FocusFlow: Localizing Edits via Spatial Velocity Decomposition

Mohamed Jalal BAIM
mohamed.baim@polytechnique.edu

Othmane BELHAJ
othmane.belhaj@polytechnique.edu

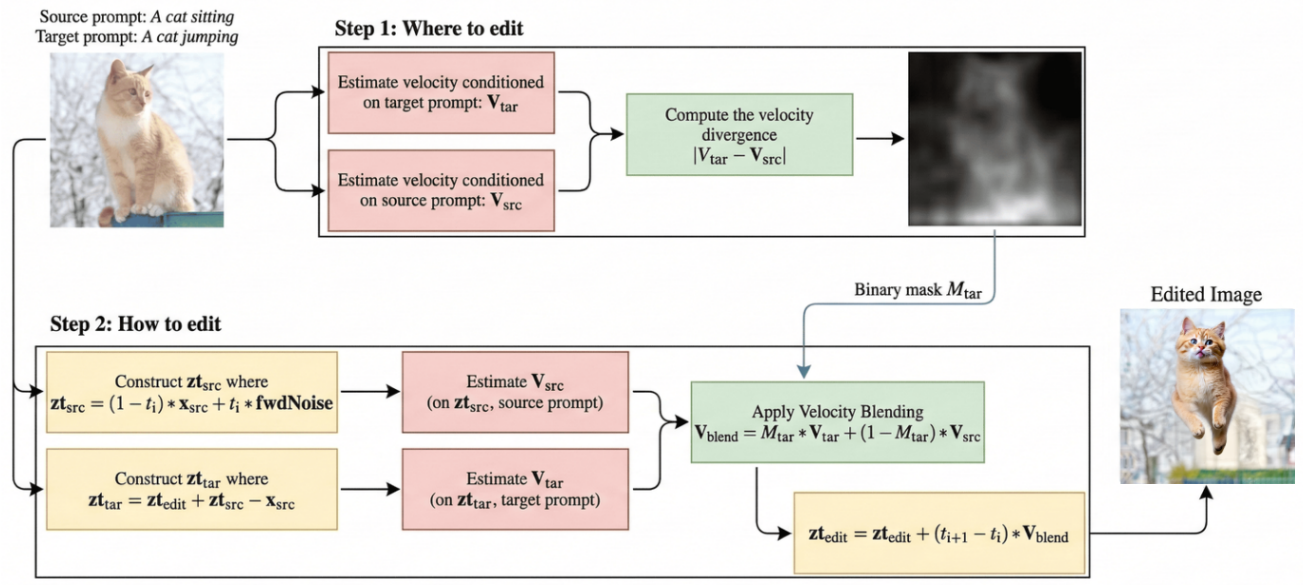

Figure 1. **FocusFlow Pipeline Overview.** Two-stage editing process: (1) Where to edit ? Mask M based on velocity difference; (2) How to edit ? Editing approach$V = M * V_{\text{tar}} + (1 - M) * V_{\text{src}}$ solves the ODE for region specific edits.

Abstract

*Editing real images with natural language prompts remains a central challenge in computer vision. In this report, we present **FocusFlow**, a localized editing framework for flow models. While existing inversion-free approaches like FlowEdit offer good edits in term of quality and structure preservation, they often suffer when it comes to large and complex semantic edits. FocusFlow addresses this by introducing **Spatial Velocity Decomposition**, a technique that automatically identifies target regions by analyzing the variance in velocity predictions between source and target prompts. Our approach implements a two-stage method that adaptively blends velocity fields, ensuring the background follows the source trajectory while the foreground undergoes target transformations. We demonstrate that FocusFlow achieves superior text adherence for complex edits compared to standard FlowEdit, all while remaining inversion-free and requiring no user-provided masks.*

1. Introduction

The emergence of text-to-image diffusion models has revolutionized content creation, yet precise editing of real-world images remains difficult. A key trade-off exists between the fidelity of the edit and the preservation of the original image structure. Early methods relied on expensive inversion processes, which are computationally expensive, require gaussian noise bottleneck, and introduce distortions during the inversion process.

