

A PRELIMINARY

Latent Diffusion Model (LDM): LDM is an efficient variant of diffusion models that operates in latent space rather than directly in image space. It consists of two primary components: an encoder ϵ , which compresses an image x into latent code $z = \epsilon(x)$, and a decoder that reconstructs the image as $x \approx \mathcal{D}(z)$.

The model learns the distribution of image latent codes $z_0 \sim p_{data}(z_0)$ through a Denoising Diffusion Probabilistic Model (DDPM) framework, involving both forward and backward processes. The forward diffusion process adds Gaussian noise at each timestep t , resulting in z_t :

$$q(z_t|z_{t-1}) = \mathcal{N}(z_t; \sqrt{1 - \beta_t}z_{t-1}, \beta_t I), \quad (1)$$

where $\{\beta_t\}_{t=1}^T$ are noise scales, and T is the total number of timesteps. The backward process predicts the less noisy latent code z_{t-1} :

$$p_\theta(z_{t-1}|z_t) = \mathcal{N}(z_{t-1}; \mu_\theta(z_t, t), \Sigma_\theta(z_t, t)). \quad (2)$$

Here, μ_θ and Σ_θ are derived from a denoising model ϵ_θ with learnable parameters, optimized via:

$$\mathcal{L}_{simple} := \mathbb{E}_{(z), \epsilon \sim \mathcal{N}(0,1), t} [\|\epsilon - \epsilon_\theta(z, t)\|_2^2]. \quad (3)$$

To generate new samples, we initialize with $z_T \sim \mathcal{N}(0, 1)$ and utilize DDIM sampling to estimate z_{t-1} :

$$z_{t-1} = \sqrt{\alpha_{t-1}} \left(\frac{z_t - \sqrt{1 - \alpha_t} \epsilon_\theta(z_t, t)}{\sqrt{\alpha_t}} \right) + \sqrt{1 - \alpha_{t-1}} \cdot \epsilon_\theta(z_t, t), \quad (4)$$

where $\alpha_t = \prod_{i=1}^t (1 - \beta_i)$. For simplicity, we denote $z_{t \rightarrow 0}$ as the predicted z_0 at timestep t . The base model employed is Stable Diffusion (SD) $\epsilon_\theta(z_t, t, \tau)$, which is a text-guided LDM trained on extensive image-text pairs, where τ represents the text prompt.

Continuous-Time Diffusion Model Framework: This section provides a summary of diffusion models (DMs) using a continuous-time framework. Let $p_{data}(x_0)$ represent the data distribution, and $p(x; \sigma)$ denote the distribution derived from adding i.i.d. Gaussian noise with variance σ^2 . For sufficiently large σ_{max} , we have $p(x; \sigma_{max}^2) \approx \mathcal{N}(0, \sigma_{max}^2)$.

DMs utilize this property, starting from high variance Gaussian noise $x_M \sim \mathcal{N}(0, \sigma_{max}^2)$ and iteratively denoising towards $\sigma_0 = 0$. This process can be described by the *Probability Flow* ordinary differential equation (ODE):

$$dx = -\dot{\sigma}(t)\sigma(t)\nabla_x \log p(x; \sigma(t)) dt, \quad (5)$$

where $\nabla_x \log p(x; \sigma)$ is the *score function*. The training of the DM focuses on learning a model $s_\theta(x; \sigma)$ that approximates this score function. It can be parameterized as:

$$\nabla_x \log p(x; \sigma) \approx s_\theta(x; \sigma) = \frac{D_\theta(x; \sigma) - x}{\sigma^2}, \quad (6)$$

where D_θ is a learnable *denoiser* aimed at predicting the clean data x_0 .

The denoiser is trained using *denoising score matching* (DSM):

$$\mathbb{E}_{(x_0, c) \sim p_{data}(x_0, c), (\sigma, n) \sim p(\sigma, n)} [\lambda_\sigma \|D_\theta(x_0 + n; \sigma, c) - x_0\|_2^2], \quad (7)$$

where $p(\sigma, \mathbf{n}) = p(\sigma) \mathcal{N}(\mathbf{n}; \mathbf{0}, \sigma^2)$. The function $p(\sigma)$ can be a discrete set or a continuous range of noise levels.

