# LookingGlass: Generative Anamorphoses via Laplacian Pyramid Warping

## Supplementary Material

## A. Laplacian Pyramid Warping

In this section, we provide additional details about Sec. 4.2. In Section A.1 and A.2, we illustrate the forward and inverse warping with the example of the conic mirror, and cover some implementation subtleties for the inverse operation. Section A.3 explains how we properly blend pyramid levels in the case of partial views. Lastly, we provide pseudo-code for Laplacian Pyramid Warping in Section A.4.

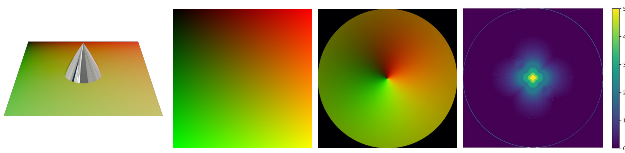



Figure 10. **Conic mirror view.** Rendering the scene (a) with a top down camera yields the UV map (c) for the conic mirror case. Using Eq. (9), the associated level-of-detail map (d) is computed.

### A.1. Forward Warping

We consider the conic mirror example and its corresponding warping function (Fig. 10). The forward warping operation was explained in Sec. 4.2. Figure 11.b) illustrates the result of applying forward warping to an image. In Figure 14.b), we visualize the pyramid levels in the identity view, highlighting in white the pixels used in the forward warping. For instance, pixels closer to the cone's center are mapped to an outer ring in the identity view. Because the view function is locally more compressed for these pixels, they are sampled from a higher level of the pyramid (*i.e.*, lower resolution).

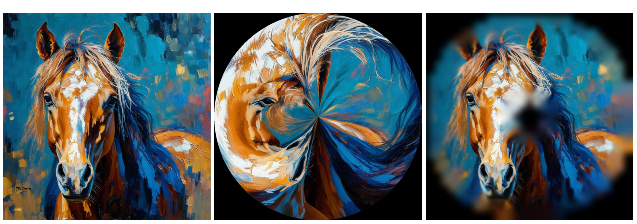



Figure 11. **Warp example.** Given an image (a), we warp it to the conic mirror view (b), and warp back to the original view (c). Our multi-scale approach warps values to different frequency bands based on the local compression of the view function $\pi$.

### A.2. Inverse Warping

Figure 11.c) shows the result of warping the transformed image back to the identity view: pixels closer to the center of the cone in the warped view naturally map back to a lower frequency band in the identity view, occupying larger regions at the boundary in the reconstructed image. Rigorously speaking, the output of the inverse warping is a Laplacian pyramid (see Fig. 14.g), not an image.

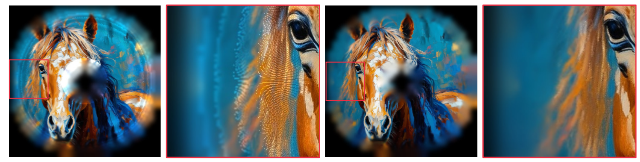



Figure 12. **Imputation.** We use nearest neighbor imputation for pixels in $\mathbb{P}(\mathbf{x})$ that has no colors (*i.e.* did not receive gradients). We consider images with values in [-1, 1].

**Implementation details.** We provide a more detailed illustration of the inverse operation in Figure 16. The subfigures show:

a) Starting from a dummy pyramid $\mathbb{P}(\mathbf{x}^0)$, we perform a Laplacian-to-Gaussian pyramid conversion to get $\mathbb{G}(\mathbf{x}^0)$ (*i.e. reparameterization*). Forward warping is then applied to obtain a warped dummy image $\tilde{\mathbf{y}}$.
b) An $L_2$ loss is computed between $\tilde{\mathbf{y}}$ and $\mathbf{y}$, the image we wish to warp back. The gradients populate each level of the dummy pyramid, yielding $\mathbb{P}(\mathbf{x})$. The reparameterization ensures that gradients from each level flows to all levels above it.
c) Lastly, we extract the final Laplacian pyramid $\mathbb{L}(\mathbf{x})$ from $\mathbb{P}(\mathbf{x})$ following