Recently, flow matching models like Stable Diffusion 3 (SD3) and Flux have gained popularity due to their efficient ODE-based sampling. **FlowEdit** [1] introduced an inversion-free editing paradigm that uses a bridge between source and target trajectories. However, it struggles when dealing with significant semantic edits.

Our proposed method, **FocusFlow**, overcomes these lim-

itations by localizing the edit through a region-based approach. By leveraging this spatial decomposition approach, FocusFlow integrates a new region-based velocity field during the ODE process.

Our contributions are threefold:

- We introduce **Spatial Velocity Decomposition**, providing a mechanism to localize flow-based edits without manual intervention.
- We propose a modified ODE solver for flow models that performs a weighted sum of source and target velocities.
- We provide a comprehensive evaluation on various editing tasks, with a meaningful comparison between different methods.

The full implementation is available on GitHub[1].

2. Methodology

Couairon et al. proposed DiffEdit [2], a method designed for semantic image editing using text-conditioned diffusion models without requiring manual user masks. It operates on the principle that a diffusion model will yield different noise estimates when conditioned on different text prompts.

By contrasting these predictions, DiffEdit can automatically identify the specific regions of the image that need to be modified to match a target query. Then push the model to focus its edits only on these relevant regions so they guarantee strong edits as well as structure preservation.

The main challenge we encounter with FlowEdit [1] is that its intrinsic ability to preserve the original image identity hinders its capacity to apply complex edits. To address this, we propose giving the model more freedom in the targeted regions while constraining it in the non-targeted areas.

Following this reasoning, we suppose that the Flow models will also generate distinct velocity fields when conditioned on different text prompts. By contrasting these velocity estimations, we can derive a binary soft mask that separates the image into targeted and non-targeted regions.

The targeted regions follow the target velocity field (V_{tar}), granting the model the semantic freedom to modify complex structures according to the new prompt. Meanwhile, the non-desired regions are anchored to the source velocity field (V_{src}), which forces the model to maintain the original image's identity and background. The pipeline 1 shows the whole approach in details.

At the end we tend to solve a direct ODE path where the update direction at each step is a weighted average of two velocities. By applying the mask M, we define the blended velocity as V_{blend}. This ensures that the flow matching process is only "freed" to change pixels within the masked area.

Henceforth, we call this new approach FocusFlow.

[1] github.com/mohamedjalalbaim/FocusFlow

2.1. Step 1: Where to edit ?

Instead of relying on user-provided masks, FocusFlow generates $\mathbf{M}$ dynamically.

Starting from the assumption that flow models will also generate distinct velocity fields when conditioned on different text prompts. Analyzing the absolute difference $\Delta V = |V_t^{\text{target}} - V_t^{\text{source}}|$ at moderate noise levels ($t \approx 0.5$), will help us identify pixels where the prompts disagree most, and then create a binary mask that shows separates targeted and non-targeted regions.

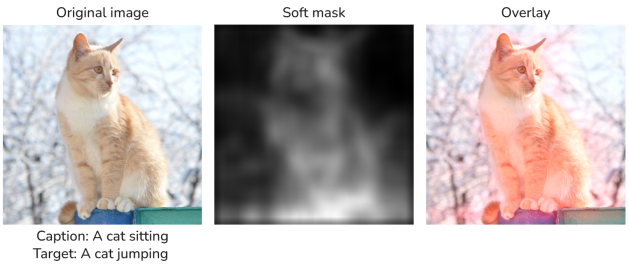

Figure 2. **Step 1:**Mask generation.

2.2. Step 2: How to edit ?

FocusFlow uses **Spatial Velocity Decomposition** to spacially separate the velocity field. As defined in our implementation, we compute a final velocity V_t^{blended} at each step:

$$V_t^{\text{blended}} = \mathbf{M} \odot V_t^{\text{target}} + (1 - \mathbf{M}) \odot V_t^{\text{source}} \quad (1)$$

where $\mathbf{M}$ is a mask, V_t^{target} is predicted using the target prompt, and V_t^{source} using the source prompt.

