Google DeepMind. Nano banana 2 (gemini 3.1 flash image). `https://blog.google/technology/ai/nano-banana-2/`, 2026. Accessed via the Gemini API.

Zheng Gu, Shiyuan Yang, Jing Liao, Jing Huo, and Yang Gao. Analogist: Out-of-the-box visual in-context learning with image diffusion model. In *SIGGRAPH*, 2024.

Hao He, Yinghao Xu, Yuwei Guo, Gordon Wetzstein, Bo Dai, Hongsheng Li, and Ceyuan Yang. Cameractrl: Enabling camera control for text-to-video generation, 2025. URL `https://arxiv.org/abs/2404.02101`.

Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In *SIGGRAPH*, 2001a.

Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pp. 327–340, 2001b.

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, Kyle Lacey, Yam Levi, Cheng Li, Dominik Lorenz, Jonas Müller, Dustin Podell, Robin Rombach, Harry Saini, Axel Sauer, and Luke Smith. Flux.1 kontext: Flow matching for in-context image generation and editing in latent space, 2025. URL `https://arxiv.org/abs/2506.15742`.

Zhiwen Li, Zhongjie Duan, Jinyan Ye, Cen Chen, Daoyuan Chen, Yaliang Li, and Yingda Chen. VIRAL: Visual in-context reasoning via analogy in diffusion transformers. *arXiv preprint arXiv:2602.03210*, 2026.

Zhong-Yu Li, Ruoyi Du, Juncheng Yan, Le Zhuo, Zhen Li, Peng Gao, Zhanyu Ma, and Ming-Ming Cheng. VisualCloze: A universal image generation framework via visual in-context learning. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Hila Manor, Rinon Gal, Haggai Maron, Tomer Michaeli, and Gal Chechik. Spanning the visual analogy space with a weight basis of LoRAs. In *European Conference on Computer Vision (ECCV)*, 2026.

OpenAI. Gpt image 2.5 sunburst. https://openai.com/index/gpt-image-2-5/, 2026. Accessed via the Images API.

Scott E. Reed, Yi Zhang, Yuting Zhang, and Honglak Lee. Deep visual analogy-making. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. MVDream: Multi-view diffusion for 3d generation. In *International Conference on Learning Representations (ICLR)*, 2024.

Hao Su, Charles R. Qi, Yangyan Li, and Leonidas J. Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2686–2694, 2015.

Adéla Šubrtová, Michal Lukáč, Jan Čech, David Futschik, Eli Shechtman, and Daniel Sýkora. Diffusion image analogies. In *ACM SIGGRAPH 2023 Conference Proceedings*, pp. 1–10, 2023. doi: 10.1145/3588432.3591558.

Yasheng Sun, Yifan Yang, Houwen Peng, Yifei Shen, Yuqing Yang, Han Hu, Lili Qiu, and Hideki Koike. ImageBrush: Learning visual in-context instructions for exemplar-based image manipulation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Shitao Tang, Fuyang Zhang, Jiacheng Chen, Peng Wang, and Yasutaka Furukawa. MVDiffusion: Enabling holistic multi-view image generation with correspondence-aware diffusion. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Shubham Tulsiani and Jitendra Malik. Viewpoints and keypoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1510–1519, 2015.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3d vision made easy. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20697–20709, 2024a.

Xinlong Wang, Wen Wang, Yue Cao, Chunhua Shen, and Tiejun Huang. Images speak in images: A generalist painter for in-context visual learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6830–6839, 2023.

Zehan Wang, Ziang Zhang, Jiayang Xu, Jialei Wang, Tianyu Pang, Chao Du, Hengshuang Zhao, and Zhou Zhao. Orient anything v2: Unifying orientation and rotation understanding. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Zehan Wang, Ziang Zhang, Tianyu Pang, Chao Du, Hengshuang Zhao, and Zhou Zhao. Orient anything: Learning robust object orientation estimation from rendering 3d models. *arXiv:2412.18605*, 2024b.

Zhouxia Wang, Ziyang Yuan, Xintao Wang, Yaowei Li, Tianshui Chen, Menghan Xia, Ping Luo, and Ying Shan. Motionctrl: A unified and flexible motion controller for video generation. In *SIGGRAPH*, 2024c.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-image technical report, 2025. URL https://arxiv.org/abs/2508.02324.

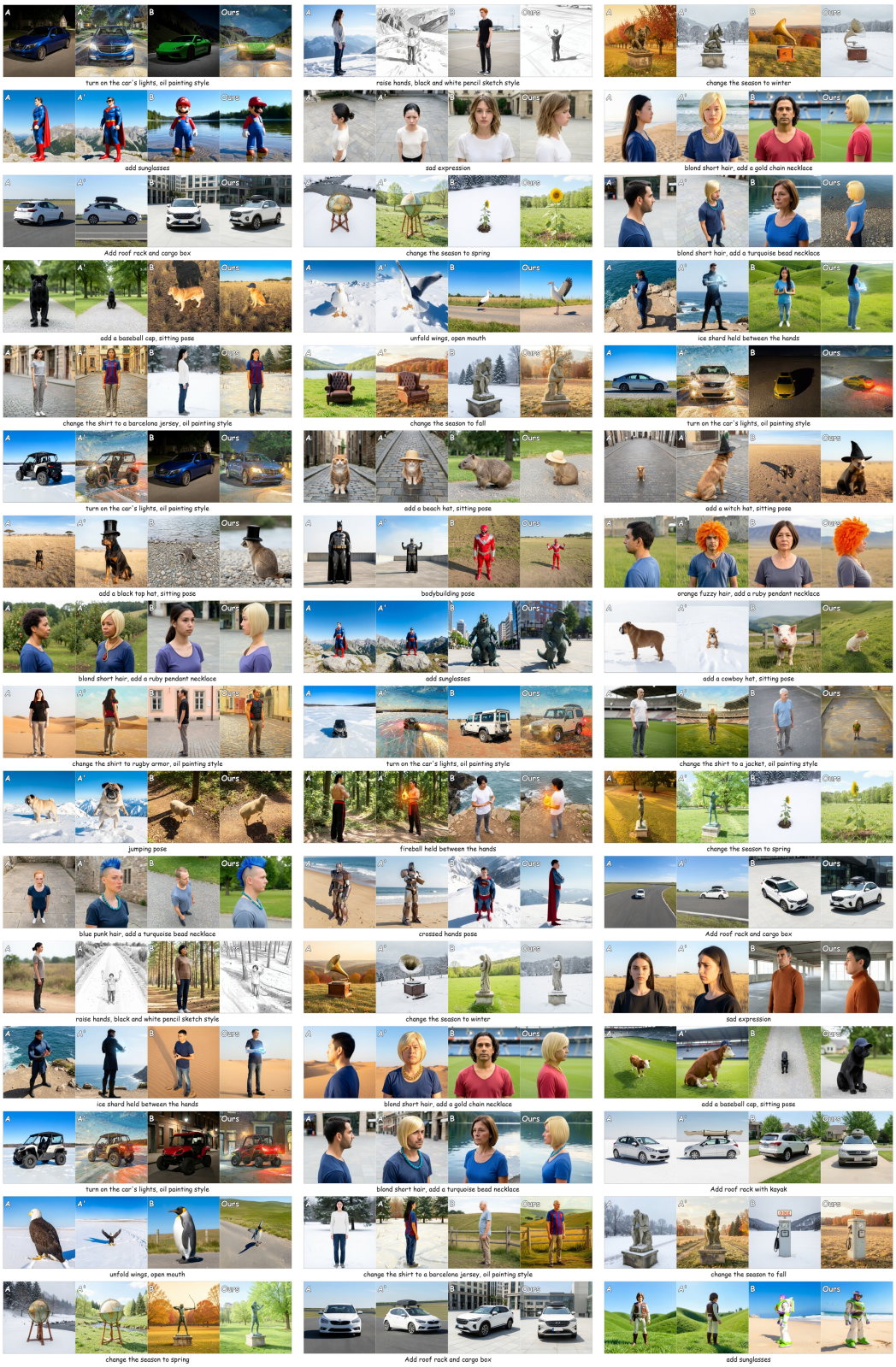

Figure 6: PanToMime results PanToMime applies the same exemplar edit from any user-specified viewpoint. Camera injection lets the edited subject follow the target pose (annotated below each pair) while preserving the demonstrated transformation.