The weighting function $\lambda_\sigma: \mathbb{R}_+ \rightarrow \mathbb{R}_+$ adjusts the importance of the noise level, and \mathbf{c} represents an arbitrary conditioning signal. In this work, we adopt the EDM preconditioning framework, defining the denoiser as follows:

$$D_\theta(\mathbf{x}; \sigma) = C_{\text{skip}}(\sigma)\mathbf{x} + C_{\text{out}}(\sigma)F_\theta(C_{\text{in}}(\sigma)\mathbf{x}; C_{\text{noise}}(\sigma)), \quad (8)$$

where F_θ is the network to be trained.

Base Model Architecture: This study utilizes the publicly available Stable Diffusion 2.1 (SD 2.1) model as the foundational architecture. In the context of the EDM (Enhanced Denoising Model) framework, SD 2.1 employs several preconditioning functions, which play a critical role in the model’s ability to manage noise levels during the diffusion process. The specific preconditioning functions for SD 2.1 are defined as follows:

$$C_{\text{skip}}^{\text{SD2.1}}(\sigma) = 1, \quad (9)$$

$$C_{\text{out}}^{\text{SD2.1}}(\sigma) = -\sigma, \quad (10)$$

$$C_{\text{in}}^{\text{SD2.1}}(\sigma) = \frac{1}{\sqrt{\sigma^2 + 1}}, \quad (11)$$

$$C_{\text{noise}}^{\text{SD2.1}}(\sigma) = \arg \min_{j \in [1000]} (\sigma - \sigma_j), \quad (12)$$

$$(13)$$

where $\sigma_{j+1} > \sigma_j$ denotes an ordered sequence of noise levels.

The training of the original SD 2.1 model employs a uniform distribution over 1000 discrete noise levels, represented as $\{\sigma_j\}_{j \in [1000]}$. A significant challenge arises from the fact that even at the maximum discrete noise level σ_{1000} , the signal-to-noise ratio remains relatively high. This high ratio can lead to difficulties, particularly when generating images that require very low brightness or exhibit dark tones.

To mitigate this issue, a previous approach introduced the concept of ”offset noise,” which modifies the training objective to create a non-isotropic Gaussian distribution for $p(\mathbf{n} | \sigma)$. However, in this work, we propose a different strategy: rather than just adjusting the noise distribution, we modify the preconditioning functions and the distribution of training noise levels entirely.

Model Fine-tuning: In this section, we describe our approach to fine-tuning the image model by replacing the original preconditioning functions with new formulations that better address the identified issues. The modified preconditioning functions are defined as follows:

$$C_{\text{skip}}(\sigma) = (\sigma^2 + 1)^{-1}, \quad (14)$$

$$C_{\text{out}}(\sigma) = \frac{-\sigma}{\sqrt{\sigma^2 + 1}}, \quad (15)$$

$$C_{\text{in}}(\sigma) = \frac{1}{\sqrt{\sigma^2 + 1}}, \quad (16)$$

$$C_{\text{noise}}(\sigma) = 0.25 \log \sigma. \quad (17)$$

These adjustments aim to improve the model’s performance across various noise levels. Specifically, the function $C_{\text{skip}}(\sigma)$ helps to normalize the contribution of noise, ensuring that the model effectively learns to denoise images across a broader spectrum of input conditions. The function $C_{\text{out}}(\sigma)$ is designed to scale the noise in a way that reflects the inherent characteristics of the data, while $C_{\text{in}}(\sigma)$ maintains the model’s responsiveness to varying noise levels.

Furthermore, the function $C_{\text{noise}}(\sigma)$ introduces a logarithmic adjustment, which aids in managing the distribution of noise levels during training. This comprehensive modification of the preconditioning functions is expected to yield a more robust model capable of generating high-quality images, particularly in scenarios where low signal-to-noise ratios are prevalent.

These new preconditioning functions can be seamlessly integrated into the EDM framework, particularly by setting $\sigma_{\text{data}} = 1$. This integration allows the modified functions to work effectively within the existing architecture, enhancing the model’s overall performance.

ControlNet: ControlNet significantly enhances the capabilities of Stable Diffusion (SD) by allowing for more controllable input conditions during the text-to-image synthesis process. This flexibility enables the incorporation of various types of input data, such as depth maps, poses, edges, and other auxiliary information.