A theoritical overview is discussed in Appendix A.

2.3. FocusFlow Algorithm

Algorithm 1 Simplified FocusFlow Algorithm

Input: real image X^{src}, source prompt p_{src}, target prompt p_{tar}, $\{t_i\}_{i=0}^{T}$, $n_{\max}$, M mask.
Initialize: $z_t^{edit} \leftarrow x_{src}$
for $i = n_{\max}$ **to** 1 **do**
  $\epsilon \sim \mathcal{N}(0, I)$
  $z_t^{src} \leftarrow (1 - t_i)x_{src} + t_i\epsilon$
  $z_t^{tar} \leftarrow z_t^{edit} + z_t^{src} - x_{src}$
  Compute velocities (V_t^{src}, V_t^{tar})
  $V_{blend} \leftarrow M \cdot V_t^{tar} + (1 - M) \cdot V_t^{src}$
  $z_t^{edit} \leftarrow z_t^{edit} + (t_{i-1} - t_i)V_{blend}$
end for
return $(x_{out}, M_{soft}, M_{bin})$

The algorithm takes as input a source image X^{src}, a source prompt p_{src}, a target prompt p_{tar}, and a spatial mask M. It

starts from the original image and iterates backwards from n_{max} to 1.

At each step, it constructs both a noisy source latent z_t^{src} and a corresponding target latent z_t^{tar}, and uses them to compute two velocity fields: V_t^{src}, conditioned on the source prompt, and V_t^{tar}, conditioned on the target prompt.

The key step is the spatial blending of these two velocities using the mask M: inside the edited region, the model follows V_t^{tar} to apply the desired changes, while outside it follows V_t^{src} to preserve the original content.

The edited latent is then updated using this blended velocity, and the process repeats until the final image is produced.

Additionally, to reduce the stochasticity introduced by the random noise sampling, the velocity estimates can be averaged over n_{avg} independent samples at each timestep, theoritically defined by the expectation over the transport equation in Appendix A, leading to a more stable and consistent editing result.

3. Experiments & Results

3.1. Experimental Setup

Dataset. We conduct our experiments on images from the DIV2K dataset. For the quantitative evaluation, due to computational constraints, we use a subset of 20 images (2 prompts each), each at a resolution of 512×512.

Implementation Details. We use the official SD3 weights available on Hugging Face. The hyperparameters are set to $T = 50$ steps and $n_{\max} = 33$ (following the FlowEdit configuration). Since SD3 relies on Classifier-Free Guidance (CFG) for text conditioning, the source and target guidance scales are set to 3.5 and 16.5, respectively.

Methods Compared. We compare FocusFlow against text-to-image (T2I) editing approaches for flow models using the same SD3 backbone. Specifically, we evaluate FlowEdit and SDEdit with the standard hyperparameters reported in their original papers.

3.2. Resultas

Qualitative results. Figure 3 shows a visual comparison between FocusFlow, SDEdit, and FlowEdit on a variety of editing tasks. SDEdit produces noticeable artifacts and struggles to preserve the background, resulting in images that look unnatural and inconsistent with the original. FlowEdit maintains better background quality, but its strong identity-preservation tendency limits its ability to apply significant changes, making it less effective for edits that require large structural or semantic modifications. FocusFlow, on the other hand, handles a wide range of editing scenarios successfully. It performs well on large semantic edits, such as changing the style of a castle or the color of a van, as well as on challenging pose edits, such as making a cat jump or a bear stand upright. In all cases, the background and non-targeted regions remain clean and faithful to the original image. These results suggest that FocusFlow strikes a good balance between editing strength and image preservation, outperforming the two baselines both in edit quality and visual coherence.
More results and comparisons are shown in Appendix B, C.

