P-Flow: Prompting Visual Effects Generation

Supplementary Material

A. Implementation Details

A.1. Prompt Optimization Procedure

Algorithm 1 summarizes the full test-time optimization loop used in P-Flow. Given a reference video V_{ref}, an initial prompt P_0, and an optional input image I, we first initialize the historical trajectory $\mathcal{H}$ and the iteration index i. We then compute an inversion noise code η_{inv} by calling `FlowMatchingInversion` on $(V_{\text{ref}}, P_0, I, \mathcal{G})$, and obtain a motion-preserving temporal prior η_{temporal} with `ProjectNoiseTemporally`(η_{inv}). These two steps implement the noise prior enhancement described in the method section of the main paper.

Starting from $P_i = P_0$, the algorithm performs an iterative refinement over $i = 0, \ldots, i_{\max} - 1$. At each iteration, we first sample a fresh Gaussian noise $\eta_{\text{new}} \sim \mathcal{N}(0, I)$ and blend it with the temporal prior to form the actual sampling noise

$$\eta \;=\; \sqrt{\alpha}\,\eta_{\text{temporal}} + \sqrt{1-\alpha}\,\eta_{\text{new}},$$

where α controls the trade-off between stability, reusing the temporal prior, and diversity and exploratory, introducing new randomness. Using this blended noise, the video diffusion model $\mathcal{G}$ generates a video $V_i = \mathcal{G}(P_i, I, \eta)$ conditioned on the current prompt and, when applicable, the input image.

To provide the VLM with a direct, visual comparison between the reference effect and the current generations, we construct a combined video V_{comb} by concatenating multiple clips. In the first iteration, V_{comb} contains only V_{ref} and V_0; in subsequent iterations, it contains V_{ref}, the previous generation V_{i-1}, and the current one V_i. This design allows the VLM to assess both the absolute discrepancy with the reference and the incremental change across iterations.

The vision-language model $\mathcal{M}$ then takes $(V_{\text{comb}}, P_i, \mathcal{H})$ as input and returns a diagnostic analysis A_i together with an updated prompt P_{i+1}. Here, $\mathcal{H}$ denotes the historical trajectory that stores past prompts, analyses, and generated videos, as described in the method section of the main paper. We update $\mathcal{H}$ via `UpdateHistory` to include (P_i, A_i, V_i), and proceed to the next iteration. After $i_{\max}$ iterations, the procedure outputs the final optimized prompt, the last generated video, and the complete trajectory $\mathcal{H}$ as summarized in Algorithm 1.

A.2. Structured instruction for VLM

The instruction provided to the VLM are detailed in Listing 1. It directs the VLM to analyze a combined video containing up to three segments (reference, last generated, and newly generated), compare their visual effects and motion dynamics, and refine the prompt to minimize the misalignments while preserving the subject and environment. The instruction operates by iteratively updating the `<current_prompt>` with the refined text prompt based on the VLM analysis, leveraging a memory of past iterations `<memory_to_replace>` to track refinement effectiveness.

The placeholders, such as `<current_prompt>` and `<memory_to_replace>`, are dynamic variables, iteratively updated by the P-Flow. While the placeholders `<subject>` and `<environment>` are fixed and automatically extracted from the initial text prompt, and the `<desired_visual_effect>` is given by user input. The instruction mandates the VLM to output a structured JSON content, containing the analysis and refined prompt, enabling automated parsing and integration into the iterative pipeline.

B. Limitations and Future Works

Despite the promising visual effect customization performance, our current framework still has limitations in terms of optimization efficiency. First, the number of optimization iterations is fixed across all cases, which may lead to suboptimal efficiency. In practice, we observe that some prompts can achieve satisfactory visual effects within a few iterations, while more challenging cases may require extended refinement. However, without an adaptive stopping mechanism, the optimization process may either run longer than needed or stop before achieving optimal results. In future work, we plan to introduce an auxiliary VLM as an evaluator to dynamically assess the alignment between the generated visual effect and the target one, thereby enabling adaptive stopping when sufficient quality is achieved.

Second, the current framework relies on full video generation through multiple flow-matching steps before evaluating the alignment with the desired visual effect. Combined with the iterative prompt optimization loop, this results in a relatively time-consuming process. Empirically, we find that the primary visual effects often emerge in the early part of the generation time steps. This motivates a future direction to perform evaluation and prompt refinement at intermediate generation time steps, potentially reducing time cost and improving overall efficiency.