$$\mathbb{L}(\mathbf{x}) = \mathbb{M}(\mathbf{x}) \odot (\mathbb{P}^*(\mathbf{x}) - \mathbf{U}(\mathbf{D}(\kappa(\mathbb{P}^*(\mathbf{x}))))), \quad (11)$$

where $\mathbb{M}$ is a pyramid of binary masks indicating pixels that have gradients, $\mathbb{P}^*$ is the result of imputing missing values in $\mathbb{P}$ with nearest color (more details below).

**Imputation.** It is important to impute the pixels that received no gradients with meaningful colors. Otherwise, if left black, the pyramid computation will pick up these discontinuities at the mask boundaries as high-frequency details, leading to incorrect values in those regions. We opt for a simple nearest neighbor imputation that fills pixels that have no gradients with the nearest pixel value. Figure 12 illustrates the difference between no imputation (default value 0) and our nearest imputation.

**Comparison with baselines.** In Figure 13, we compare our Laplacian-based inverse warping with standard warping using nearest or bilinear interpolation. Because the view function has extreme compression around the cone center, this translates to missing values on the outer ring when warping back with nearest or bilinear, as some pixels in the identity view are not sampled during forward warping due to the compression. We showed in Fig. 6 that this can lead to artifacts in the generated image.

It is worth noting that another way to avoid missing values could be to do a *forward* warping with the *inverse* UV map. However, the inverse map is usually not trivial to compute. Additional problems arise when the mapping is not bijective or has discontinuities, which can be quite tricky to solve in a robust way. In contrast, our method automatically handles these cases, and only requires the forward UV map, which is easily obtainable through simple rendering of the desired view.

(a) nearest (b) bilinear (c) Laplacian (ours)

Figure 13. **Backward warping baseline comparison.** In regions where the view function is compressing, standard interpolation like nearest (a) or bilinear interpolation (b) creates holes, while our Laplacian Pyramid Warping ensures smooth results (c).

### A.3. Pyramid Blending

As explained in Sec. 4.2, special care needs to be taken when blending the pyramids.

**Blending with partial views.** In standard Laplacian Pyramid Blending, the blended pyramid is obtained by averaging each level of the input pyramids. Given two pyramids $\mathbb{L}^0, \mathbb{L}^1$, the level $k$ of the blended pyramid $\mathbb{L}$ is given by:

$$\mathbb{L}_k = \frac{1}{2}\left(\mathbb{L}_k^0 + \mathbb{L}_k^1\right). \tag{12}$$

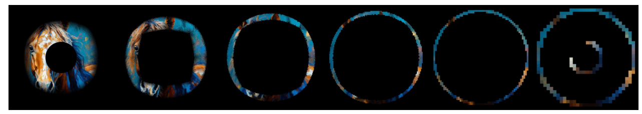
In our case, Laplacian pyramids come from inverse warping of views, and might look like Fig. 11.c), with missing values. In this case, the average should only be computed over defined pixels. Assuming each level $\mathbb{L}_k$ is associated with a binary mask $\mathbb{M}_k$, the blending is:

$$\mathbb{L}_k = \frac{1}{\mathbb{M}_k^0 + \mathbb{M}_k^1}\left(\mathbb{M}_k^0 \odot \mathbb{L}_k^0 + \mathbb{M}_k^1 \odot \mathbb{L}_k^1\right). \tag{13}$$

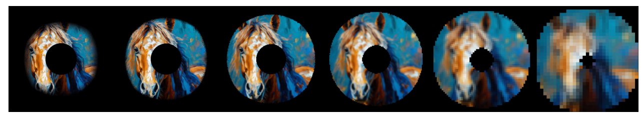
In practice, we map missing values to `torch.nan`, and use `torch.nanmean()` to perform averaging.