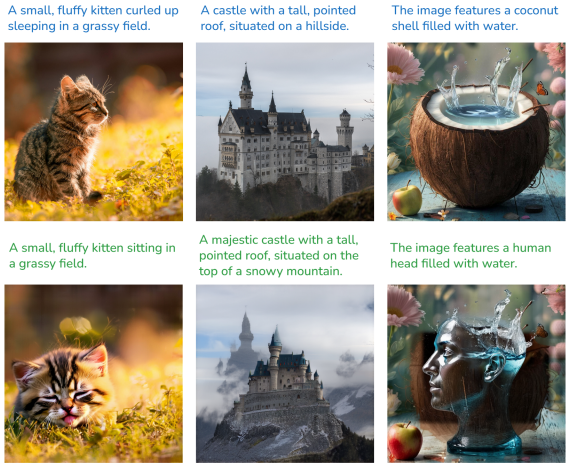

Figure 4. **FocusFlow results.** The first row shows the original image and the source prompt (in blue). The last row shows the edited image and the target prompt (in green).

Quantitative results. We numerically evaluate the results of FlowEdit and the other methods using LPIPS [3] to measure the semantic structure preservation (lower is better) and CLIP [4] to assess text adherence (higher is better).

As summarized in Table 1, FocusFlow achieves a significantly higher CLIP score (0.296) compared to standard FlowEdit (0.241), indicating better adherence to the target prompt.

While the LPIPS distance is slightly higher, this is expected as FocusFlow enables larger semantic changes within the target region while maintaining the identity of the original image as possible.

Notably, both FocusFlow and FlowEdit outperform the SDEdit baseline in structural consistency. However, FlowEdit remains consistent in terms of structure preservation due to its conservative approach of editing.

Table 1. Quantitative comparison on 40 test cases.

Method	n_{cases}	CLIP ↑	LPIPS ↓
SDEdit	40	0.218	0.466
FlowEdit	40	0.241	**0.196**
FocusFlow	40	**0.296**	0.289