The architecture of ControlNet is built upon the same U-Net framework utilized by the SD model. However, it introduces a crucial modification: the weights of the U-Net are fine-tuned specifically to accommodate these task-specific conditions. This is reflected in the transformation of the denoising function from $\epsilon_{\theta}(\mathbf{z}_t, t, \tau)$ in the original SD model to $\epsilon_{\theta}(\mathbf{z}_t, t, \mathbf{c}, \tau)$ in ControlNet, where \mathbf{c} represents the additional conditions provided as input.

To clarify the distinction between the U-Net components within these two architectures, we refer to the U-Net used in the SD model as the *main U-Net*, while the U-Net tailored for ControlNet is labeled as the *auxiliary U-Net*. This nomenclature helps in understanding the specific roles and functionalities of each U-Net within their respective frameworks, emphasizing the enhanced control and specificity that ControlNet brings to the image generation process.

B MORE RESULTS

Additional results from our video generation experiments are presented below. These examples further demonstrate the capabilities of our model in creating dynamic and consistent video sequences across various settings and scenarios. Detailed analyses of these results help in understanding the strengths and potential areas for improvement in our approach.



Figure 1: Examples of the generated videos.



Figure 2: Examples of the generated videos.

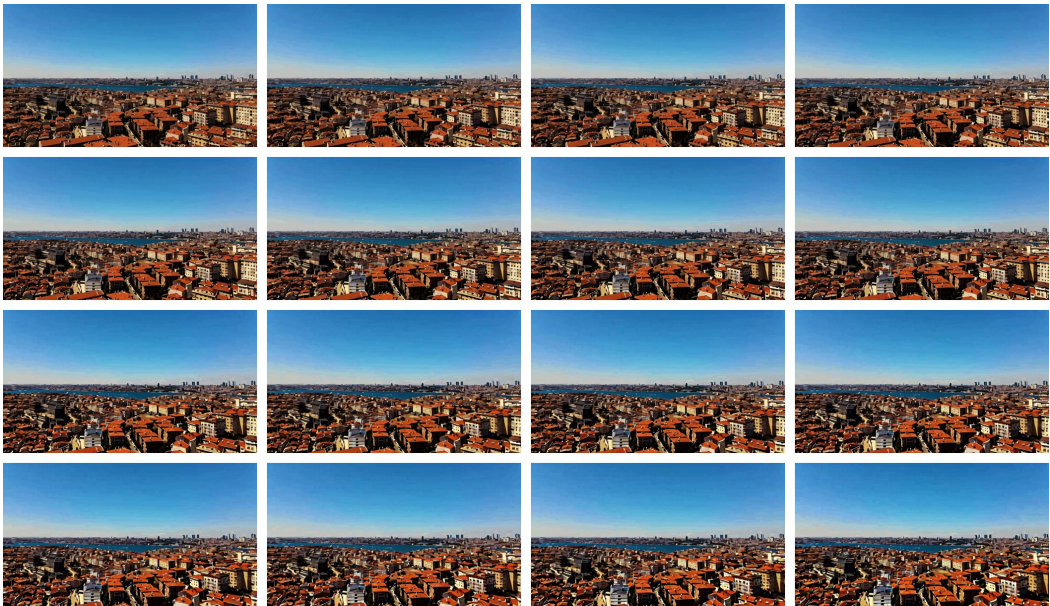


Figure 3: Examples of the generated videos.