(a) input image as Gaussian Pyramid $\mathbb{G}(\mathbf{x})$

(b) forward warping: pixels fetched from $\mathbb{G}(\mathbf{x})$ during warping (highlighted)

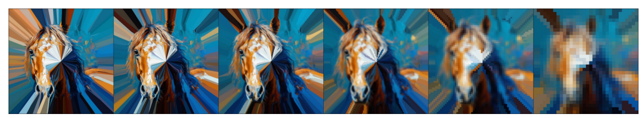


(c) backward warping: intermediate pyramid obtained with backpropagation $\mathbb{G}(\mathbf{x})$

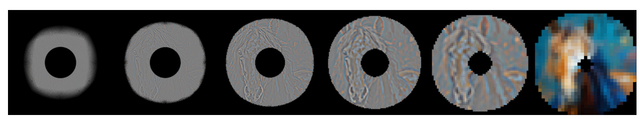


(d) backward warping: propagated pyramid $\mathbb{P}(\mathbf{x})$

(e) backward warping: computed binary mask $\mathbb{M}(\mathbf{x})$



(f) backward warping: pyramid with missing values imputed using nearest value $\mathbb{P}^*(\mathbf{x})$



(g) backward warping: final Laplacian pyramid returned by backward warping $\mathbb{L}(\mathbf{x})$

Figure 14. **Intermediate pyramids.** We visualize some intermediate pyramids that appear in forward (a, b) and backward warping (c-g). Refer to the text and Fig. 16 for more context.

**Detail-preserving averaging.** Another issue with averaging in general is that it reduces variance and washes out details. While this is already improved with the Laplacian pyramid, averaging still leads to the loss of sharp details. Given two pixel values $x, y \in [-1, 1]$, we define a normal averaging `avg` and a value-weighted averaging `vavg`:

$$\texttt{avg}(x,y) = \frac{1}{2}(x+y), \ \texttt{vavg}(x,y) = \frac{|x|x + |y|y}{|x| + |y|} \tag{14}$$

The value-weighted average will give more weights to extreme pixel values, hence preserving better the value range. In the results shown, we linearly interpolate between the two types of averaging through a parameter $\alpha \in [0, 1]$:

$$z = \texttt{avg}(x,y) + \alpha(\texttt{vavg}(x,y) - \texttt{avg}(x,y)). \tag{15}$$

Figure 15 shows the effect of $\alpha$ for two examples with the cylinder mirror. When standard averaging is used ($\alpha =$