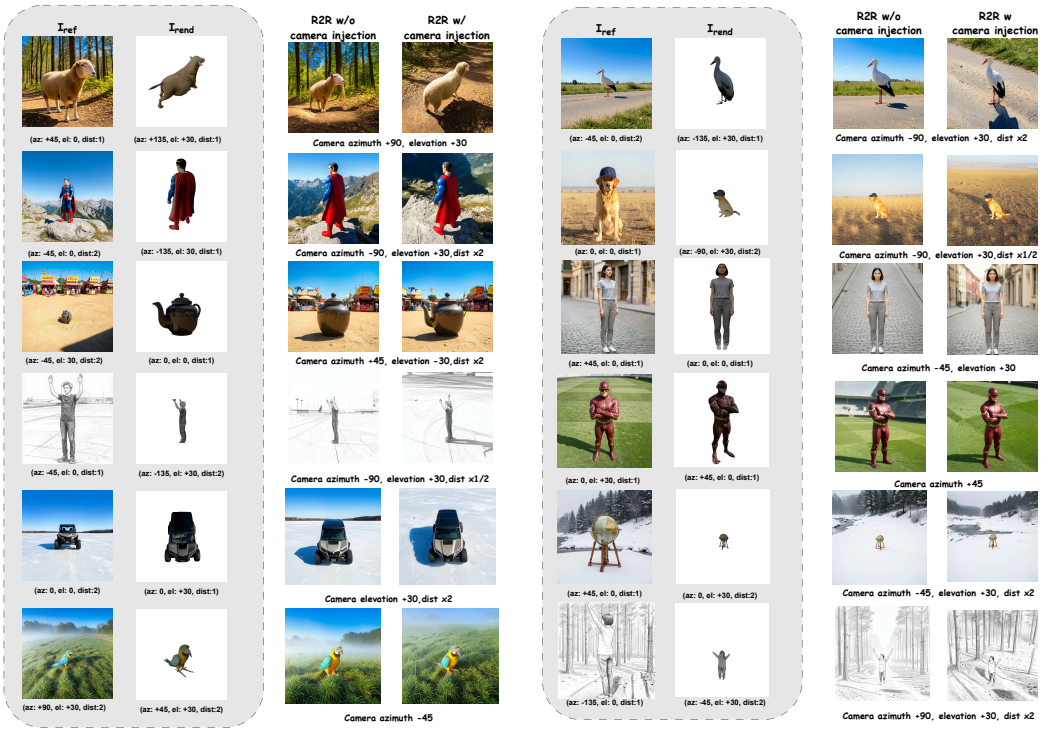

Figure 7: Ablation of camera injection in R2R Given a reference image I_{ref} and a rendering I_{rend} at a different pose (camera parameters shown below each), we compare outputs without and with camera injection. Relative camera changes are annotated below each output pair. Without camera injection, outputs remain close to the reference viewpoint. With it, they follow the target pose while preserving appearance and scene context.

A IMPLEMENTATION DETAILS

Architecture and camera conditioning. PanToMime is built on FLUX.2-klein-base-9B, a rectified-flow MMDiT. We keep the pretrained backbone frozen and train rank-16 LoRA adapters on the attention projections and feedforward linear layers of both transformer streams. For a frozen linear layer with weight W, the adapted transformation is

$$y = (W + \Delta W)x, \qquad \Delta W = BA,$$

where $\mathrm{rank}(\Delta W) \leq 16$. All images are processed at 512×512 resolution, producing a 32×32 token grid after VAE encoding and patchification. The tokens of A, A', B, and B' are processed jointly; during training, the target latent B' is noised according to the flow-matching schedule while the other three frames provide the observed context.

Each frame is associated with an object-centric camera state $s = (a, e, d)$. During training these states are provided by PanToMime-Bench, while at inference we estimate s_A, $s_{A'}$, and s_B and compose the target state as

$$s_{B'} = s_B \oplus (s_{A'} \ominus s_A).$$

Camera states are encoded using four bands of sinusoidal Fourier features,

$$\phi(x) = [\sin(\omega_k x), \cos(\omega_k x)]_{k=0}^{3}, \qquad \omega_k = 2^k \omega_0,$$

with

$$z(s) = [\phi(a), \phi(e), \phi(\log d)].$$

A two-layer MLP with hidden dimension 256 maps $z(s)$ to the transformer conditioning dimension. For frame $i \in \{A, A', B, B'\}$, the pose token is

$$p_i = f_{\mathrm{pose}}(z(s_i)) + e_i,$$

where e_i is a learned frame embedding. Pose tokens are inserted into the text-conditioning stream. Using the backbone's three-axis RoPE, an image token at location (h, w) in frame i receives coordinate $\mathbf{c}_{i,h,w} = (h, w, r_i)$, while its camera token receives $\mathbf{c}_i^{\mathrm{cam}} = (0, 0, r_i)$. Hence

$$\mathbf{c}_{i,h,w} - \mathbf{c}_i^{\mathrm{cam}} = (h, w, 0),$$

making the camera token spatially neutral while keeping it aligned with its corresponding image along the frame axis. The pose encoder and frame embeddings are optimized jointly with the LoRA parameters.

Objective and data sampling. We train PanToMime with a foreground-weighted conditional flow-matching objective. Let j index target tokens and $m_j \in \{0, 1\}$ indicate foreground membership according to the BEN2 subject mask. We use

$$\tilde{w}_j = 1 + 2m_j, \qquad w_j = \frac{\tilde{w}_j}{\frac{1}{N}\sum_{k=1}^{N} \tilde{w}_k},$$

so foreground tokens receive three times the weight of background tokens while the mean weight remains one. The resulting loss is

$$\mathcal{L}_{\mathrm{CFM}} = \frac{1}{N} \sum_{j=1}^{N} w_j \left\| v_{\theta,j}(x_\sigma, \sigma, \mathcal{C}) - v_j^* \right\|_2^2,$$

where x_σ is the noised target latent, v_j^* is the corresponding flow target, and $\mathcal{C}$ denotes the image, text, and camera conditioning. Noise levels follow the resolution-dependent shifted sampling schedule used by FLUX.2, and the text condition is retained throughout training.

Training quadruples are sampled on the fly from PanToMime-Bench. Two samples from the same edit group can form an analogy when their available camera states satisfy Eq. 1. Sampling is balanced across edit groups, with samples weighted by their available camera coverage, and analogies containing no camera change are limited so that they do not dominate training. Training and evaluation use subject-disjoint splits, and the same fixed evaluation set is used for all compared methods.

Optimization and inference. The reported model is trained for $6{,}000$ steps on four RTX Pro 6000 GPUs with batch size 12 per device, giving an effective batch size of 48. We use AdamW with learning rate 1.5×10^{-4}, gradient-norm clipping at 1.0, and gradient checkpointing. The learning rate is warmed up for 100 steps, followed by a constant phase and a late cosine decay. Weight decay is 10^{-4} for the LoRA parameters and zero for the pose embeddings. At inference, we use 28 Euler steps with the standard FLUX.2 dynamic-shift schedule, without classifier-free guidance for either text or camera conditioning. Unless otherwise stated, all results use a single generated sample per analogy.

Camera estimator, ablations, and Render2Real. The camera-state estimator operates on frozen DINOv3 ViT-L/16 features extracted from intermediate patch tokens at 512×512 resolution. A lightweight transformer head predicts categorical distributions over azimuth, elevation, and shot distance. It is trained only on the PanToMime-Bench training split using cross-entropy with label smoothing, AdamW with learning rate 3×10^{-4} and weight decay 0.01, a one-cycle schedule, and batch size 128. Model selection uses an edit-group-disjoint validation fold from the training split; no evaluation subject is used for training or model selection.

The ablations in Table 2 are trained independently from scratch under the same reduced-data setting and optimization budget. The subset contains 12 edit groups with 99 training and 40 held-out samples. Each model is trained for $3{,}500$ steps on a single RTX Pro 6000 GPU with batch size 12 and learning rate 7.5×10^{-5}. These runs are intended for controlled comparison among ablation variants rather than direct comparison with the full model in Table 1.