C. Potential Broader Implications

We present a prompt optimization framework that enables visual effect customization in video generation. By improving the controllability of video outputs through natural language,

Algorithm 1 P-Flow Framework

Require: Reference video V_{ref}, initial prompt P_0, optional input image I, video diffusion model $\mathcal{G}$, VLM $\mathcal{M}$, max iterations $i_{\max}$, blending weight α

Initialize historical trajectory $\mathcal{H} \leftarrow \emptyset$, iteration index $i \leftarrow 0$
Compute inversion noise $\eta_{\text{inv}} \leftarrow \text{FlowMatchingInversion}(V_{\text{ref}}, P_0, I, \mathcal{G})$
Compute temporal noise $\eta_{\text{temporal}} \leftarrow \text{ProjectNoiseTemporally}(\eta_{\text{inv}})$
Set current prompt $P_i \leftarrow P_0$
for $i < i_{\max}$ **do**
    Sample random noise $\eta_{\text{new}} \sim \mathcal{N}(0, I)$
    Blend noise $\eta \leftarrow \sqrt{\alpha} \cdot \eta_{\text{temporal}} + \sqrt{1-\alpha} \cdot \eta_{\text{new}}$
    Generate video $V_i \leftarrow \mathcal{G}(P_i, I, \eta)$
    if $i = 0$ **then**
        Combine videos $V_{\text{comb}} \leftarrow \text{CombineVideos}([V_{\text{ref}}, V_i])$
    else
        Combine videos $V_{\text{comb}} \leftarrow \text{CombineVideos}([V_{\text{ref}}, V_{i-1}, V_i])$
    end if
    Analyze and refine prompt: $(A_i, P_{i+1}) \leftarrow \mathcal{M}(V_{\text{comb}}, P_i, \mathcal{H})$
    Update history: $\mathcal{H} \leftarrow \text{UpdateHistory}(\mathcal{H}, P_i, A_i, V_i)$
    $i \leftarrow i + 1$
end for

return Optimized prompt P_i, generated video V_i, trajectory $\mathcal{H}$

our method lowers the barrier for users to generate videos with desired visual effects. This could benefit creative industries such as animation, marketing, and virtual content creation, while also advancing research in the customization of video generation.

However, as with all generative models, our framework inherits potential risks, including the amplification of societal biases and the possibility of misuse, such as generating misleading or harmful content. To mitigate these risks, we will include explicit terms of use in the user agreement, warning against the generation of violent, obscene, or deceptive content. These terms are intended to discourage unethical usage and clarify user responsibility when interacting with the system.

Besides, our framework builds upon pre-trained video generation models and vision-language models that have integrated safety checkers. These built-in mechanisms help detect and filter out undesirable outputs during generation.

D. More Results

We provided more image-to-video generation results in Fig. 1 and text-to-video generation results in Fig. 2. The video version of the results presented in the appendix and main paper can be found in the zip file attached within the supplementary material.

In Fig. 1, we present image-to-video generation results on two challenging visual effects: *Crumble* and *Cake-ify*. Compared with Wan 2.1 and HunyuanVideo, both of which tend to either preserve the input appearance with minimal effect expression or generate inappropriate transformations, P-Flow achieves substantially more faithful and temporally consistent effect reproduction. Leveraging the refined prompts generated during optimization, our method is able to perceive visual cues from the input image while inducing effect behaviors that closely match the reference dynamics, for instance, controlled disintegration patterns in *Crumble* or revealing the internal structure of an object in *Cake-ify*. These results demonstrate that P-Flow maintains strong visual coherence with the source image while enabling expressive and high-fidelity dynamic visual effect customization.