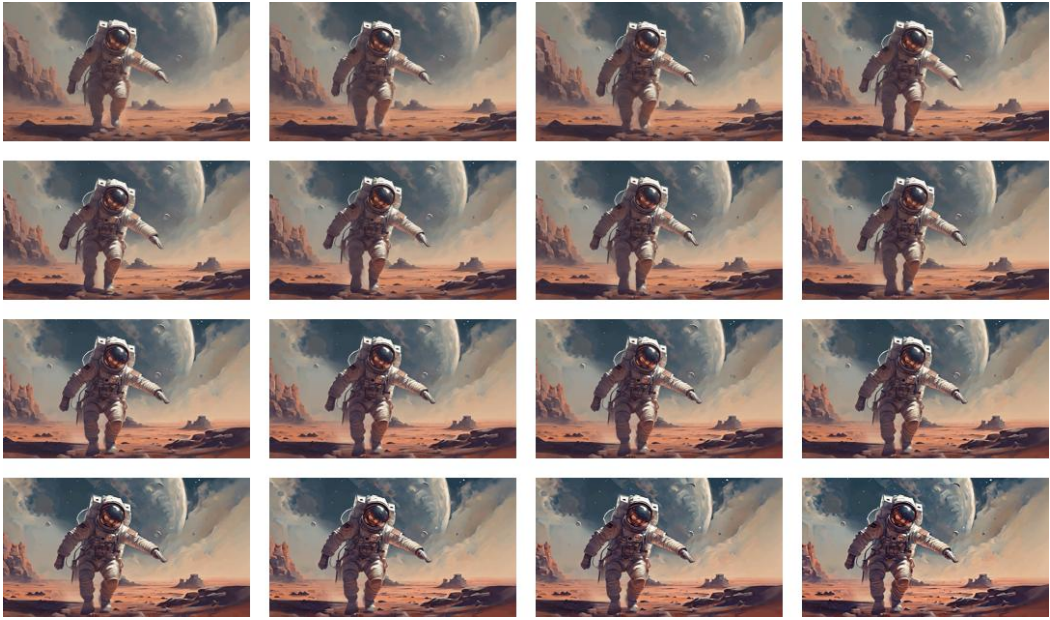


Figure 4: Examples of the generated videos.

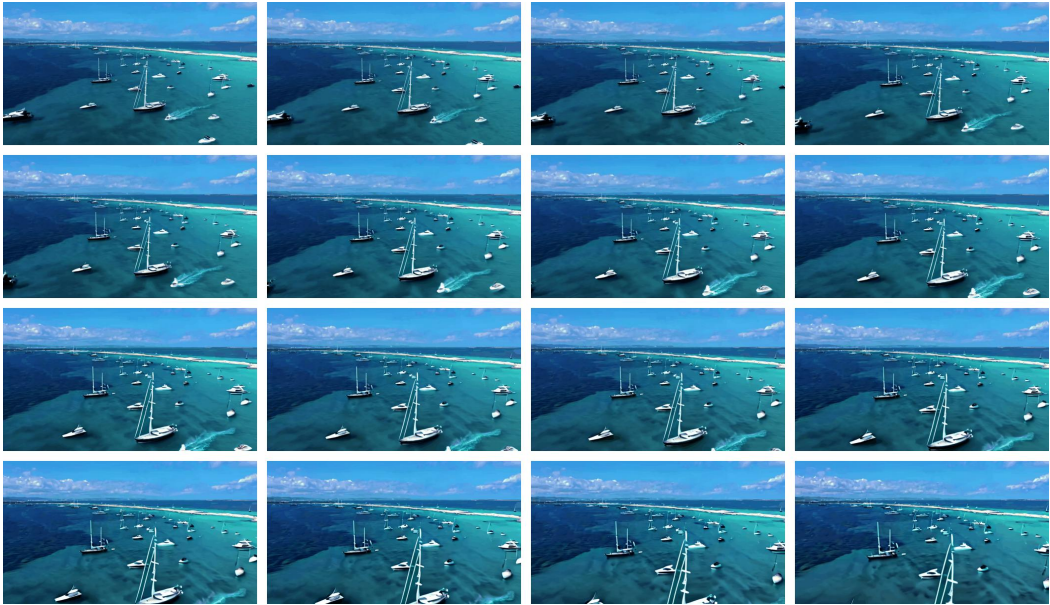


Figure 5: Examples of the generated videos.