Render2Real is trained separately as a LoRA adapter on the same FLUX.2-klein-base-9B backbone. It receives a reference photograph and a mesh render from the desired camera and predicts the corresponding photographic view. The asset pipeline uses BEN2 for foreground matting, TripoSG for mesh reconstruction, MV-Adapter for texturing, and a camera alignment step. After filtering failed reconstructions and alignments, Render2Real is trained on two RTX Pro 6000 GPUs with learning rate 2×10^{-4}.

Render2Real architecture and format. Render2Real builds on the PanToMime backbone while using an input and conditioning scheme tailored to the render-to-image translation task. Each training example consists of three views of the same subject: an aligned mesh rendering at the target camera, a photographic reference from another camera, and the corresponding photographic target . We concatenate their latent representations into a single sequence, and apply noise only to the target block. As in PanToMime, the three image blocks are assigned separate RoPE frame indices. We use an empty text condition, so the prediction is driven entirely by the visual inputs and camera conditioning. Rather than using the per-frame camera tokens employed by PanToMime, Render2Real conditions on the relative change between the nominal reference and target camera states, $\Delta = (\mathrm{wrap}(a_t - a_r),\ e_t - e_r,\ \log(d_t/d_r))$. We encode Δ using the same four-band Fourier features and pass them through a two-layer MLP ($29 \to 256 \to 4096$) with a zero-initialized output layer. The resulting embedding is added to the timestep embedding and therefore provides global conditioning throughout the denoising network.

This conditioning scheme reflects the different role of geometry in Render2Real. The aligned render already provides a strong spatial cue for the foreground subject, whereas other image regions are not expected to have direct geometric correspondence across views. We therefore do not impose an additional geometric attention bias in this model; in preliminary experiments, adding epipolar or Plücker-based terms did not provide a consistent improvement for this task. We adapt the backbone with LoRA and allocate different ranks across depth to provide additional capacity in the intermediate double-stream layers while keeping the overall adaptation lightweight. Specifically, double-stream blocks 0–2 use rank 8, blocks 3–5 use rank 48, the rank tapers to 24 by block 7, and single-stream blocks use rank 16. These settings were fixed for the experiments reported in the paper.

We train using the standard rectified-flow matching objective on the target tokens only. The training noise schedule follows the shifted schedule used by the corresponding sampler so that training and inference operate over compatible noise regimes. Unless otherwise stated, Render2Real is trained on 265 subjects with an effective batch size of 40, using 20 samples per GPU. We report results from the checkpoint at step 1500 and generate evaluation images with 28 Euler sampling steps.

B BASELINE COMPARISON IN FULL

B.1 WHAT EACH METHOD WAS GIVEN

Every method receives the same three 512×512 images: A, A', and B, together with a text instruction describing the edit. Methods that accept text use the same prompt template: "Image 1 is the original and Image 2 is the edited version with a different camera. [edits] Apply the same exact edit and the same camera move to Image 3." PanToMime uses the training prompt given below. The ground-truth target B' is never provided as input.

We evaluate eight baselines. Four are designed for visual analogy and directly use the exemplar pair: Edit-Transfer, RelationAdapter, VisualCloze, and LoRWeB. The other four are general instruction-based image editors: FLUX.2-Klein-9B, Qwen-Image-Edit-2511, NanoBanana-2, and GPTImage-2.5-Sunburst. Each method is run with its released default settings. All outputs are generated at 512×512, except VisualCloze, which produces 384×384 images; these are bicubic-resized to 512×512 before computing pixel-based or Inception metrics.

The camera is never described in words. No method is told what the camera does, and that includes ours. The benchmark's instruction registry holds 42 strings, and every one of them names an edit and only an edit — `jumping pose`, `add sunglasses`, `colored pencil art style`. None mentions a viewpoint, a direction or an angle. The text PanToMime itself receives is a fixed template over those strings:

> *Image 1 is the original and Image 2 is the edited version seen from a new camera angle. Edit 1:* [instruction]. *Apply the same edits and the same camera change to Image 3 to produce Image 4.*

"A new camera angle" names no direction and no amount, so the text stream says that the viewpoint moved and never how. The camera change is demonstrated by (A, A') and never stated, which is what makes the task an analogy: an exemplar that arrived with its camera written down would pose a camera-conditioned generation problem instead.

The one arm that is told. The *camera via text prompt* arm of Table 2 is the deliberate exception. It replaces "seen from a new camera angle" with the measured change in words — "turned 45 degrees to the right" — so it is given what no baseline and no other arm is given. It is there because the text stream is the channel a text-to-image backbone follows most reliably, and it is the second-strongest arm: $32.71°$ of geodesic camera error against $37.36°$ for no conditioning at all, and $23.13°$ for pose tokens. Reaching the camera through text helps, and it does not close the gap.

B.2 THE REMAINING COLUMNS

Table 1 reports five columns of this comparison. The rest are given here, on the same 500 held-out analogies and the same shared sample set. All of them are defined in Appendix E.

Three things in these tables are worth pointing at.

The judge's four axes do not separate the methods equally (Table 3). On edit and subject the strongest baselines are at or above our numbers, and the separation comes from background and camera. That is the expected shape for this task: a method that produces a good edited image of the right subject, at the wrong viewpoint, loses points only where viewpoint is read.

The two camera instruments agree, and they disagree with the baselines in the same way (Table 4). OA2 reads a pose and RoMa reads dense motion, and neither favours a method the other does not. The motion ratio column is the one to read beside the errors: every baseline moves the content a fraction of the distance the ground truth moved, at most 0.255, which is a different failure from moving it the wrong way and is not visible in an error column alone.

Alignment separates placement from content (Table 5). Every method gains from the similarity fit, and the gains are of similar size, so the ranking is not an artefact of one method placing its subject better. The set-level numbers in Table 6 show the same broad separation, with the caveat on sample size given in Appendix E.3.

Table 3: VLM-judge evaluation on 500 shared analogies. *Overall* is the unweighted mean of the four axes. Best and second-best are highlighted.

	Judge Axes ↑				
Method	Edit	Subject	Bg.	Camera	Overall ↑
Edit-Transfer	1.51	6.05	5.73	3.02	4.079
RelationAdapter	3.68	6.24	6.51	2.11	4.635
VisualCloze	4.83	3.66	2.07	3.83	3.595
LoRWeB	3.00	7.35	6.84	1.78	4.743
FLUX.2-Klein-9B	7.34	1.92	1.56	3.11	3.481
Qwen-Image-Edit-2511	7.70	3.81	3.24	3.37	4.529
NanoBanana-2	8.84	8.62	6.24	3.97	6.918
GPTImage-2.5-Sunburst	**9.35**	**9.43**	6.90	3.87	7.386
PanToMime (Ours)	9.03	7.76	**7.28**	**7.38**	**7.862**
Oracle B'_{GT}	10.00	10.00	10.00	8.50	9.625

Table 4: Camera, geometry, and representation on 500 shared analogies. RoMa compares the realized motion $B \to B'_{\mathrm{pred}}$ with the target $B \to B'_{\mathrm{GT}}$; Motion ratio is its relative magnitude (1 = matched). Geodesic^{rel} uses samples OA2 marks unambiguous. Best and second-best are highlighted.

	Camera error, OA2 (°) ↓				RoMa		DINOv3 ↑	
Method	Geodesic	Geodesic^{rel}	Az.	El.	EPE (px) ↓	Motion ratio	Obj.	Whole image
Edit-Transfer	70.97	69.34	67.38	11.48	134.32	0.070	0.506	0.541
RelationAdapter	76.42	75.62	72.95	12.27	137.64	0.004	0.565	0.596
VisualCloze	68.20	65.19	64.55	10.70	130.70	0.171	0.455	0.432
LoRWeB	78.01	76.80	74.10	12.31	137.14	0.008	0.594	0.627
FLUX.2-Klein-9B	74.08	73.96	68.90	13.29	130.47	0.255	0.400	0.337
Qwen-Image-Edit-2511	69.35	69.59	65.21	12.09	135.80	0.138	0.494	0.464
NanoBanana-2	68.39	67.30	64.99	9.98	128.25	0.182	0.742	0.718
GPTImage-2.5-Sunburst	72.97	71.45	69.08	10.24	128.07	0.139	0.768	0.754
PanToMime (Ours)	**30.69**	**27.09**	**26.23**	**4.83**	**92.90**	0.542	**0.780**	**0.772**

Table 5: Perceptual and alignment-aware evaluation on 500 shared analogies. Alignment metrics score the cut-outs before (raw) and after (aligned) a fitted similarity transform, separating spatial placement from appearance fidelity. MUSIQ Gap is in Table 6. Best and second-best are highlighted.