In Fig. 2, we compare P-Flow with Wan 2.1 and HunyuanVideo on two dynamic visual effects: *Levitate* and *Inflate*. For each effect, we show the reference video, baseline generations, and the result generated by our method accompanied by the refined prompt. As illustrated, baseline models often generate weakly expressed motions that loosely resemble the intended visual dynamics, whereas P-Flow successfully induces high-fidelity, temporally coherent visual effects that more closely match the reference progression. The refined prompts generated by our method capture richer temporal and effect-related semantics, which enable the video generation model to reproduce more high-fidelity and expressive visual effect behaviors in novel scenes.

```
Instruction = """
Your task is to optimize a text prompt for the video generation model to match the reference video's dynamic visual effect "<
    desired_visual_effect>".

Input: Combined video with up to three segments:
- "A" (top): Reference video.
- "B" (middle, if present): Last generated video. Corresponding text prompt: "<last_text_prompt>".
- "C" (bottom): New generated video. Corresponding text prompt: "<current_text_prompt>".

Steps:
1. **Analyze**:
   - "A": Describe visual effects (focusing on "<desired_visual_effect>" related dynamics), followed by related motion dynamics (speed,
       direction, pattern) and transitions (timing, rhythm).
   - "B" (if present): Summarize visual effects, motion dynamics, and transitions.
   - "C": Summarize visual effects, motion dynamics, and transitions.

2. **Compare**:
   - Compare "C" (and "B", if present) to "A" for differences in visual effects, motion dynamics, and transitions.
   - For "B", identify prompt terms causing misalignments in visual effects or motion dynamics.
   - Evaluate how the prompt changes from "B" to "C" affects the visual effects alignment with "A".

3. **Refine Prompt**:
   - Keep "<subject>" and "<environment>" unchanged.
   - Refine the text prompt "<current_prompt>" to match "A"'s visual effects " <desired_visual_effect>", and related motion dynamics
       and transitions better, and fix its errors.
   - Avoid instructional language and problematic terms.

4. **Output**:
   - JSON:
     - "analysis":
       - "reference_description": "A"'s visual effects, motion dynamics, and transitions.
       - "last_generated_description" (if "B" exists): "B"'s visual effects, motion dynamics, and transitions.
       - "new_generated_description": "C"'s visual effects, motion dynamics, and transitions.
       - "comparison": Summary of differences of "C" and "A" in visual effects, motion dynamics, and transitions, including errors in "
           B"'s prompt and their impact..
     - "refined_prompt": Optimized prompt for "C" to minimize the misalignment with "A"'s visual effects.


Guidelines:
- Use "<memory_to_replace>" to track the history of prompt refinements and their effectiveness.
- Prioritize "<desired_visual_effect>" and visual effects, then motion dynamics and transitions.
- Do not include non-visual effect details from "A" (e.g., specific colors or other appearance-related elements unless part of "<
    desired_visual_effect>").

Previous history: <memory_to_replace>
Subject: <subject>
Environment: <environment>
Desired Visual Effect: <desired_visual_effect>
Current prompt: <current_prompt>
"""
```