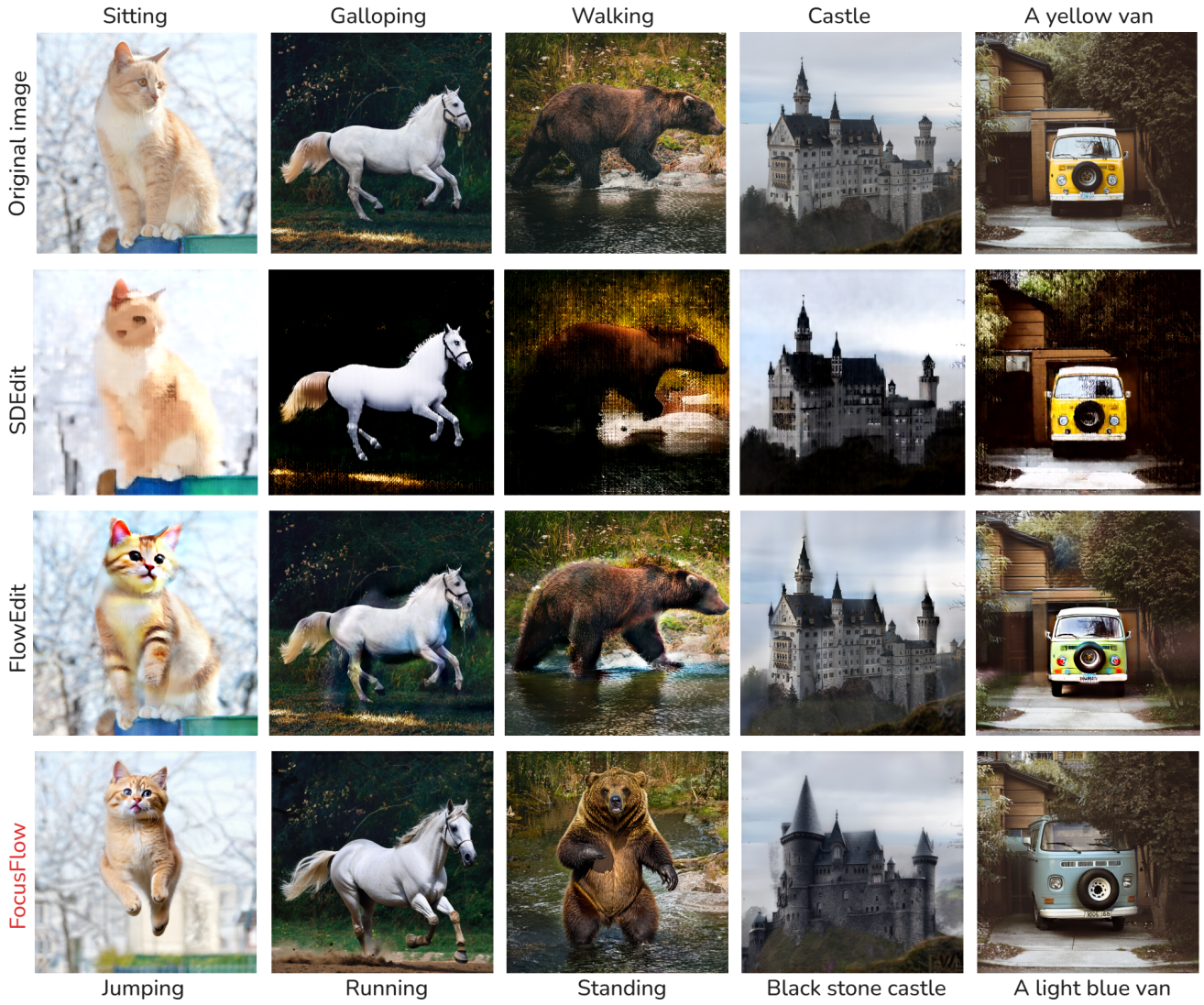

Figure 3. **Qualitative results** comparison between SDEdit, FlowEdit, and FocusFlow, from our evaluation set demonstrating complex semantic edits including background replacement and pose editing while preserving non-target regions.

4. Discussion & Conclusion

In this work, we present **FocusFlow**, a localized, inversion-free editing framework built on FlowEdit. The method introduces in Fig. 1 a two-step pipeline : (1) automatic mask generation from source/target velocities, (2) image editing using a blended velocity combining target and source velocities through a mask during ODE integration. This design allows strong semantic transformations in tqrgeted regions while preserving the identity of non-targeted regions.

Our main contribution is the integration of mask-guided velocity blending directly into the editing trajectory, enabling spatial control without manual masks or explicit inversion. Across 40 evaluation cases, FocusFlow improves prompt alignment over FlowEdit, as shown in Tab. 1, (CLIP: 0.296 vs. 0.241), while outperforming SDEdit baseline. Qualitative comparisons further show that FocusFlow handles challenging edits (e.g., pose and background changes) more effectively than both baselines. Also our approach remains a zero-cost extension of FlowEdit: it reuses the same velocity computations and simply routes them spatially through the mask, with no additional model calls.

We also identify failure cases, as shown in Appendix D: when edits require very large structural changes or the addition of complex new semantic attributes, FocusFlow can still fail even with hyperparameter tuning. These limitations highlight an important direction for future work: improving robustness for extreme edits while retaining the current trade-off between edit strength and content preservation.

References

[1] Kulikov, V., Kleiner, M., Huberman-Spiegelglas, I., Michaeli, T. *FlowEdit: Inversion-Free Text-Based Editing Using Pre-Trained Flow Models.* ICCV 2025. 1, 2

[2] Couairon, G., Verbeek, J., Schwenk, H., Cord, M. *DiffEdit: Diffusion-Based Semantic Image Editing with Mask Guidance.* arXiv preprint arXiv:2210.11427, 2022. 2

[3] Zhang, R., Isola, P., Efros, A. A., Shechtman, E., Wang, O. extitThe Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 586–595, 2018. 3

[4] C. Qi, X. Cun, Y. Zhang, C. Lei, X. Wang, Y. Shan, and Q. Chen, "FateZero: Fusing Attentions for Zero-shot Text-based Video Editing," Proc. IEEE/CVF International Conference on Computer Vision (ICCV), pp. 15886–15896, 2023. Available: https://api.semanticscholar.org/CorpusID:257557738 3

