

# MagicQuill V2: Precise and Interactive Image Editing with Layered Visual Cues

## Supplementary Material

### 1. Implementation Details

#### 1.1. Content Layer

**Training and Inference Details.** We integrate the content layer capability by fine-tuning the FLUX Kontext backbone with a LoRA adapter of rank  $r = 32$ . The model was trained for 9,000 steps on 8 H20 (140GB) GPUs, using the AdamW optimizer with a constant learning rate of  $1 \times 10^{-4}$ . For inference, the model performs 20 denoising steps, requiring approximately 30 seconds and 30GB of VRAM on a single H20 GPU.

**Completion Augmentation.** We utilize a dedicated object completion LoRA to restore foreground objects that are occluded in the source images. To train this completion model, we constructed a dataset of 3,000 high-resolution object images with white backgrounds, generated using Flux.1 Krea [8]. We simulate occlusions by applying random white brushstroke masks to these images. As demonstrated in Fig. 1, we train the model by providing pairs of complete, white-background objects (top row) and their counterparts with random brushstroke masks applied (bottom row).



Figure 1. Training data generation for the completion LoRA.

The LoRA is then fine-tuned on FLUX Kontext, with a rank  $r = 32$ , to reconstruct the original complete object from the masked input, guided by its descriptive caption. The training was conducted on 8 H20 (140GB) GPUs. We used the AdamW optimizer with a learning rate of  $1 \times 10^{-4}$  with 10 epochs of optimization. Qualitative results of this completion model are shown in Fig. 2, demonstrating its ability to restore occluded items extracted from scenes.

**Relight Augmentation.** We perform photometric augmentation using the SD1.5 [16] version of ICLight [23]. For each foreground object, we first generate a random light map. This map is used as the initial latent in an image-

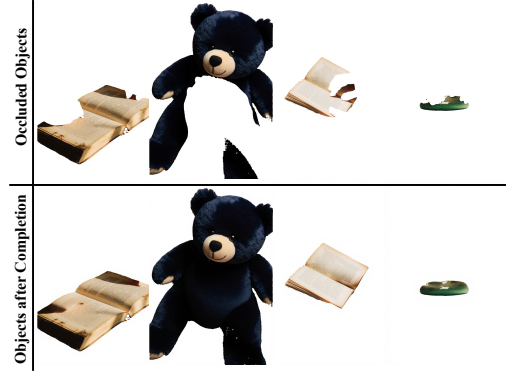


Figure 2. Qualitative results of the completion LoRA.

to-image relighting process. The light map’s color properties are randomly sampled: 50% are grayscale, 30% are low-saturation color, and 20% are high-saturation color, as illustrated in Fig. 3. This forces the model to learn robust lighting harmonization rather than just copy-pasting.

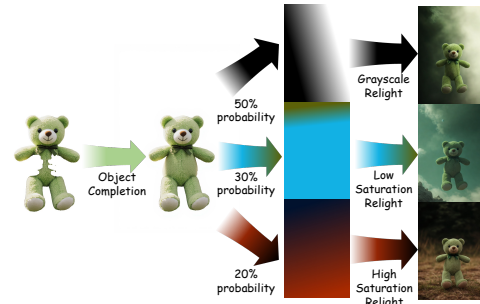


Figure 3. Overview of the relight augmentation pipeline. Completed objects are randomly relit using one of three light map categories to improve photometric robustness.

**Perspective Augmentation.** To simulate geometric mismatches, we apply a random perspective transformation. Given a foreground object  $F$  within its  $w \times h$  bounding box, we define its four corner coordinates as the source points  $P_{src} = \{(0, 0), (w, 0), (w, h), (0, h)\}$ . We sample a maximum perturbation ratio  $\rho \sim U(0.1, 0.3)$  and define the maximum displacement values  $\Delta x = w \cdot \rho$  and  $\Delta y = h \cdot \rho$ . We then generate four destination points  $P_{dst} = \{(x'_i, y'_i)\}_{i=1}^4$  by adding a random displacement  $\delta \sim U([- \Delta x, - \Delta y], [\Delta x, \Delta y])$  to each corresponding source point  $p_i \in P_{src}$ , clipping the result to stay within the original  $w \times h$  boundaries. A  $3 \times 3$  perspective transformation matrix  $H$  is computed by solving the homography that maps  $P_{src}$  to  $P_{dst}$ . This transformation  $H$  is then applied to both the foreground object  $F$  and its mask  $M_F$  to produce the geometrically augmented  $F'$  and  $M'_F$ .