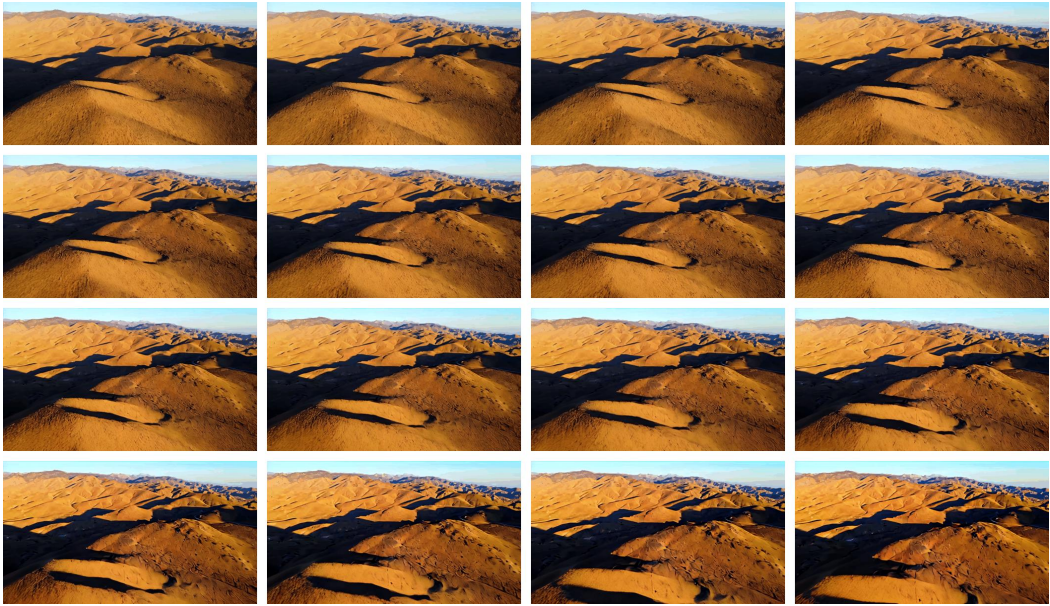


Figure 6: Examples of the generated videos.



Figure 7: Examples of the generated videos.

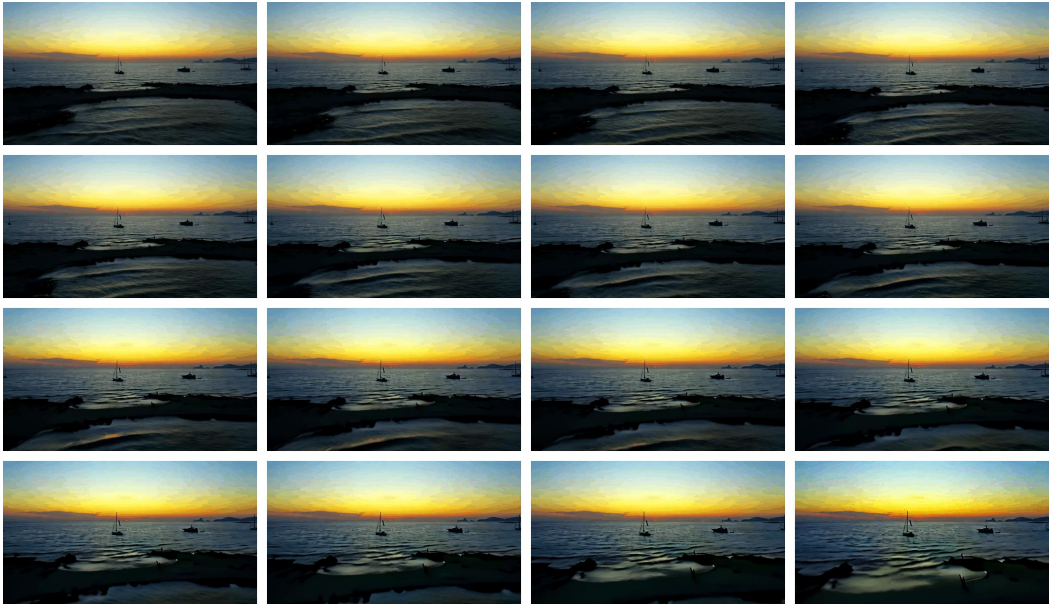


Figure 8: Examples of the generated videos.



Figure 9: Examples of the generated videos.