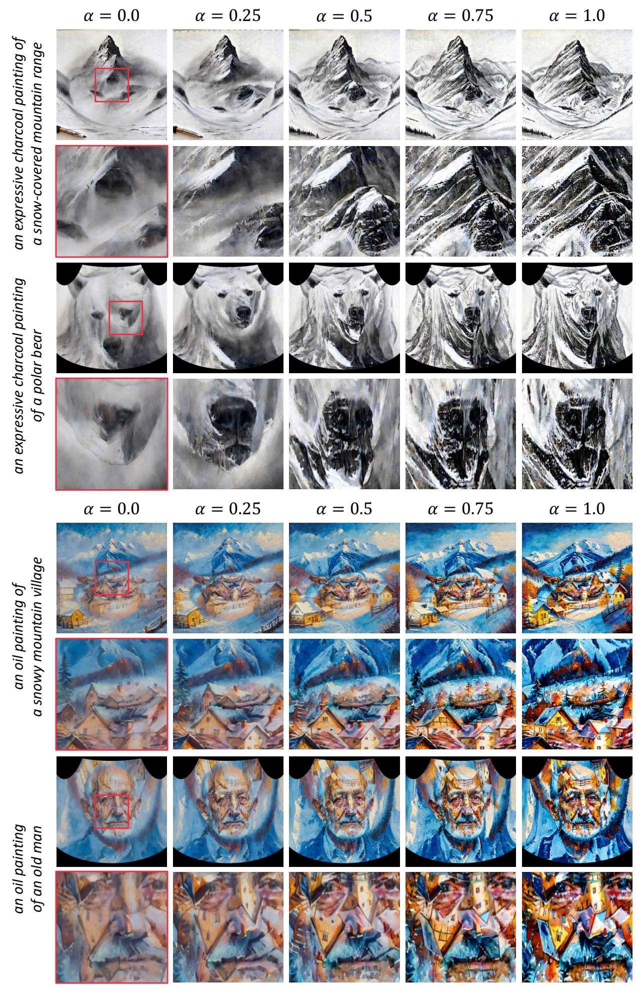



Figure 15. **Effects of parameter $\alpha$.** The parameter affects how views are merged together at each level of the pyramid. $\alpha = 0$ corresponds to standard average, while $\alpha = 1$ gives more weights to pixels with extreme values, preserving the details in all the views.

0), the image looks blurrier. However, when value-weighted average is used ($\alpha = 1$), all details from both views are preserved, creating very saturated, over-sharpened results. In most of our results, we opt for $\alpha \in [0.25, 0.5]$.

## A.4. Pseudo-code

We provide a pseudo-code for our novel Laplacian Pyramid warping. Some notes:

- l.8: `_compute_lod_level` returns the LOD level map as a $(H, W)$ tensor using Equation (9);
- l.29: `pyrStack` takes a pyramid, upsamples all levels to highest resolution, and stacks them along a dimension;
- l.73: `impute_with_nearest` fills missing values with nearest ones in the image given the mask.

```
1
2  def _get_grid(warp, maxLOD):
3      """
4      Takes in numpy array warp of shape (1, 3, 1024, 1024)
5      """
6      # compute lod and normalize to (-1, 1)
7      mapping = 1024 * warp[:, :2]
8      lod = _compute_lod_level(mapping, maxLOD=maxLOD)
9      lod = 2 * lod / maxLOD - 1.0  # (h, w)
10     lod = lod.unsqueeze(0).unsqueeze(-1)
11
12     # normalize uv to (-1, 1)
13     grid = torch.tensor(warp[:, :2]).permute(0, 2, 3, 1)
14     mask = torch.all(grid == 0.0, dim=-1, keepdim=True)
15     mask = mask.expand(-1, -1, -1, 2)
16     grid[mask] = torch.nan  # replace undefined with NaN
17     grid = 2 * grid - 1
18
19     # combine into a 3D coordinate grid (lod, u, v)
20     grid = torch.cat([lod, grid], -1).unsqueeze(1)
21     return grid  # (1, 1, dim, dim, 3)
22
23
24 def view(lp, warp, leveln):
25     # get 3d coordinate grid
26     grid = _get_grid(warp, maxLOD=leveln-1)
27
28     # sample values
29     layers = pyrStack(lp, dim=-1)
30     new_im = F.grid_sample(
31         layers,
32         grid,
33         mode='nearest',
34         padding_mode="zeros",
35         align_corners=True,
36     ).squeeze(2)
37
38     # replace by NaN where image is 0
39     new_im[new_im == 0.0] = torch.nan
40     new_im = torch.nanmean(new_im, 0).half()
41     return new_im
42
43
44 def inverse_view(im, warp, leveln):
45     c, h, w = im.shape
46     grid = _get_grid(warp, maxLOD=leveln - 1)
47     with torch.enable_grad():
48         # create an empty pyramid
49         opt_var = torch.zeros(1, c + 1, h, w)
50         opt_var = LaplacianPyramid(opt_var, leveln)
51         for lvl in opt_var:
52             lvl.requires_grad_()
53
54         # convert to Gaussian pyramid
55         opt_gp = Laplacian2Gaussian(opt_var)
56         target = torch.cat([im, torch.ones_like(im[:1])])
57         layers = pyrStack(opt_gp, -1).float()
58         warped = F.grid_sample(
59             layers,
60             grid,
61             mode='nearest',
62             padding_mode="zeros",
63             align_corners=True,
64         ).squeeze(2)
65         loss = 0.5 * ((warped - target) ** 2).sum()
66         loss.backward()
67         result = [-l.grad.detach() for l in opt_var]
68         result = [(r[:, :c] / r[:, -1:]) for r in result]
69
70     # extract laplacian pyramid
71     for k in range(leveln - 1):
72         mask = torch.isnan(result[k])
73         imputed = impute_with_nearest(result[k], ~mask)
74         result[k] = imputed - pyrUp(pyrDown(imputed))
75         result[k][mask] = torch.nan
76
77     return result
78
```