**Resolution Augmentation.** To ensure robustness against varying input qualities, we simulate low-resolution inputs. For each foreground object, we sample a random scaling factor  $s \sim U(0.15, 0.9)$ . The object is downsampled to  $s$  of its original size and then upsampled back to its original resolution using bilinear interpolation.

## 1.2. Control Layer

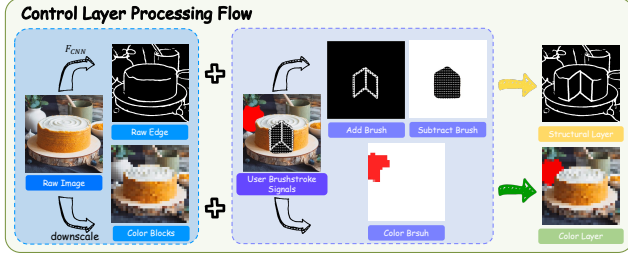


Figure 4. Illustration of converting control layers into edge and color block signals.

**Training and Inference Details.** We implement the structural, color, and spatial layers using the unified control module, each with a rank of  $r = 128$ . All visual control cues (e.g., edge maps, color maps) are resized to a fixed  $512 \times 512$  resolution before being processed. The control modules were trained for about 10,000 steps on 8 H20 (140GB) GPUs, using the AdamW optimizer with a constant learning rate of  $1 \times 10^{-4}$ . For inference, the model performs 20 denoising steps. When a control cue is active, this process requires approximately 45 seconds and 40GB of VRAM on a single H20 GPU. These control modules can be composed and activated simultaneously (e.g., using structural and color cues together for high-fidelity edits) and are also compatible with other LoRAs over FLUX Kontext weights. Following the interactive paradigm of MagicQuill V1 [12], our system automatically selects and provides the appropriate control cues based on the user’s brushstroke type.

**Structural Layer.** To train the structural layer, we use a large-scale, self-collected dataset, with all images resized to approximately  $1000 \times 1000$  pixels with detailed captions. For each image, we generate a structural map by randomly selecting one edge and lineart extractor from Canny [5], PidiNet [19], TEED [18], HED [21], and Informative Drawings [3]. As illustrated in Fig. 5, this ensures the model is robust to various styles of structural inputs.

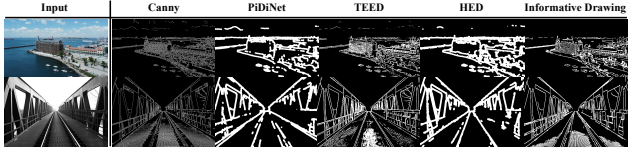


Figure 5. Visual results of different edge/lineart extractors.

The structural layer is trained for conditional generation, optimizing the model to approximate  $p(x|c, E)$ , where  $c$  is the text caption and  $E$  is the sampled edge map without the context image. We found that the structure-following capa-

bility, learned via conditional generation, robustly generalizes to the conditional editing task at inference time. Following the interactive paradigm of MagicQuill V1 [12], as shown in Fig. 4, we define binary masks  $M_{add}$  and  $M_{sub}$  corresponding to the user’s add and subtract brushstrokes, respectively. The subtract brush removes edges from the map, while the add brush introduces new edges. Let  $E$  denote the extracted edge map; the final control condition  $E_{cond}$  is formally computed as:

$$\begin{aligned} E_{sub} &= E \odot (1 - M_{sub}), \\ E_{cond} &= E_{sub} + M_{add} \odot (1 - E_{sub}). \end{aligned} \quad (1)$$

where  $\odot$  denotes the element-wise product. The resulting modified edge map  $E_{cond}$  serves as the precise geometric constraint for the generation process.

**Color Layer.** The training paradigm for the color layer mirrors that of the structural layer. We utilize the same dataset and train the model for conditional generation without the context image  $y$ . During training, the color condition map  $C$  is generated by downsampling to  $16 \times 16$  and resizing back to  $512 \times 512$ . This removes high-frequency details, forcing the model to learn the mapping from the palette to the final image.