Listing 1. Instructions for VLM

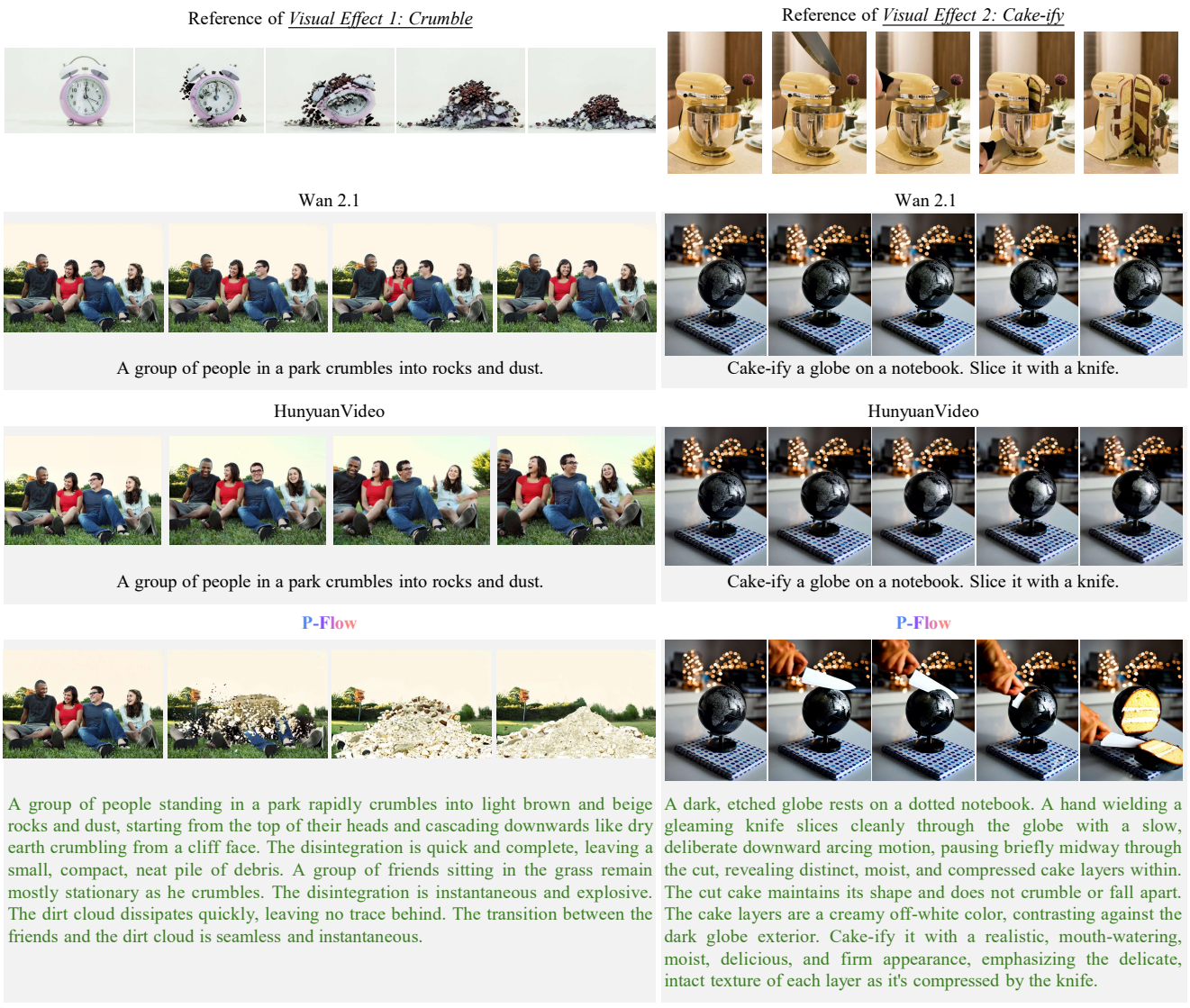
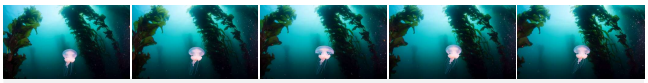


Figure 1. Image-to-Video Generation Results.

Reference of *Visual Effect 3: Levitate*

Wan 2.1



A single jellyfish levitates through a kelp forest.

HunyuanVideo

A single jellyfish levitates through a kelp forest.

P-Flow

A single jellyfish levitates through a kelp forest. It drifts upward with the barest hint of a wobble, as if suspended in a weightless dream. Its tentacles trail languidly below, pulsing with a slow, hypnotic rhythm that suggests the breathing of a sleeping giant. Focus on the jellyfish, maintaining a completely fixed camera angle. The jellyfish should rise with an almost imperceptible drift, as if time itself were slowed. Emphasize the ethereal nature of the wobbling and pulsing movements, creating a serene and otherworldly ascent.

Reference of *Visual Effect 4: Inflate*

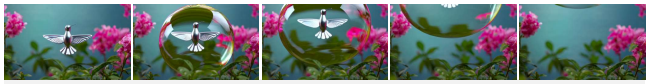
Wan 2.1

The chrome hummingbird inflates in a jungle.

HunyuanVideo

The chrome hummingbird inflates in a jungle.

P-Flow



The chrome hummingbird inflates as if it's enveloped within an infinitely thin, perfectly transparent, and rapidly expanding spherical force field within a softly blurred jungle, adorned with diffuse pink flowers. This invisible spherical field should expand extremely smoothly and uniformly, seamlessly lifting the hummingbird from its perch as the sphere grows. The chrome hummingbird, retaining its intricate form and gleaming metallic texture, should appear to effortlessly float upwards within this imperceptible expanding force field. As the hummingbird ascends, the vibrant hues of the jungle and the delicate pink flowers should gradually melt into the background.

Figure 2. Text-to-Video Generation Results.