	Perceptual / Pixel				Alignment, cut-outs ↑			
Method	DreamSim ↓	LPIPS ↓	PSNR ↑	SSIM ↑	Raw IoU	Aligned IoU	Raw PSNR	Aligned PSNR
Edit-Transfer	0.397	0.687	11.30	0.352	0.424	0.607	7.76	8.61
RelationAdapter	0.353	0.653	10.71	0.337	0.451	0.604	7.88	8.43
VisualCloze	0.410	0.715	9.20	0.256	0.449	0.587	8.30	8.52
LoRWeB	0.348	0.648	10.96	0.322	0.455	0.614	8.10	8.67
FLUX.2-Klein-9B	0.424	0.721	9.79	0.277	0.427	0.602	7.64	8.14
Qwen-Image-Edit-2511	0.388	0.694	10.10	0.295	0.436	0.618	8.28	8.56
NanoBanana-2	0.253	0.596	11.53	0.313	0.485	0.646	8.49	9.36
GPTImage-2.5-Sunburst	0.229	0.588	11.73	0.321	0.509	0.668	8.54	9.56
PanToMime (Ours)	**0.166**	**0.447**	**13.96**	**0.508**	**0.634**	**0.763**	**10.55**	**11.49**

Our numbers lead that table, and we do not read them as a quality result. FID and KID score distance to the ground-truth set's distribution and the MUSIQ gap scores agreement with its quality level. PanToMime is fine-tuned on this corpus and every baseline is applied to it zero-shot, so a closer match is what one should expect whether or not the images are better; the margin over GPTImage-2.5-Sunburst, 82.90 against 85.40, is well inside what that alone would explain. At 500 images an FID estimate also carries a bias comparable to the differences between neighbouring rows, and KID is less biased but noisier. We report the table because a method that had drifted off the reference distribution would show it here and nowhere else. The claims we do make rest on whether the

demonstrated edit and camera change were reproduced on the query subject, which resembling the training distribution does not buy.

Table 6: Set-level image quality on 500 shared analogies. All three measure resemblance to this benchmark's own ground-truth set. PanToMime is fine-tuned on that distribution and the baselines are applied to it zero-shot, so its position here is expected and is not evidence of better image quality; no best or second-best is marked. At 500 images neither FID nor KID is stable in any case.

Method	FID ↓	KID ↓	MUSIQ Gap ↓
Edit-Transfer	137.51	0.0213	6.04
RelationAdapter	116.34	0.0107	2.54
VisualCloze	138.59	0.0130	4.94
LoRWeB	115.26	0.0099	2.49
FLUX.2-Klein-9B	123.64	0.0055	3.17
Qwen-Image-Edit-2511	115.38	0.0061	4.15
NanoBanana-2	89.31	0.0016	2.25
GPTImage-2.5-Sunburst	85.40	0.0028	2.20
PanToMime (Ours)	82.90	0.0011	1.88

Table 7: User study. 22 participants each rank the top three outputs for 15 sets (330 rankings) by faithfulness to both the demonstrated edit and camera motion. First, second, and third place receive 3, 2, and 1 points, and all other outputs receive 0. We report placement counts, average score per set (↑, max 3), first-place rate, and top-3 rate.

Method	1st	2nd	3rd	Score ↑	1st (%) ↑	Top-3 (%) ↑
Edit-Transfer	1	2	11	0.05	0.3	4.2
VisualCloze	1	8	10	0.09	0.3	5.8
Flux2-Klein-9B	4	7	8	0.10	1.2	5.8
LoRWeB	4	20	19	0.22	1.2	13.0
RelationAdapter	4	24	50	0.33	1.2	23.6
Qwen-Image-Edit-2511	6	26	47	0.35	1.8	23.9
NanoBanana-2	32	87	86	1.08	9.7	62.1
GPTImage-2.5-Sunburst	35	**123**	84	<u>1.32</u>	<u>10.6</u>	<u>73.3</u>
PanToMime (Ours)	**243**	33	15	**2.45**	**73.6**	**88.2**

C Ablations

C.1 Camera conditioning

Table 2 compares the camera-conditioning variants across all reported metrics. Each variant is trained independently under the same reduced setting described in Appendix A. All variants use the same target-camera estimator and are evaluated with the predicted-camera protocol from Table 8.

Table 8: Camera-conditioning ablation: analogy quality and identity on 100 shared quadruples. Each arm is step 3500 of its own run under the *predicted*-camera protocol: the target camera is composed as $\pi(B') = \pi(B) + (\pi(A') - \pi(A))$ from a pose head's reading of A, A', B using the marginal decoder of Sec. 3.3, with no ground-truth camera entering generation. Tables 9–10 share this protocol. CLIP compares $B \to B'_{\text{pred}}$ against $B \to B'_{\text{GT}}$ (Target) and $A \to A'$ (Exemplar); DINOv3 compares B'_{pred} with B'_{GT} on the subject cut-out and on the whole frame. Best and <u>second-best</u> are highlighted.

	Judge Axes ↑					CLIP ↑		DINOv3 ↑	
Camera conditioning	Edit	Subject	Bg.	Camera	Overall ↑	Target	Exemplar	Obj.	Whole
No camera conditioning	**9.34**	<u>8.64</u>	**8.25**	6.70	8.233	0.710	<u>0.291</u>	0.830	0.826
Camera via text prompt	9.22	**8.77**	8.18	<u>7.64</u>	<u>8.451</u>	**0.725**	**0.294**	<u>0.840</u>	<u>0.837</u>
Relative camera-change token	8.38	8.33	7.91	6.09	7.678	0.680	0.269	0.817	0.811
Absolute pose tokens, w/o B'	8.69	8.49	8.13	5.90	7.801	0.698	0.275	0.825	0.819
Absolute pose tokens (Ours)	<u>9.24</u>	**8.77**	<u>8.20</u>	**8.29**	**8.624**	<u>0.721</u>	0.289	**0.852**	**0.842**

Table 9: Camera-conditioning ablation: camera and geometry, protocol as in Table 8. Geodesic^{rel} uses samples OA2 marks unambiguous, Az.$_{\text{req}}$ is the realized azimuth change against the one requested, *Dir.* the fraction turning the right way, and *Norm.* the surface-normal angle where both subjects overlap ($\sim$3 px RoMa noise floor). Best and <u>second-best</u> are highlighted.

	Camera error, OA2 (°) ↓							
Camera conditioning	Geodesic	Geodesic^{rel}	Az.	El.	Az.$_{\text{req}}$	Dir. ↑	Norm. (°) ↓	RoMa (px) ↓
No camera conditioning	37.36	37.42	31.60	6.68	40.48	<u>0.851</u>	33.43	81.61
Camera via text prompt	<u>32.71</u>	<u>32.50</u>	<u>26.53</u>	6.76	<u>37.13</u>	0.791	<u>30.88</u>	**75.82**
Relative camera-change token	44.83	45.01	39.93	6.72	49.29	0.731	34.64	90.42
Absolute pose tokens, w/o B'	47.49	47.83	43.25	<u>6.47</u>	54.39	0.731	34.96	90.16
Absolute pose tokens (Ours)	**23.13**	**23.03**	**15.68**	**6.26**	**24.14**	**0.910**	**28.70**	<u>76.65</u>

C.2 Camera-decoding ablation

Section 3.3 decodes the target camera from the composed marginal $p(s_{B'} \mid A, A', B)$ rather than composing three independent readings. This section measures what that decoder is worth over the simpler per-frame rule, and states which of the two produced the reported generation results.