At inference, we adopt the color brushstroke logics from MagicQuill V1 [12], as shown in Fig. 4. Each user color stroke is parameterized as a tuple  $(M_{color}, c, \alpha)$ , where  $M_{color}$  denotes the binary mask of the painted region,  $c$  specifies the target RGB color, and  $\alpha \in [0, 1]$  (default to 0.4) represents the stroke opacity. Let  $C$  denote the initial color map extracted from the source image (or a blank canvas); the updated control condition  $C_{cond}$  is computed via alpha blending:

$$C_{cond} = (1 - \alpha \cdot M_{color}) \odot C + \alpha \cdot M_{color} \cdot c, \quad (2)$$

where the specific color  $c$  is applied over the region defined by  $M_{color}$  with intensity  $\alpha$ . This blending mechanism allows users to achieve both subtle color tinting and opaque color replacement.

**Spatial Layer.** Unlike the structural and color layers, which can be trained on single images via conditional generation, the spatial layer (represented by a binary mask  $M$ ) requires training on local editing pairs  $(I_{src}, I_{tgt}, c, M)$ . The goal is to constrain the model’s generative changes strictly within the user-specified region. To achieve this, we construct a large-scale dataset via a rigorous Self-Distillation Pipeline. We leverage Qwen2.5-VL-72B [1] to propose a set of plausible local editing instructions  $c$  for each  $I_{src}$ , which are then executed by the base FLUX Kontext [2] model to produce the edited target image  $I_{tgt}$ .

To derive robust masks  $M$  from these generated pairs, we implement a two-stage difference extraction pipeline. First, a pre-screening step is performed using standard pixel-wise differences. We evaluate the edit magnitude across multiple threshold parameters, filtering out samples



Figure 6. Visualization of the data filtering strategy for the Spatial Layer. We calculate the area ratio of the convex hull covering the changed regions between the source and target images. The pipeline automatically rejects failed edits with no significant changes (Row 1) or excessive global changes (Row 2), retaining only high-quality local editing pairs (Rows 3 & 4) for training.

where the changing hull area has ratio outside the range  $[0.001, 0.75]$  to eliminate insignificant or excessive global edits, as illustrated in Fig. 6. For the retained pairs, the final mask is generated in the CIELAB color space. We compute the Euclidean distance between the aligned Lab vectors to capture perceptual photometric differences, offering superior robustness to compression artifacts compared to standard RGB. To prevent mask fragmentation, we compute the Convex Hull of the detected change regions, ensuring the mask covers the entire semantic object. Finally, this hull is refined using rolling-circle smoothing to simulate the continuous topology of natural human brushstrokes.

To further augment the model’s capability for object removal, we follow Jiang et al. [7] to create a synthetic dataset. This involves randomly extracting SAM-based foregrounds and pasting them onto arbitrary backgrounds, treating the pre-paste image as the ground truth target.

## 2. User Study for Content Layer

Standard quantitative metrics often fail to capture perceptual nuances in image composition, such as lighting integration and occlusion handling. To evaluate the human-perceived quality of our method, we conducted a user preference study focusing on the content layer ( $L_{fg}$ ).

**Setup and Methodology.** We recruited 30 participants to evaluate 10 editing scenarios sampled from our test set. For each scenario, participants compared results from six methods: Nano Banana [4], Insert Anything [17], FLUX Kontext [2], Kontext + Put It Here LoRA [6], Qwen-Image-Edit [20], and ours. Participants selected the single “best” result based on foreground preservation, visual coherence, and interaction plausibility.

**Results.** As shown in Fig. 7, **MagicQuill V2 achieved**

User Preferences: 30 users

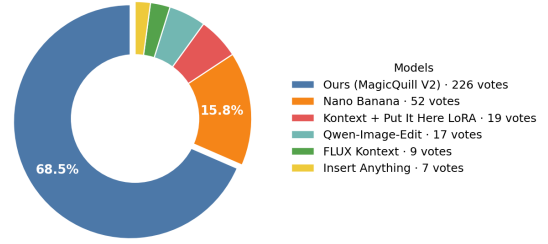


Figure 7. **User Preference Distribution.** MagicQuill V2 (Ours) dominates with **68.5%** of the votes, significantly outperforming the strongest baseline, Nano Banana (15.8%).

**a dominant preference rate of 68.5%** (226 votes), significantly outperforming the second-best model, Nano Banana (15.8%, 52 votes). Qualitative feedback indicates that our method consistently delivers results that are physically grounded and harmoniously integrated.

## 3. Comparison with additional baselines

In order to further contextualize our contributions within the recent literature, we extend our evaluation to include several state-of-the-art controllable image generation and interactive editing frameworks: ControlAR [10], EditAR [15], AnyEdit [22], and Generative Photomontage (GPM) [11].