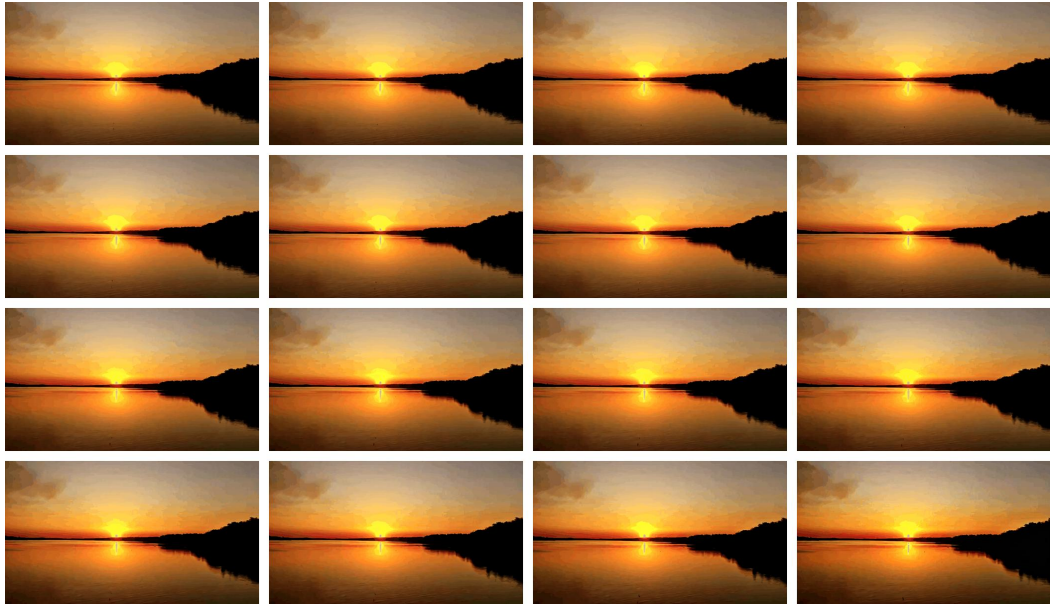


Figure 10: Examples of the generated videos.

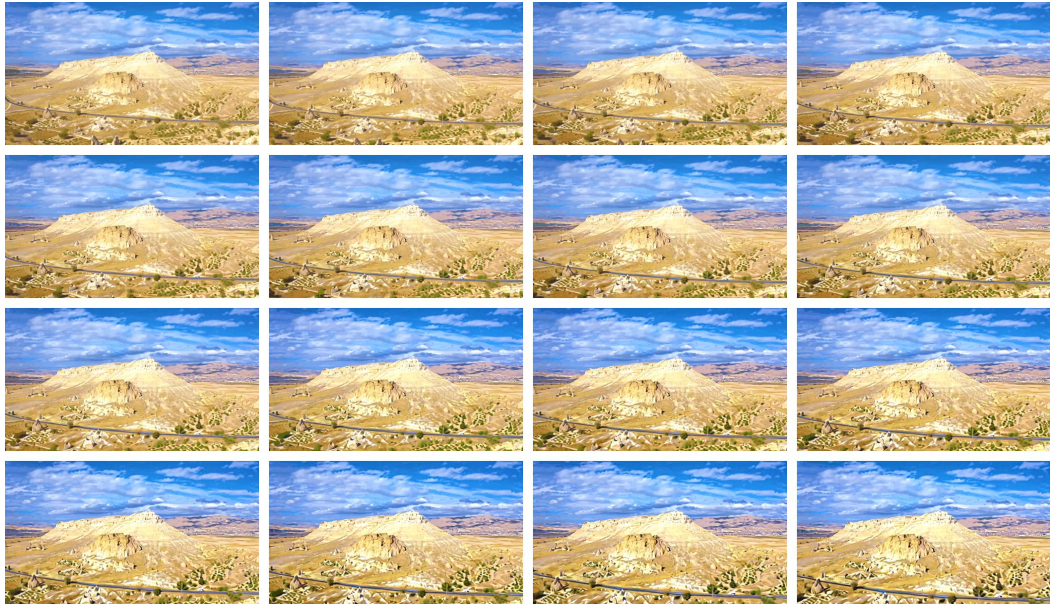


Figure 11: Examples of the generated videos.