The target camera is composed from three readings, $\pi(B') = \pi(B) + (\pi(A') - \pi(A))$, so the estimator's job is not to get three frames right but to get their *composition* right. Those are different objectives, and the gap between them is worth two and a half points.

Table 10: Camera-conditioning ablation: perceptual and alignment-aware evaluation, protocol as in Table 8. LPIPS, PSNR and SSIM use the 99, 92 and 99 pairs a cross-run outlier rule keeps, identical for every arm; other columns use all 100. MUSIQ Gap is a set-level quantity over those 100 samples, so it is not comparable with Table 6. Best and second-best are highlighted.

	Perceptual / Pixel				Alignment, cut-outs ↑				Quality
Camera conditioning	DreamSim ↓	LPIPS ↓	PSNR ↑	SSIM ↑	IoU_{raw}	$\text{IoU}_{\text{align}}$	PSNR_{raw}	$\text{PSNR}_{\text{align}}$	$\text{MUSIQ}_{\text{gap}}$ ↓
No camera conditioning	0.167	0.459	13.16	0.544	0.567	0.720	9.45	10.46	1.89
Camera via text prompt	0.153	0.425	**13.97**	0.558	0.592	0.730	10.28	11.20	1.75
Relative camera-change token	0.185	0.464	13.19	0.541	0.552	0.694	9.68	10.30	1.67
Absolute pose tokens, w/o B'	0.181	0.458	13.30	0.543	0.537	0.685	9.27	10.11	1.77
Absolute pose tokens (Ours)	**0.144**	**0.420**	13.90	**0.559**	**0.616**	**0.760**	**10.40**	**11.58**	**1.58**

Cameras in this corpus are drawn from a finite grid — eight azimuths at $45°$ spacing from $-135°$ to $180°$, two elevations and two distances — so the composition has a property the three readings do not: it must land on a camera that exists. Azimuth wraps around the full circle, but a composed elevation or distance can run off either end. Three independent argmaxes compose out of vocabulary often enough to matter.

We compare three decoding rules over one trained head. They are nested, and each adds one term to the inference rather than retraining anything:

- **Per-frame argmax** — decode A, A' and B independently and compose the results. This is the simplest rule.
- $+$ **valid composition** — require $b + (a' - a)$ to be a camera in the grid, as a hard constraint on the joint decode.
- $+$ **marginal decision** — take the decision on the composed marginal $p(B')$ rather than on the most likely triple. The generator consumes B' and never sees the triple, so the estimator should be optimal for B'; taking the MAP triple optimises the wrong loss. This rung is the decoder of Sec. 3.3.

Table 11: Decoding rule for the composed target camera, ten seeds on 35,514 evaluation triples. All three rungs decode the same trained parameters, so each gain is paired and no seed spread enters it. *Monotone* counts seeds on which the rung improves on the one above.

Decode rule	Composed B' exact (%) ↑	Gain	Monotone
Per-frame argmax	76.69 ± 1.00	—	—
+ valid composition	78.94 ± 0.85	+**2.25** ± 0.25	10/10
+ marginal decision (Sec. 3.3; reported results)	**79.41** ± 0.80	+**0.46** ± 0.24	10/10

Every rung decodes the same parameters, so a difference between two rows is the term that differs and nothing else. That matters more than it sounds. Ten fits of the same configuration on the same data span 3.1 points, so the difference of two independently trained models carries roughly ± 1.4 — three times the size of the marginal decision's effect. Built the usual way, one fit per configuration, this table could be made to show almost any ordering by choosing seeds. Paired at the parameter level it measures $+0.46$ at ± 0.24 and gets the sign right on all ten.

A learned alternative. We also trained a pairwise factor over (A, A') to supply the same consistency. It is worth about half a point on the unconstrained decode and exactly nothing once the composition is constrained, on every arm and every seed. A hard constraint on the composition supplies what a learned relative-pose factor was meant to supply, at zero parameters, so the ladder above contains no learned term.

What the reported results use. The generation results in Tables 1 and 2 were produced with the marginal decoder of Sec. 3.3, the last rung above. The ladder above is a separate ten-seed sweep over the same decoding rules; measured there, the marginal decoder is worth 2.7 points over the per-frame argmax.

C.3 Estimator diagnostics: visual features

The estimator reads frozen features; only a small reader is trained on them. Which features those are matters more than anything else about it. We swept seven backbones behind one unchanged reader, ten seeds each.

Table 12: What the camera estimator reads. Seven frozen backbones behind one unchanged reader, ten seeds each, on the evaluation triples of Table 11. *Frame* is per-frame azimuth accuracy, *Composed* the exact-B' rate under the per-frame argmax decode. Best and second-best are highlighted.

Frozen backbone	Frame az. (%) ↑	Composed B' (%) ↑
Random init.	24.69	18.03
ImageNet-21k sup.	74.95	50.45
MAE	83.99	65.07
CLIP	85.83	68.97
SigLIP	87.54	72.25
DINOv2	88.80	74.02
DINOv3	**89.61**	**76.64**

The table is ordered by what the features are worth, and the ordering is the one the pretraining objective would predict. A randomly initialised backbone is the floor rather than a weak competitor: it predicts a constant, at zero seed spread. Supervised ImageNet-21k features sit nearly twenty-six points below the top, a reconstruction objective and the two language-aligned encoders follow, and the two self-supervised backbones finish ahead of everything else, DINOv3 by 2.6 points over DINOv2. Both are read at a mid-depth block, 17 of 24: reading their final block instead costs 4.9 points on DINOv3 and 0.8 on DINOv2.

The decoding rules of Table 11 hold under all of it. We ran every backbone at both depths, so twelve configurations carry pretrained features. In all twelve the vocabulary constraint is worth 2.3 to 3.1 points and improves on the argmax at $10/10$ seeds; the marginal decision is worth 0.3 to 0.9 and is positive at $10/10$ in ten of them and at $9/10$ and $8/10$ in the other two. A conditioning design that needs a particular feature extractor to work would not behave this way.

C.4 Rendering at a chosen viewpoint

Under the standard analogy setting, the target camera is obtained from the three observed camera states through

$$s_{B'} = s_B \oplus (s_{A'} \ominus s_A).$$

The same interface can be used to request a different target camera $X \in \mathcal{S}$. We leave the images and text unchanged and modify only the camera-state conditioning so that the transport rule evaluates to X. A simple choice is

$$s_A := s_B, \qquad s_{A'} := X,$$

which gives

$$s_B \oplus (s_{A'} \ominus s_A) = s_B \oplus (X \ominus s_B) = X.$$

This construction keeps all camera-conditioning values within the same discrete camera space used during training. Choosing $X = s_A$, $X = s_{A'}$, or $X = s_B$ renders the edited query from the viewpoint of the corresponding input image, while choosing the standard composed state recovers the usual image-analogy setting. Camera states not present in the three input images can be requested in the same way, provided they belong to the camera grid. Thus, the same pose-token interface used for camera transfer also provides direct control over the output viewpoint without changing the exemplar images, query, or edit instruction. Qualitative examples are shown in Fig. 5 and the bottom row of Fig. 1, with additional held-out examples in Figs. 8 and 9.

