

ControlVP: Interactive Geometric Refinement of AI-Generated Images with Consistent Vanishing Points

Supplementary Material

In this supplementary material, we provide the following:

- A. Detailed algorithm for VP loss computation.
- B. Graphical user interface for ControlVP.
- C. Evaluation dataset creation details.
- D. User study details.
- E. Additional qualitative results.

A. Implementation of Vanishing Point Loss

The main paper introduces our Vanishing Point (VP) Loss and its conceptual design. Here, we provide the complete algorithmic implementation in detail and implementation considerations.

Our VP loss computation involves three key phases: edge detection, angular analysis, and weighted aggregation. The algorithm operates on pixel-level edge information to enforce geometric consistency between predicted and ground truth images.

The algorithm takes three inputs: \mathbf{x}_0 is the clean ground truth image before noise addition, $\hat{\mathbf{x}}_0$ is the predicted clean image computed from the noisy latent and predicted noise, and $\{VP_i\}_{i=1}^N$ are the vanishing point coordinates annotated in the dataset for \mathbf{x}_0 .

Edge Detection: We employ Sobel operators to extract both magnitude and direction information from images.

Angular Analysis: For each pixel, we compute the angle between the edge direction and each vanishing point direction. Unlike previous approaches that apply uniform weighting across all edges, our method focuses on edges that are closely aligned with VP directions through threshold-based filtering.

Weighted Aggregation: We introduce a sigmoid weighting mechanism $\sigma(\theta_{thresh} - \theta_i)$ that provides differentiable weighting while maintaining selectivity for relevant edges. This differentiable formulation enables end-to-end training through backpropagation.

B. Graphical User Interface for ControlVP

Please watch the supplementary video for a demonstration of our interface. We developed an intuitive graphical user interface (GUI) shown in Figure 9 to make the correction process accessible to users without technical expertise. The correction process is straightforward: (1) a single tap to select the VP position, (2) two taps to mark the endpoints of the correct building outline that should align with the VP, and (3) two taps to mark the endpoints of the original misaligned outline. With a minimum of five taps per inconsis-