We quantitatively compare the performance of our method (specifically utilizing the Edge Layer) against ControlAR, EditAR, and AnyEdit. The evaluation is conducted on the Pico-Banana-400k benchmark, as demonstrated in Tab. 1, our approach significantly outperforms all competing baselines across every metric.

Table 1. Quantitative comparison against more baselines.

Model	$L_1 \downarrow$	$L_2 \downarrow$	CLIP-I $\uparrow$	DINO $\uparrow$	LPIPS $\downarrow$
ControlAR	0.250	0.110	0.776	0.670	0.409
EditAR	0.194	0.073	0.831	0.741	0.485
AnyEdit (Scribble)	0.183	0.068	0.817	0.670	0.552
AnyEdit (Sketch)	0.170	0.062	0.846	0.729	0.371
Ours (Edge Layer)	<b>0.107</b>	<b>0.030</b>	<b>0.938</b>	<b>0.909</b>	<b>0.317</b>

We qualitatively compare our layered composition approach against Generative Photomontage (GPM). GPM typically requires multiple, complex conditioning steps to achieve desired edits (e.g., precise references and structural scribbles). As illustrated in Fig. 8, GPM requires three separate user inputs to guide the local edit. In contrast, our method yields a highly seamless and context-aware blending result using significantly simpler inputs.

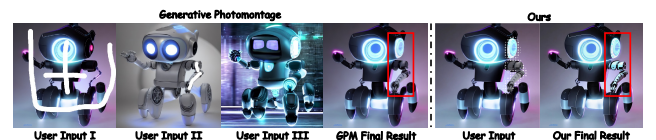


Figure 8. Qualitative comparison with Generative Photomontage.



## 4. Limitation

Despite the significant advancements in precise controllability, MagicQuill V2 has several limitations that present avenues for future research.

**Inference Latency.** As a diffusion transformer-based model utilizing a heavy backbone (12B) and multiple control adapters, our system prioritizes generation quality over speed. A single edit requires approximately 30-45 seconds on a high-end H20 GPU. This latency creates friction in the “interactive” workflow, as users must wait for feedback. Future work could explore consistency distillation [13, 14] or quantization techniques [9] to achieve near real-time performance for interaction.

**Conflict between Modalities.** While our layered architecture enables flexible composition, the freedom to stack multiple control layers introduces the potential for contradictory inputs. Conflicts can arise not only between semantic text and visual cues (e.g., a prompt describing a “circle” while the structural layer dictates a “square”) but also between distinct visual layers. For instance, a user might provide a detailed structural layer (edge map) that outlines complex geometry, while simultaneously applying a color layer that contradicts those boundaries. Currently, the model attempts to reconcile these disjointed signals based on learned priors and the per-layer control strength  $\sigma$ . However, when cues are heavily conflicting, this implicit resolution can result in artifacts, where the model fails to satisfy mutually exclusive constraints.

**Articulation and Rotation.** While our framework provides versatile multi-modal control, we acknowledge certain limitations regarding extreme spatial transformations. Specifically, the content layer inherently tends to preserve the original pose and orientation of the input assets. This behavior is a byproduct of prioritizing pixel-alignment during the composition process. However, users are not strictly bound by this constraint. As illustrated in Fig. 9, users can achieve precise re-articulation and rotation by utilizing the structural layer to guide the specific structural changes. Alternatively, users can leverage semantic references (e.g., pose conditions) to override the default pose preservation.

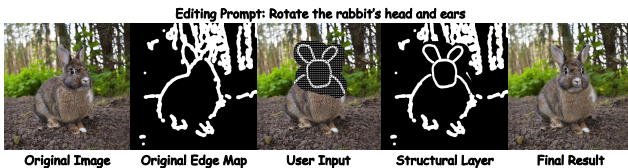


Figure 9. Utilizing the structural layer for re-articulation (rotating the rabbit’s head).

## 5. Future Directions

While our current framework excels with pixel-aligned controls (such as structural boundaries and color maps), a natu-

ral progression for interactive editing is the incorporation of high-level semantic conditions. Our Unified Control Module is inherently modular, allowing plug-and-play integration of off-the-shelf LoRAs as **semantic layers**. This enables robust reference-driven control, such as pose and appearance, without retraining, as shown in Fig. 10.