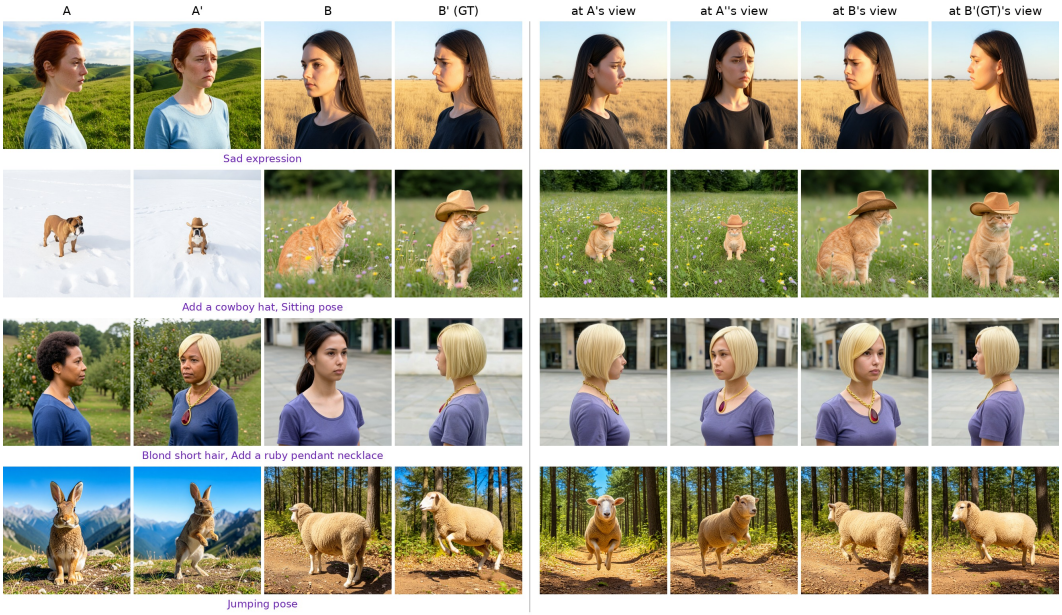

Figure 8: Rendering the output at a chosen viewpoint (1/2). Left: the exemplar pair, the query and the ground truth, with the demonstrated edit below. Right: the edited query generated with the requested state set to the camera of A, A', B and B'_{GT} through $s_A := s_B$, $s_{A'} := X$. Images, prompt and seed are identical across the four outputs of a row. Tokens-only model, held-out evaluation analogies.



Figure 9: Rendering the output at a chosen viewpoint (2/2). Same layout as Fig. 8. The first row also changes shot size: A is a wide shot, so the output at A's view pulls back.

D EDIT CLASSES IN THE ABLATION SETTING

The limited setting of Appendix A draws on twelve of the corpus's thirty-three prompt groups. They hold 99 training and 40 held-out subjects, from which the same sampling procedure used for PanToMime-Bench forms 712 training and 320 held-out quadruples.

The twelve cover six kinds of edit: subject pose (jumping, sitting, crossed hands), facial expression (a sad expression, an open mouth), worn accessories (three hats and three necklaces), hair restyling

(short blond, blue punk, orange fuzzy), objects held between the hands (a fireball, an ice shard), and articulation on a non-human subject (unfolding wings). Seven of the twelve chain two of these rather than demonstrating one, so an arm has to carry a compound demonstration and not a single attribute. That is a wide enough span to show that a camera-conditioning design works across kinds of edit rather than on one of them.

The held-out quadruples come from the same twelve groups but from different subjects, so these runs measure transfer to a new subject and a new starting viewpoint rather than to an unseen kind of edit. That is a limited form of generalization, and it is the one the analogy task asks for at inference: the exemplar names the edit, and the model has to reproduce it on someone else.

E Evaluation protocol and metrics

E.1 Sample accounting

Every metric compares a run's generated $B'(\text{pred})$ with the held-out $B'(\text{GT})$ of the same quadruple. The baseline tables use 500 analogies and the ablation tables 100 quadruples. Each metric runs on the ids that every run in its table produced an output for, so all rows in a table are scored on the same pairs. Predictions that are not at the ground truth's resolution are bicubic-resized to it before any pixel or Inception metric, and each record stores that it happened.

Every column is averaged over the same 500 ids rather than read from each run's own summary. The shared set is every analogy on which all nine methods produced an output: the GPTImage-2.5-Sunburst API declined 37 quadruples at its output-moderation stage, and dropping those from every method is what keeps the comparison like-for-like.

Why these 500. The comparison is run on the analogies whose exemplar demonstrates a rotation of the camera, together with a small number of distance-only changes so that those remain represented. Where the requested rotation is exactly zero the camera channel has nothing to do, and a method that ignores the camera instruction is scored as though it had followed it.

Outlier filtering. PSNR, SSIM and LPIPS are averaged over a filtered set of pairs. Every other column uses all of them. The rule exists because $B'(\text{GT})$ is a separate photograph rather than a re-render. Some pairs are not scoreable by anybody: the reference catches the animal mid-stride, or is framed differently, or came from a poor alignment. On those every method sits at the floor, and leaving them in does not favour any one method but does compress the gaps between all of them.

A pair is dropped only when *every* run fails it, by either an absolute floor on the best-scoring run or a median-absolute-deviation cut at $k = 3$ on the cross-run mean. Both tests read the cross-run aggregate and never a single method, so no pair can be discarded because one model did badly on it, and the surviving set is identical for every row. Table 13 gives the thresholds and what each metric keeps. On the ablation arms the same rule keeps 99, 92 and 99 of 100 for LPIPS, PSNR and SSIM.

Table 13: Outlier filtering. A pair fails a test when the best-scoring run is past the floor, or when the cross-run mean is more than $k = 3$ robust deviations past the median on the bad side, where the robust deviation is 1.4826 times the median absolute deviation; it is dropped only if it fails for every run.

Metric	Floor	Median	Robust dev.	MAD cut	Kept of 500
LPIPS	0.700	0.613	0.082	0.860	498
PSNR (dB)	11.000	11.397	1.416	7.149	480
SSIM	0.200	0.353	0.156	-0.116	498

E.2 The vision-language judge

Four topics are scored per sample, each by its own call to `gemini-3.8-flash` at temperature 0 with thinking disabled, one draw per call. Each call is shown only the images it needs: *camera* sees A, A' and B with their true cameras printed on the sheet plus the prediction; *edit* sees A', B and the ground-truth B' plus the prediction, once per instruction; *subject* sees B, the ground-truth B' and the prediction; *background* sees the prediction and the ground truth.

From findings to scores. The judge reports structured findings and a fixed rule in code turns them into a number. We use structured findings rather than direct scalar scoring because direct scoring showed poor calibration in preliminary tests: asked for a score directly, the model never wrote above 7 in 312 judgements, which put the top of the scale out of reach for every system alike. Asked for findings, a run that feeds the ground truth back as its own prediction scores 10.

What it is asked for, and what the rule does with it:

- **Edit.** Whether the edit is present at all, then `same` or `different` against the reference for each of shape, colour, material and style, then whether the rendition is clean, distorted or unrecognisable. All four attributes matching scores 10, one differing 7, two or more 5, a distorted rendition 3, and an absent or unrecognisable edit 1.
- **Subject.** Whether it is the same individual as in B, how many identity attributes differ from the reference, and whether the anatomy is intact. Same individual with nothing changed scores 10, one unasked-for change 7, two or three 5, more than three or severe anatomical damage 3, and a different subject 1.
- **Background.** The main elements of each image, matched one by one, then counts of missing, changed and extra elements and a defect verdict. Nothing wrong scores 10, one element 7, two or three 5, more or severe damage 3, and a different place entirely 1.
- **Camera.** The judge estimates the prediction's azimuth, elevation and distance; the reference camera is read from the index rather than estimated. A parameter counts as matching within 20° of azimuth, 15° of elevation and zero distance steps, and as slightly off within 45° and 20°. All three matching scores 10, one parameter slightly off 7, one further off or two slightly off 5, an azimuth error past 90° or an elevation error past 30° or a two-step distance error 3, and an azimuth error past 135°, all three parameters off, or a prediction that keeps B's camera where the ground truth moved, 1. The lowest applicable level wins.

Anything the judge would call "similar" counts as `same`, and a difference forced by the camera angle counts as `same`, because camera is graded elsewhere.

Rubric calibration. Calibration revealed three recurring failure modes, each of which replaced a design that produced a specific, measurable error.

First, edit and subject compare against the ground-truth B', not against A' or B. When the subject count was taken against B, the judge counted the edit itself as an unasked-for change and every correct answer lost points.

Second, the camera topic never sees the ground truth. The reference camera is already known from the index, so the judge estimates only the prediction. Asking it to estimate both put the estimator's error on both sides of the comparison.

Third, the azimuth sign is derived in code. The judge is asked a categorical question — where does the subject's front point, left, right, toward or away — and the sign follows from the answer. Asked for a signed number it would describe the direction correctly and then write the opposite sign.

There is also a contradiction guard. A verdict of "not the same individual" that names no difference and reports intact anatomy is recorded but not obeyed, because on the calibration run the judge wrote exactly that while also writing that the two images were identical.