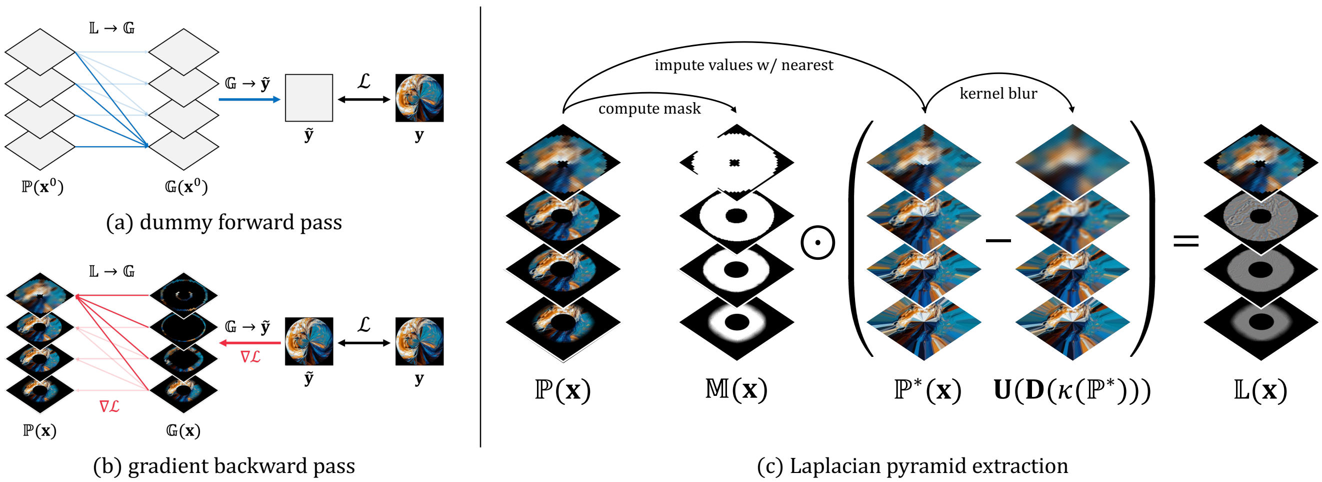



Figure 16. **Detailed look at inverse warping.** We provide an illustration to Equation (10), explaining how to obtain the final Laplacian pyramid using back-propagation, as mentioned in the main paper.

# B. Additional Ablations

We provide more qualitative ablations to show the effects of Laplacian warping, view prioritization, and time travel.

## B.1. Prioritizing a Single View

In Section 4.3, we proposed to let the last $x\%$ of the denoising process only denoise a single one of the views. Figure 17 shows the effect of varying values of $x$ for an example with 90° rotation. As the ratio increases, details from the first view (*"a snowy mountain village"*) dominate over the second view (*"a horse"*). This is a useful parameter to control the trade-off between views.

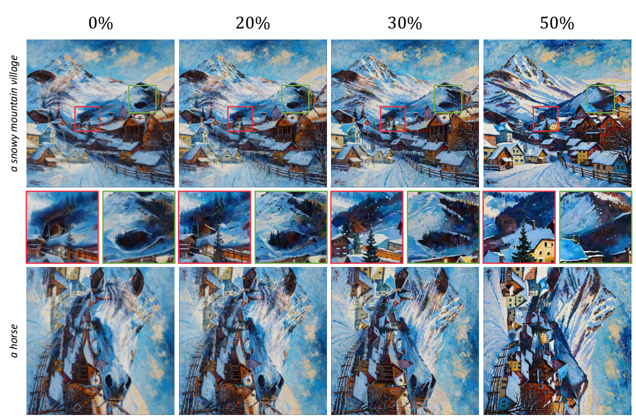



Figure 17. **Prioritizing a single view.** We show the effect of prioritizing the first view (*"a snowy mountain village"*) over the second (*"a horse"*) in the example of 90° rotation.