ALGORITHM 1: VP Loss Computation

```

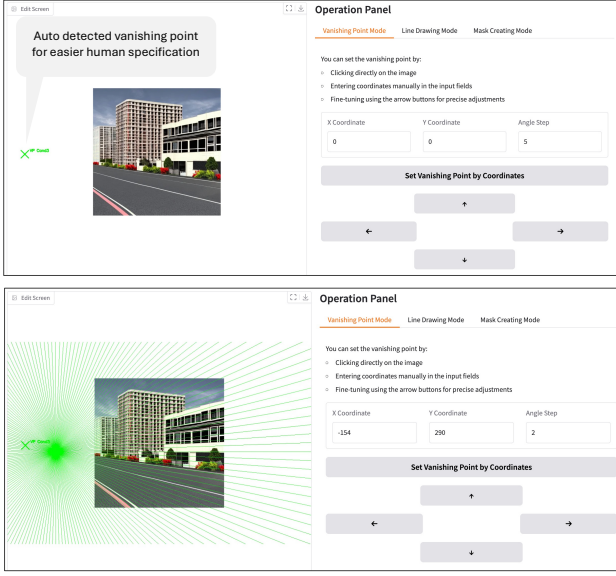
1: Function ComputeVPLoss( $\hat{\mathbf{x}}_0, \mathbf{x}_0, \{VP_i\}_{i=1}^N$ )
2: Input:
3:    $\hat{\mathbf{x}}_0$ : Predicted image
4:    $\mathbf{x}_0$ : Ground truth image
5:    $\{VP_i\}_{i=1}^N$ : Set of vanishing point positions
6: Output:  $\mathcal{L}_{VP}$ : Vanishing Point Loss
7: Step 1: Compute edge using Sobel filter
8:  $(g_x^{pred}, g_y^{pred}) \leftarrow \text{Sobel}(\hat{\mathbf{x}}_0)$ 
9:  $(g_x^{gt}, g_y^{gt}) \leftarrow \text{Sobel}(\mathbf{x}_0)$ 
10:  $M^{pred} \leftarrow \sqrt{(g_x^{pred})^2 + (g_y^{pred})^2}$ 
11:  $M^{gt} \leftarrow \sqrt{(g_x^{gt})^2 + (g_y^{gt})^2}$ 
12: Step 2: Compute edge direction scores
13: for each pixel  $(u, v)$  do
14:   Compute edge direction vectors
15:    $\vec{d}^{pred} \leftarrow (-g_y^{pred}, g_x^{pred}) / M^{pred}$ 
16:    $\vec{d}^{gt} \leftarrow (-g_y^{gt}, g_x^{gt}) / M^{gt}$ 
17:   for each VP  $VP_i$  do
18:     Compute VP direction vector
19:      $\vec{v}_i \leftarrow (VP_i^x - u, VP_i^y - v) / \|\vec{v}_i\|$ 
20:     Compute angle with edge direction
21:      $\theta_i^{pred} \leftarrow \arccos(|\vec{d}^{pred} \cdot \vec{v}_i|)$ 
22:      $\theta_i^{gt} \leftarrow \arccos(|\vec{d}^{gt} \cdot \vec{v}_i|)$ 
23:     Compute soft weight using sigmoid
24:      $w_i^{pred} \leftarrow \sigma(\theta_{thresh} - \theta_i^{pred})$ 
25:      $w_i^{gt} \leftarrow \sigma(\theta_{thresh} - \theta_i^{gt})$ 
26:      $\mathcal{S}_i^{pred} \leftarrow \mathcal{S}_i^{pred} + M^{pred} w_i^{pred}$ 
27:      $\mathcal{S}_i^{gt} \leftarrow \mathcal{S}_i^{gt} + M^{gt} w_i^{gt}$ 
28:   end for
29: end for
30: Step 3: Compute final loss
31:  $\mathcal{L}_{vp} \leftarrow \frac{1}{N} \|\mathcal{S}^{gt} - \mathcal{S}^{pred}\|_2^2$ 
32: return  $\mathcal{L}_{vp}$ 

```

tent region, users can specify the correct outlines and define the modification area as the region between the desired and original outlines.

The system streamlines the correction workflow through automated assistance. We employ an off-the-shelf VP detection model [17] to automatically detect and display potential VP candidates, allowing users to easily select or refine the desired VP position based on these suggestions. After VP selection, auxiliary lines are overlaid from the selected VP, serving as visual guides that help users easily

1. Vanishing point position specification with visual guides



2. Correct building outline specification



3. Original building outline specification to calculate masks



Figure 9. Graphical user interface for vanishing point correction. (1) Users select a VP position with the assistance of auto-detected VP suggestions displayed, simplifying the specification process. The system overlays auxiliary lines from the selected VP as visual guides. (2) Users mark the correct building outlines that should align with the VP. (3) Users specify the original misaligned outlines to define the mask regions for correction.

identify where building outlines deviate from proper perspective. For each identified inconsistency, users mark the desired and original outlines, and the system automatically generates a mask from the area between these outlines. This targeted masking approach is crucial to the precision of our method, as it restricts modifications exclusively to geometrically inconsistent regions, preserving the original image’s visual characteristics elsewhere.

C. Evaluation Dataset Creation Details

We generated evaluation images using the following prompts across three text-to-image models:

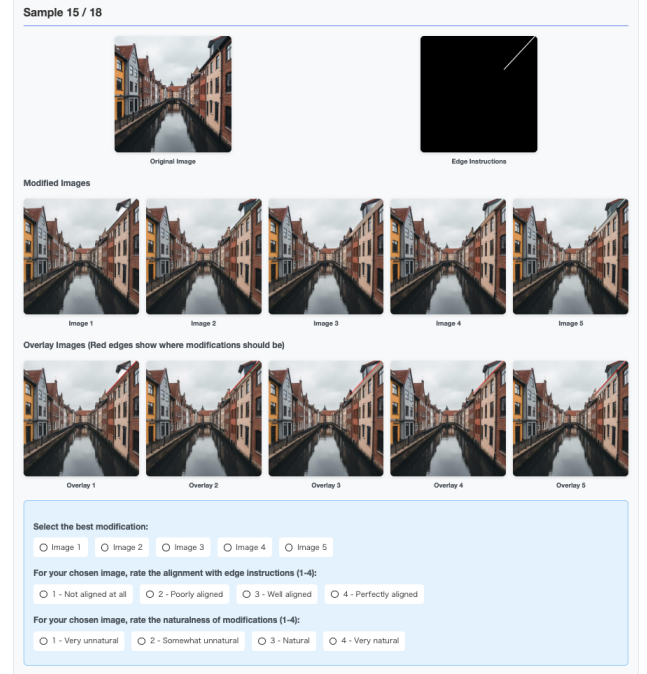


Figure 10. User study interface. Participants evaluate five corrected images for each sample, selecting the one that best follows the edge instructions while maintaining natural appearance. The interface displays the original image, edge instructions showing desired corrections, and five generated variations with overlay visualizations highlighting modification areas.

1. “Buildings on both sides of the road, high quality, photorealistic”
2. “Row of buildings alongside a straight road, high quality, photorealistic”
3. “Row of buildings, high quality, photorealistic”

Each prompt was used to generate multiple images with SD v2.1 [25], FLUX.1-dev [14], and PixArt- α [5], resulting in three distinct evaluation datasets. For each selected image, we manually annotated the target vanishing point, correct building outlines, and modification areas using the GUI described in Section B, and selected images with VP errors. Depth maps for each image were generated using Depth Anything V2 [32] for use with Depth-Persp. This resulted in 250 samples for SD v2.1, and 50 samples each for FLUX.1-dev and PixArt- α .

D. User Study Details

We conducted a user study to evaluate the perceptual quality and geometric fidelity of our correction results. Figure 10 shows our user study interface, which presents participants with the original image alongside five consecutive correction results generated by our method. To help participants understand the intended modifications, we provide edge in-

structions and overlay visualizations showing where corrections should occur. This user study is approved by the IRB of Graduate School of Information Science and Technology, The University of Tokyo (UT-IST-RE-250508_6).

Study Design. In practical image generation workflows, users typically select the best images from multiple generated candidates. Therefore, it is important to evaluate the best rather than the average performance across multiple generations. For each sample, we ask participants to: (1) select the image that best follows the provided edge instructions, and (2) rate both the alignment accuracy (1-4 scale: not aligned to perfectly aligned) and naturalness of modifications (1-4 scale: very unnatural to very natural) for their chosen image.

We randomly sampled 12 images from each of our three evaluation datasets (SD v2.1, FLUX.1-dev, and PixArt- α), selecting only images with AA greater than 3° to ensure meaningful geometric inconsistencies requiring correction. Twenty participants completed the study, providing 720 total evaluations (20 participants \times 12 samples \times 3 models).

E. More Qualitative Results

Figure 11 presents additional qualitative results of ControlVP on representative samples from our three evaluation datasets. The examples include various architectural scenes with different styles and perspectives generated by SD v2.1, FLUX.1-dev, and PixArt- α . For each image pair, ControlVP successfully corrects the vanishing point inconsistencies while preserving the original visual characteristics. The corrected images show proper alignment of building outlines toward the specified vanishing point, demonstrating consistent performance across different source models and scene types.



(a) Correcting generated images from SD v2.1



(b) Correcting generated images from FLUX.1-dev



(c) Correcting generated images from Pixart-α

Figure 11. Additional qualitative results of ControlVP on images generated from three different text-to-image models. (a) Results on SD v2.1 generated images, (b) Results on FLUX.1-dev generated images, and (c) Results on Pixart-α generated images. For each example, we show the original image with VP inconsistencies (top) and the corrected result using ControlVP (bottom). Red lines indicate misaligned building outlines, while green lines show properly aligned outlines after correction. ControlVP successfully corrects geometric inconsistencies across diverse architectural styles and perspectives while maintaining visual quality.