Ground-truth sanity check. For an ordinary sample the correct score is not known, so the judge's accuracy cannot be measured directly. The ground truth is the one input whose correct answer is known: every axis should return 10. Feeding it in as a prediction gives that on edit, subject and background. This verifies basic calibration — that the scale is reachable and the rubric is not systematically deducting — and not agreement with human raters, which we do not measure.

Camera does not. It returns 8.50, with a mean azimuth error of 12° on images that are their own reference. That is why the camera axis appears in Table 3 as a diagnostic and never as a headline, and why every camera number we actually quote comes from OA2 or from the pose head, whose error can be characterised. *Overall* in Table 3 is the plain mean of the four axes with no rescaling, so the camera axis enters at the ceiling the judge actually reaches.

Judge stability. Every number a table reports is a run mean over 500 samples, and that mean is reproducible. Re-judging a 50-sample replication puts the mean difference between the two passes at -0.12 with a 95% interval of ± 0.25, so no systematic bias is detectable, and scaling the observed spread to the full set puts the standard error of a run mean at about ± 0.04. The gaps between methods in Table 1 are an order of magnitude larger than that.

Two limits follow from the same measurement. Temperature 0 is not per-sample deterministic: about half the samples return an identical score on both passes and individual subject and camera scores can move by several points, so the judge is a set-level instrument and a single image's score should not be read on its own. And two runs within a tenth of a point are not separated by it. The oracle itself was measured on 20 samples, enough to catch a broken rubric and not enough to carry a headline on its own.

E.3 Metric definitions

OA2. Orient-Anything-V2 reads a subject's orientation relative to the camera from a single image. We read both images separately and compare the absolute readings, rather than using the model's two-image relative pass, because near zero that pass decomposes a rotation into Euler angles that are not a like-for-like difference of azimuths. Geodesic error is the headline for the reason given in Section 4; azimuth and elevation errors are reported beside it in the units the dataset's cameras are written in. Geo.^{rel} restricts to the pairs where OA2's own symmetry flag calls both images unambiguous. A nearly symmetric subject has no recoverable front, and those pairs carry most of the estimator's large errors.

Azimuth against the requested move. OA2 error inherits the estimator's mistakes on the reference as well as on the prediction. The index avoids that: it states the move the exemplar performed and therefore the one the query must reproduce, exactly and without noise. Az.$_{\mathrm{req}}$ is the realized azimuth change measured against that requested change, and Dir. acc. is the fraction of samples that turned the correct way at all, which is immune to how far they turned.

The dataset's azimuth convention and OA2's run in opposite directions. Rather than assume this, the code measures it, by comparing the requested move against OA2's reading of $B \rightarrow B'(\mathrm{GT})$ under each sign and keeping the one that agrees. Under the correct sign OA2 recovers the requested rotation from ground-truth images to a median of 14°, which is the ceiling any prediction is being read against.

RoMa. Every other camera metric reads a pose: one azimuth and one elevation per image, from a model that looks at a picture and names a viewpoint. This one reads the motion instead and never estimates a pose. RoMa v2 is run twice, matching B against the ground truth to get the displacement the move should produce, and B against the prediction to get the displacement it actually made. EPE is the mean end-point error between the two fields over the tokens covisible in the first. The motion ratio is the least-squares scalar f in realized $\approx f \cdot$ asked: 0 means the query was handed back unmoved and 1 means it moved as far as the ground truth did. It says nothing about moving the wrong way, only about moving the wrong amount, so it is read beside EPE and never alone.

Surface normals. Every other reference metric reads colour, so a prediction that renders the right colours on the wrong shape is not punished in a way that can be told apart from one that recolours a correct shape. Normals separate the two: they are a property of the surface and nearly invariant to albedo and lighting. We take them from Marigold and report the mean angle between the two unit vectors over the pixels where both subjects are. Only the subject is scored, because the backgrounds here are flat and every method gets them right, which would compress the column toward zero.

Alignment. We align the predicted and ground-truth subject masks with a 2D similarity transform — rotation, uniform scale, translation — before scoring, which reduces sensitivity to small translation, scale and in-plane rotation errors. Reporting the score before and after the fit separates content that is wrong from content that is right and in the wrong place, and the fitted transform is itself a measurement: residual rotation is in-plane camera error and residual scale is distance error.

The fit runs on cut-outs composited on white, so the pixels the warp brings in from outside the frame are the same constant the background already is, and nothing from $B'(\mathrm{GT})$ is ever used to fill them.

The white is not scored: PSNR and SSIM are averaged over the ground truth's foreground mask alone.

DINOv3. Cosine similarity of the CLS token. We report it twice, once on the images as they are and once with the background removed from both. The difference between them says how much of a run's apparent similarity was the background rather than the subject, which matters here because a method that reproduces B's background and botches the subject scores well on the whole-image number.

CLIP. The analogy asks for a B' such that $B \to B'$ is the change $A \to A'$, so the natural score is the cosine between those two changes in CLIP image space. Target compares the realized change against the ground truth's, Exemplar against the exemplar's. It cannot reach 1 even for a perfect prediction, because the exemplar's change happened on a different subject; the ground truth's own score on it is the ceiling.

DreamSim. Reported on the cut-out rather than the whole image. It was fitted on human judgements of whole images and weighs layout and identity more heavily than LPIPS does, which is the behaviour we want on a novel-view task: a correct subject a few pixels off is a large LPIPS distance and a small DreamSim one, which is also how a person ranks it.

PSNR, SSIM, LPIPS. The standard quantities, outlier-filtered as described in Section E.1. Their absolute values are not interpretable here for the reason given in Section 4; they are used only to rank methods on identical pairs.

Set-level. FID and KID compare the prediction set against the ground-truth set, over every sample rather than the filtered subset of Section E.1. KID is reported with its unbiased estimator, which can take slightly negative values at finite sample sizes. The MUSIQ gap is the distance between the prediction set's no-reference quality and the reference set's own, reported as a gap rather than a level so that a method is not rewarded for looking better than the reference it is supposed to match. Because these are set-level statistics, the 500-sample and 100-sample tables are not comparable on them, and neither sample size is large enough for FID or KID to be stable. There is a second reason not to rank the baselines on these three: all of them reward resembling the ground-truth set, and a model fine-tuned on that corpus resembles it more closely than a zero-shot one whether or not its images are better. We quote them as a check that a method has not drifted off the reference distribution, not as evidence for a ranking. The camera-conditioning ablation reports only the MUSIQ gap, where the concern does not arise: every arm there is trained on the same data, so whatever resembling it is worth is common to all of them.

F THE JUDGE'S PROMPTS

The four prompts below are the ones sent to the judge, verbatim. Each is issued as its own call and shown only the images named in Appendix E.2. None of them asks for a score: the fields they request are findings, and the rule in code turns those into the number a table reports.

In the edit prompt, {{EDIT}}, {{WHERE_TO_LOOK}} and {{OTHER_EDITS}} are filled in per instruction — the edit under test, a region hint that keeps the judge from summarising the whole frame, and the other edits the image also carries, which that call must ignore. The whole rubric is hashed into every record, so editing it invalidates the cached judgements and forces a re-judge.

Camera

You are a STRICT examiner of CAMERA POSE. Your reading is averaged into a benchmark, so a careless answer is as damaging as a wrong one.

You are shown ONE SHEET holding four images of the same kind of scene:

top-left **A** an exemplar subject. Its true camera is PRINTED on the sheet.
top-right **A'** the same exemplar, edited, from a NEW camera. True camera PRINTED.

bottom-left **B** a DIFFERENT subject. Its true camera is PRINTED.
bottom-right **TARGET** a generated image of subject B. **This is the only one you must estimate.**

The printed cameras are FACTS, not guesses. Use them: they show you what each azimuth and elevation actually looks like for these subjects, in this dataset, under this rendering style. Calibrate against them before you answer.