Integrating semantic layers directly addresses current limitations regarding complex spatial transformations. As noted during our evaluations, the Content Layer inherently tends to preserve the pose and orientation of the original asset to prioritize pixel-alignment. By extending our system with Semantic Layers (e.g., a pose-guided LoRA), users can easily achieve precise re-articulation and rotation of foreground subjects.



Figure 10. Integration of semantic layers to provide references.

## 6. Open Source Commitment

To facilitate reproducibility and future research, we are committed to fully open-sourcing our project. We will publicly release the complete codebase, including training scripts, model checkpoints, and the interactive system, including the user interface.



## References

- [1] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibao Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025. 2
- [2] Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, et al. Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv e-prints*, pages arXiv–2506, 2025. 2, 3
- [3] Caroline Chan, Fredo Durand, and Phillip Isola. Learning to generate line drawings that convey geometry and semantics. 2022. 2
- [4] Google Deepmind. Gemini 2.5 flash image (nano banana). <https://aistudio.google.com/models/gemini-2-5-flash-image>, 2025. 3
- [5] Lijun Ding and Ardeshtir Goshtasby. On the canny edge detector. *Pattern recognition*, 34(3):721–725, 2001. 2
- [6] Futurlunatic. Put it here: Kontextv4 lora. <https://civitai.com/models/1808575/put-it-herekontextv4>, 2025. 3
- [7] Longtao Jiang, Zhendong Wang, Jianmin Bao, Wengang Zhou, Dongdong Chen, Lei Shi, Dong Chen, and Houqiang Li. Smarteraser: Remove anything from images using masked-region guidance. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 24452–24462, 2025. 3
- [8] Black Forest Labs Krea. Flux krea. <https://github.com/krea-ai/flux-krea>, 2025. 1
- [9] Muyang Li\*, Yujun Lin\*, Zhekai Zhang\*, Tianle Cai, Xiuyu Li, Junxian Guo, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. Svdquant: Absorbing outliers by low-rank components for 4-bit diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025. 4
- [10] Zongming Li, Tianheng Cheng, Shoufa Chen, Peize Sun, Haocheng Shen, Longjin Ran, Xiaoxin Chen, Wenyu Liu, and Xinggang Wang. Controlar: Controllable image generation with autoregressive models. *arXiv preprint arXiv:2410.02705*, 2024. 3
- [11] Sean J Liu, Nupur Kumari, Ariel Shamir, and Jun-Yan Zhu. Generative photomontage. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 7931–7941, 2025. 3
- [12] Zichen Liu, Yue Yu, Hao Ouyang, Qiuyu Wang, Ka Leong Cheng, Wen Wang, Zhiheng Liu, Qifeng Chen, and Yujun Shen. Magicquill: An intelligent interactive image editing system. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 13072–13082, 2025. 2
- [13] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference, 2023. 4
- [14] Simian Luo, Yiqin Tan, Suraj Patil, Daniel Gu, Patrick von Platen, Apolinário Passos, Longbo Huang, Jian Li, and Hang Zhao. Lcm-lora: A universal stable-diffusion acceleration module. *arXiv preprint arXiv:2311.05556*, 2023. 4
- [15] Jiteng Mu, Nuno Vasconcelos, and Xiaolong Wang. Editor: Unified conditional generation with autoregressive models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 7899–7909, 2025. 3
- [16] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 1
- [17] Wensong Song, Hong Jiang, Zongxing Yang, Ruijie Quan, and Yi Yang. Insert anything: Image insertion via in-context editing in dit. *arXiv preprint arXiv:2504.15009*, 2025. 3
- [18] Xavier Soria, Yachuan Li, Mohammad Rouhani, and Angel D. Sappa. Tiny and efficient model for the edge detection generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 1364–1373, 2023. 2
- [19] Zhuo Su, Wenzhe Liu, Zitong Yu, Dewen Hu, Qing Liao, Qi Tian, Matti Pietikäinen, and Li Liu. Pixel difference networks for efficient edge detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5117–5127, 2021. 2
- [20] Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025. 3
- [21] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403, 2015. 2
- [22] Qifan Yu, Wei Chow, Zhongqi Yue, Kaihang Pan, Yang Wu, Xiaoyang Wan, Juncheng Li, Siliang Tang, Hanwang Zhang, and Yueting Zhuang. Anyedit: Mastering unified high-quality image editing for any idea. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 26125–26135, 2025. 3
- [23] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Scaling in-the-wild training for diffusion-based illumination harmonization and editing by imposing consistent light transport. In *The Thirteenth International Conference on Learning Representations*, 2025. 1