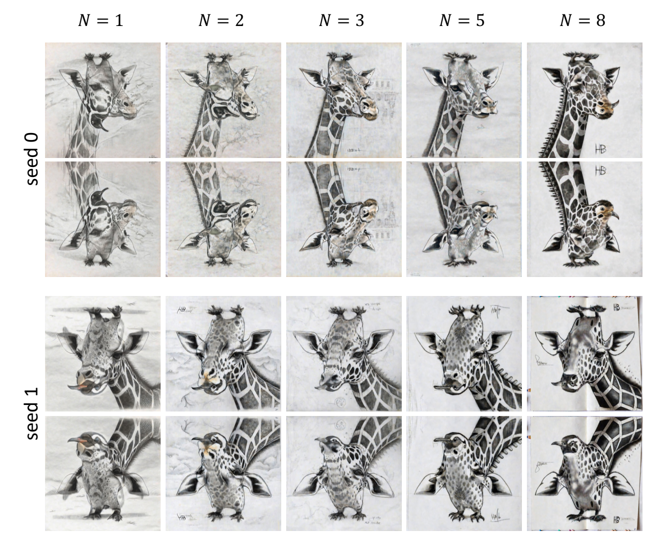



Figure 18. **Time travel.** We show the effect of time traveling for the flip example ("a giraffe head" / "a penguin") with two different sampled seeds. Time traveling is only applied for timesteps between 20% and 80%, by repeating each step $N$ times.

## B.2. Time travel

We show in Figure 18 that time traveling is an effective strategy for improving image consistency and blending between views. Without time traveling ($N = 1$), the *penguin* and the *giraffe head* seem like two independent entities overlayed on top of each other in the image. As the repeating number $N$ increases, the two views align better, resulting in a sharper image with more coherent details.

Obviously, runtime scales linearly with the repeating number. In Burgert et al. [4], image quality and blending quality are entangled, and improving with the number of optimization steps. Here, time traveling is only responsible for blending quality, but is independent of the generated image quality. We believe this provides a more intuitive control parameter than the number of optimization steps.

Naturally, better blending still yields better overall quality, which explains the lower FID in Table 2. However, it seems to come at the cost of slightly worse prompt alignment. We hypothetize that blending better means making more compromise between views, which in turn makes it harder to match the prompts as well.

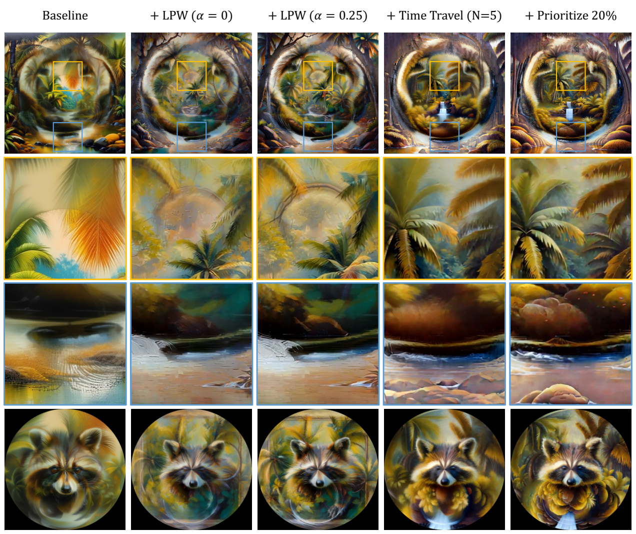



Figure 19. **Baseline ablation.** We consider the cone mirror example with two views (*"a tropical jungle forest"/"a raccoon"*. We show successively the effect of Laplacian Pyramid Warping (LPW), value-weighted average ($\alpha$), time traveling, and single-view prioritization.