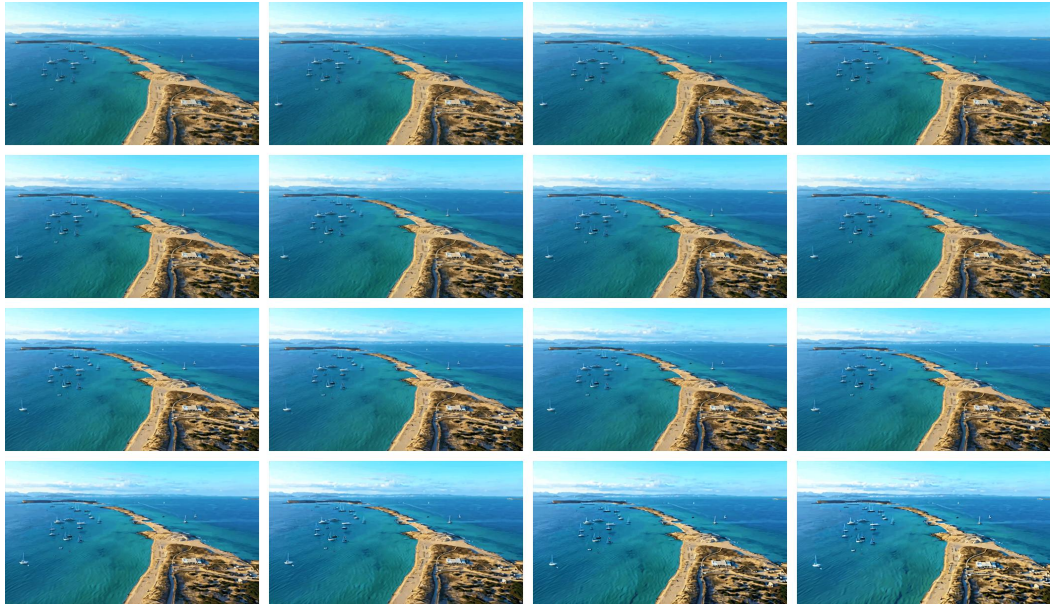


Figure 12: Examples of the generated videos.



Figure 13: Examples of the generated videos.



Figure 14: Examples of the generated videos.

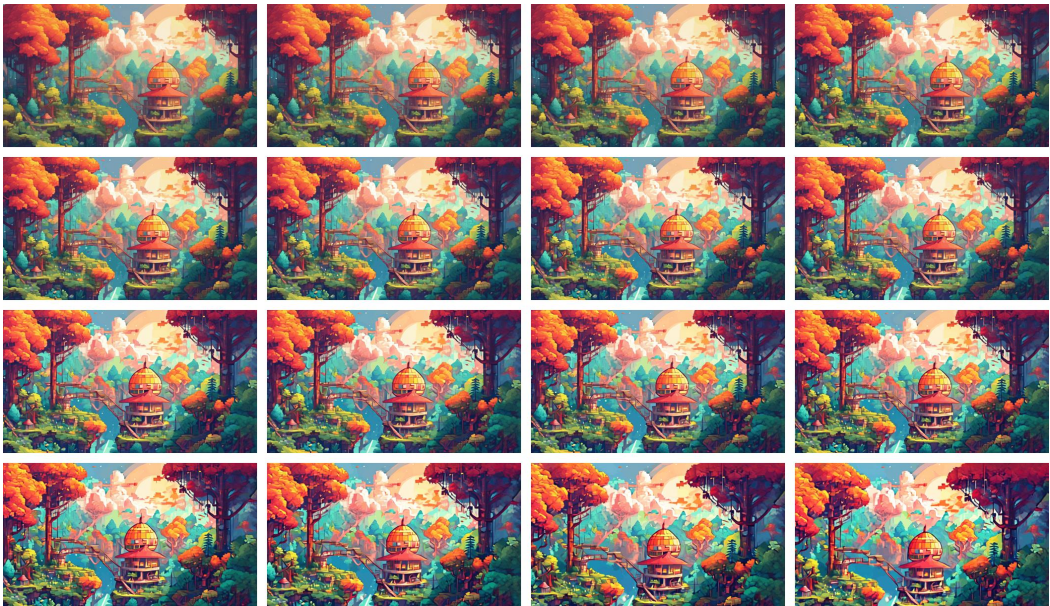


Figure 15: Examples of the generated videos.



Figure 16: Examples of the generated videos.