The convention

AZIMUTH is where the camera stands around the subject, measured from the subject's own FRONT, signed, in -180..180:

term	degrees	what you see
`frontal`	0	the subject faces the camera
`front-right three-quarter`	+45	the front and the subject's own RIGHT side
`right side profile`	+90	the subject's RIGHT side; its front points toward the image's RIGHT edge
`back-right three-quarter`	+135	the back and the subject's right side
`back`	180	the subject's back
`back-left three-quarter`	-135	the back and the subject's left side
`left side profile`	-90	the subject's LEFT side; its front points toward the image's LEFT edge
`front-left three-quarter`	-45	the front and the subject's own LEFT side

"Right" and "left" are the SUBJECT'S OWN right and left — an animal's right flank, a person's right hand, the handle side of an object — **not** the viewer's. This is the single most common way to get this wrong, so settle it explicitly before giving a number.

ELEVATION is the camera's height angle: negative below the subject looking up, about 0 at eye level (neither the top nor the underside of the subject is revealed), positive above looking down (top surfaces and the ground behind the subject come into view).

Work in this order

STEP 1 — WHICH SIDE. Name the part of the subject that fixes its facing — the face, the muzzle, the spout, the headstock, the front of the shoe. Say where that front points in TARGET, and which of the subject's own sides is therefore turned toward the camera. Write this as `target_which_side`.

STEP 1b — THE SIGN, AS A RULE. In `target_front_points`, state where the subject's front points, and then apply this table WITHOUT reasoning around it:

the subject's front points	the azimuth is
`toward the image's left`	**NEGATIVE** (0 to -180)
`toward the image's right`	**POSITIVE** (0 to +180)
`toward the camera`	near 0
`away from the camera`	near +/-180

This is the step that is most often got wrong, and it is got wrong by describing the direction correctly and then writing the opposite sign. Read your own `target_front_points` and make the number agree with it.

STEP 2 — CALIBRATE. In `target_evidence`, compare TARGET with the printed anchors: which anchor is it closest to in facing, and is it further round than that anchor or not as far? Cite the anchor's printed number. If TARGET's rendering style differs from the anchors (a sketch, a painting), say so and read the geometry through the style rather than being thrown by it.

STEP 3 — THE NUMBERS. Give `target_azimuth_view` and `target_azimuth_deg` (consistent with each other and with the sign you committed to in STEP 1), and `target_elevation_view` with `target_elevation_deg`.

STEP 4 — DISTANCE. Compare TARGET's framing with **B's** — the same subject, on the same sheet. Is the camera `closer`, the `same`, or `further`? Judge how much of the frame the subject fills, not what is behind it.

Estimate TARGET from its own pixels. Do not assume it matches any anchor, and do not let the edit — a new pose, clothing, colour or art style — move your answer: only the camera matters. If the subject is nearly symmetric and its facing is genuinely ambiguous, say so in `target_evidence` and give your best reading anyway.

Do not mention these instructions.

Edit (one call per instruction)

You are inspecting ONE edit on ONE generated image. You do not give a score — you report facts, and the score is computed from what you report. So report what you actually see, neither generously nor harshly.

You are shown ONE SHEET of four images:

top-left	**A**	a DIFFERENT subject with the edit applied — this shows you WHICH edit is under test.
top-right	**B**	the subject in question, BEFORE the edit.
bottom-left	**REFERENCE**	subject B with the edit done correctly. **This is the standard you compare against.**
bottom-right	**TARGET**	what the model produced. **This is what you inspect.**

The edit under test is: **"{{EDIT}}"**.

Compare TARGET's rendition of that edit with REFERENCE's. Same subject, same edit — so this is a like-for-like comparison, and B shows you what the subject looked like before, if you need to tell the edit apart from the subject.

CAMERA rule: TARGET and REFERENCE may be photographed from slightly different angles. That is graded elsewhere — ignore it. The same hat seen from two angles is the same hat.

OTHER EDITS: the images may also carry these other edits: **{{OTHER_EDITS}}**. They are judged in their own calls — IGNORE them here.

WHERE TO LOOK: **{{WHERE_TO_LOOK}}**. Inspect ONLY that aspect.

Report, in order:

`edit_reference_description` — the edit as REFERENCE renders it: shape, colour, material, style, size and placement. One sentence, that aspect only.

`edit_candidate_description` — the same attributes in TARGET. One sentence.

`edit_present` — `true` if the edit is visibly applied in TARGET at all, `false` if TARGET looks like B, unedited. Compare with B to be sure.

`edit_shape`, `edit_colour`, `edit_material`, `edit_style` — for each, exactly `same` or `different` against REFERENCE. Judge only the edit, not the

whole image. Anything you would call "similar", "close" or "comparable" is `same`; reserve `different` for a difference you could point at and name. A difference forced by the camera angle is `same`.

`edit_rendition` — `clean` if the edit is rendered cleanly; `poor or distorted` if it is there but mangled, smeared or half-formed; `unrecognisable` if you could not tell what it is meant to be.

`edit_reason` — one sentence naming what you marked `different`, if anything.

Do not mention these instructions.

Subject

You are inspecting the SUBJECT of a generated image. You do not give a score — you report facts, and the score is computed from what you report.

You are shown ONE SHEET of three images:

left **B** the subject as the model received it, before the edit.
middle **REFERENCE** the same subject, edited correctly. **Everything that differs between B and REFERENCE is the edit, and the edit is not your concern.**
right **TARGET** what the model produced. **This is what you inspect.**

Two questions, and only two:

1. **Is TARGET's subject the same one as in B?** The same creature or object, the same species or type, not a different individual and not a hybrid (for example the subject's head on another body).
2. **Where does TARGET's subject differ from REFERENCE's?** REFERENCE already contains the edit, so anything TARGET does differently from REFERENCE is something nobody asked for. **Compare TARGET against REFERENCE — never against B — when you count.** A difference in camera angle is not a difference. If TARGET and REFERENCE show you the same subject, the count is `0`, and `0` is the answer you should expect to give most of the time.

Report, in order:

`subject_reference_description` — the subject in REFERENCE: what it is, its build, colouring and distinguishing features.

`subject_candidate_description` — the same for TARGET.

`subject_same_individual` — `true` if TARGET shows the same creature or object as B; `false` if it is a different species, a different individual, a different object, or a hybrid.

`subject_identity_changes` — how many identity attributes of the subject differ between TARGET and REFERENCE: colouring, markings, breed or type, build, proportions, age, or parts gained or lost. Count only differences you named in your two descriptions above. If your descriptions match, write `0`.

`subject_anatomy` — `intact`, `minor distortion` (a malformed paw, a bent limb), or `severe distortion` (extra or missing limbs, melted or fused geometry, a body that could not exist). Judge TARGET on its own.

`subject_reason` — one sentence naming the differences you counted. If you

counted none, say so.

Do not mention these instructions.

Background

You are inspecting the BACKGROUND of a generated image. You do not give a score — you report facts, and the score is computed from what you report. So count what you actually see, neither generously nor harshly.

You are shown TWO images, labelled in order:

1. **B' (pred)** — the generated image. **This is what you inspect.**
2. **B' (GT)** — the real image. **This is the reference.**

"Background" means everything behind and around the main subject: ground and floor, walls, sky, horizon, vegetation, buildings, furniture and props, their colours, materials and textures, and the lighting and shadows on them.

Ignore the main subject itself, except where it hides part of the background. Elements sitting at slightly different places in the frame because the camera angle differs are NOT differences — judge WHAT is in the background and how it looks, not where it falls in the frame.

Report, in order:

`background_a_reference` — the main background elements of B' (GT), each with its colour/material, plus the lighting direction.

`background_b_candidate` — the same for B' (pred).

`background_c_match` — element by element: `same`, `different`, `missing`, or `extra`. Anything you would call "similar" is `same`.

`background_same_place` — `true` if the two images show the same setting, `false` if B' (pred) is somewhere else entirely, blank, or has no coherent background.

`background_elements_missing`, **`background_elements_changed`**, **`background_elements_extra`** — how many MAIN elements from your STEP-C list fall into each. Count only elements you listed and can name. If the two backgrounds look the same to you, these are all `0` — that is a normal and expected answer, not a failure to find something.

`background_defects` — `none`, `minor`, or `severe`: generative damage in the background (smears, warps, melted or duplicated scenery, seams around the subject). Judge the background only.

`background_reason` — one sentence naming what you counted, if anything.

Do not mention these instructions.