A. FocusFlow: Theoritical overview

Both FocusFlow and FlowEdit are based on the theory of rectified flow models, where data generation is maintained by an ordinary differential equation (ODE).

$$\frac{dZ_t}{dt} = V(Z_t, t), \qquad t \in [0, 1],$$

trained such that the marginals satisfy

$$Z_t \sim (1-t)X_0 + tX_1, \qquad X_1 \sim \mathcal{N}(0, I).$$

This property guarantees that all intermediate states lie on the data noise manifold seen during training.
For text-based editing, let the velocity the vector field that tells how an image representation moves over time along the generative path.

$$V^{\text{src}}(x, t) = V(x, t, c_{\text{src}}), \qquad V^{\text{tar}}(x, t) = V(x, t, c_{\text{tar}}),$$

and consider inversion-based editing defined by

$$\frac{dZ_t^{\text{src}}}{dt} = V^{\text{src}}(Z_t^{\text{src}}, t), \; Z_0^{\text{src}} = X_{\text{src}},$$

$$\frac{dZ_t^{\text{tar}}}{dt} = V^{\text{tar}}(Z_t^{\text{tar}}, t), \; Z_1^{\text{tar}} = Z_1^{\text{src}}.$$

FlowEdit shows that this procedure implicitly defines a direct path

$$Z_t^{\text{inv}} = Z_0^{\text{src}} + Z_t^{\text{tar}} - Z_t^{\text{src}},$$

satisfying $Z_1^{\text{inv}} = X_{\text{src}}$ and $Z_0^{\text{inv}} = Z_0^{\text{tar}}$. So by differenting, we get the direct ODE

$$\frac{dZ_t^{\text{inv}}}{dt} = V^{\text{tar}}(Z_t^{\text{tar}}, t) - V^{\text{src}}(Z_t^{\text{src}}, t).$$

Due to the shared noise content of Z_t^{src} and Z_t^{tar}, the velocity difference removes noise components.
FlowEdit generalizes this path by replacing Z_t^{src} with a random process,

$$\hat{Z}_t^{\text{src}} = (1-t)Z_0^{\text{src}} + tN_t, \qquad N_t \sim \mathcal{N}(0, I),$$

and defining $\hat{Z}_t^{\text{tar}} = Z_t^{\text{FE}} + \hat{Z}_t^{\text{src}} - Z_0^{\text{src}}$. The resulting ODE is

$$\frac{dZ_t^{\text{FE}}}{dt} = \mathbb{E}\Big[V^{\text{tar}}(\hat{Z}_t^{\text{tar}}, t) - V^{\text{src}}(\hat{Z}_t^{\text{src}}, t) \,\Big|\, Z_0^{\text{src}}\Big], \qquad Z_1^{\text{FE}} = Z_0^{\text{src}}.$$

This construction guarantees in-distribution model queries and defines a well-posed transport ODE. While FlowEdit does not guarantee optimal transport or exact matching of the target distribution, it induces a shorter transport path than inversion, explaining its improved structure preservation.

FocusFlow extends this idea by introducing a spatial decomposition of the velocity field. Instead of applying the target velocity globally, we blend V^{tar} and V^{src} using a binary mask M:

$$\frac{dZ_t^{\text{FF}}}{dt} = \mathbb{E}\Big[M \odot V^{\text{tar}}(\hat{Z}_t^{\text{tar}}, t) + (1-M) \odot V^{\text{src}}(\hat{Z}_t^{\text{src}}, t) \,\Big|\, Z_0^{\text{src}}\Big]$$

Inside the masked region, the model follows V^{tar} and is free to apply the desired edit. Outside it, the model follows V^{src} and preserves the original content. This simple modification retains the theoretical guarantees of FlowEdit while giving the model the freedom needed to perform stronger and more localized edits.

B. Pose Edits

Based on Figure 5, FocusFlow successfully performs pose edits, unlike FlowEdit, which struggles with this task as reported in the original paper.

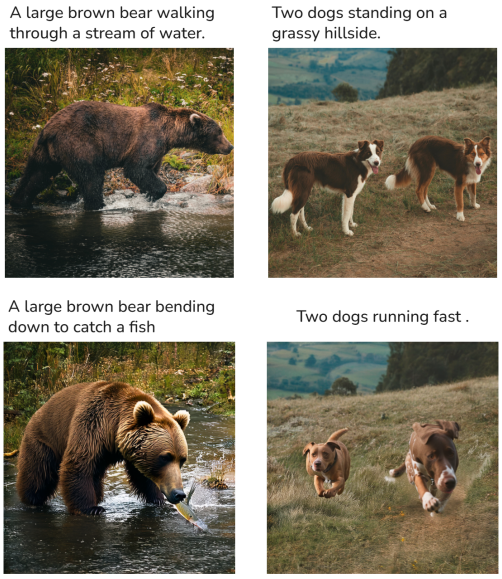

Figure 5. Edits focusing on pose changes.

C. Background Edits

As shown in Figure 6, FocusFlow successfully performs background edits, unlike FlowEdit, which struggles with this task as reported in the original paper.

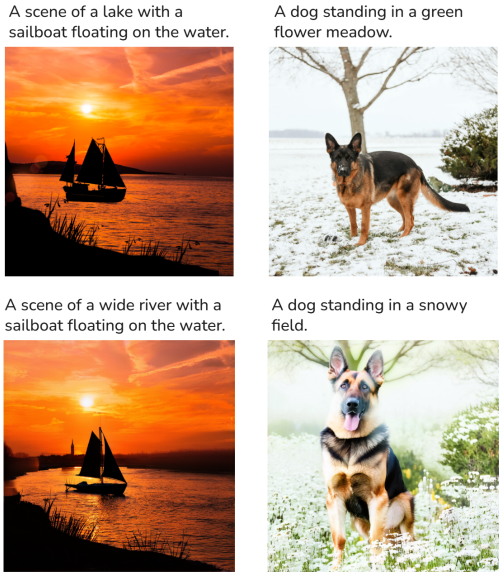

Figure 6. Edits focusing on background changes.

D. Failed examples

Figure 7 shows several failure cases of FocusFlow. When edits are highly complex and require large image modifications (e.g., adding new semantic attributes or editing large regions), FocusFlow may fail to produce satisfactory results, even after hyperparameter tuning (higher n_{max}, higher n_{min}, and larger T).

Figure 7. Edits focusing on background changes.

E. Computational Complexity

FocusFlow introduces minimal computational overhead compared to FlowEdit. At each timestep, FlowEdit requires two forward passes through the model to compute V^{tar} and V^{src}. FocusFlow follows the exact same structure and therefore requires the same number of forward passes. The only additional operation is the element-wise masking step:

$$V_{\text{blend}} = M \odot V^{\text{tar}} + (1 - M) \odot V^{\text{src}}$$

which is a simple pixel-wise multiplication and addition, with complexity $\mathcal{O}(H \times W)$. This is negligible compared to the cost of a single forward pass through the diffusion model.

Overall, FocusFlow can be seen as a zero-cost extension of FlowEdit: it reuses the same velocity computations and simply routes them spatially through the mask, with no additional model calls or architectural changes required.

To perform our experiments, We relied on multiple cloud GPU providers, mainly Kaggle and Google Colab. On Kaggle, experiments were conducted using an NVIDIA P100 GPU, while on Colab an NVIDIA H100 GPU was used. Due to computational constraints, We could only use 20 images (2 prompts each), also experiments were limited to Stable Diffusion 3. The FLUX model is significantly more demanding and we could not execute it reliably on the available GPU resources.

Table 2 summarizes the computational resources and runtimes used for these experiments.

Platform	GPU	Memory	Runtime
Kaggle	NVIDIA P100	16 GB	30 hours
Google Colab	NVIDIA H100	22 GB	15 hours

Table 2. Summary of GPU resources and computation times used for the experiments.