### B.3. Comparison with Baseline

Lastly, we show in Figure 19 the effects of these different features. Laplacian Pyramid Warping addresses the artifacts at the edge of the ring, visible in the baseline. Value-weighted average recovers sharp details lost in standard averaging. Time traveling ensures a smoother blending, while single-view prioritization adds back the final details in the identity view without destroying the second view.

## C. Quantitative Evaluation Details

We provide additional details regarding the quantitative evaluation in this section.

**Dataset generation.** Similar to Geng *et al.* [18], we use a custom list of 50 prompt pairs in the form of `[<style>, <prompt1>, <prompt2>]` for our quantitative evaluation. These are a mix between prompt pairs from Visual Anagrams' paper [18], and from querying ChatGPT. For each method, we generate 10 samples per prompt pair. This results in 500 pairs of images, *i.e.* 1k images per method. For FID/KID computation, we generated a reference dataset comprised of 3.2k images from SD3 and 3.2k images from SD3.5 following the same prompts (only single view).

**Comparing with prior work.** For Visual Anagrams [18], we used the original codebase from Geng *et al.*: https://github.com/dangeng/visual_anagrams. Results are generated with DeepFloyd IF [1] in two stages, and subsequently up-sampled to $1024 \times 1024$ using Stable Diffusion x4 Upscaler, as provided by the code.

For Burgert et al. [4], we used the available codebase at (https://github.com/RyannDaGreat/Diffusion-Illusions) and added a function to rotate the inner circle of an image by 135°. Each image is generated with 1000 optimization steps using Stable Diffusion v1.4, the default model from their codebase.

Lastly, we set the hyperparameters of our pipeline to replicate the original implementation of Tancik [37] and SyncTweedies [23]. The results are generated with Stable Diffusion 3.5 Medium, as in our method.

**Runtime.** We generate our results with Stable Diffusion 3.5 Medium on a Nvidia GeForce RTX 4090 GPU. With 30 steps of inference and time traveling between 20% and 80% repeating 2 times, we can generate an image pair in $\sim$ 80s.

| Geng *et al.* [18] | Tancik *et al.* [37] | SyncTweedies [23] | Burgert *et al.* [4] | Ours SD 3.5 |
|---|---|---|---|---|
| 18.6s | 17.2s | 18.8s | 176.0s | 79.4s |

Table 3. **Inference time comparison.**

## D. Using Other Latent Models

Most of our results were generated using Stable Diffusion 3.5. However, we also experimented with other models such as SD2.1 and SDXL.

Here, we show a simple experiment with *two identity views* associated with two distinct prompts. This way, we abstract out the VAE, as well as other problems that come from warping. The results are shown in Figure 20 for two pairs of prompts. Interestingly, even in such a simple setting, not all models behave the same: SD2.1 tend to generate images with poor quality, while SD3+ attempts to blend the two concepts by compositing them as much as possible. Curiously, SDXL seems to blend the concepts *semantically*: "a snowy mountain village" and "a horse" give horses *in* a village, while "people at a campfire" and "an old man" produce "old men at a campfire".

While intriguing, this is not ideal for ambiguous images, as the goal is more to blend *spatially* the different views. Further investigation is needed to explain the discrepancy between SDXL and SD3+, possibly due to the switch from diffusion to flow matching or from U-Net to a transformer backbone. A deeper study is left for future work.


Figure 20. **Comparing different models for two-view setting.** We consider the case of two identity views, associated with two distinct prompts. While most models blend the two concepts *spatially*, SDXL seems to blend them *semantically*.

## E. User Study

As mentioned in Section 5.5, we conducted a user study over 27 participants to compare human preferences between our proposed method and existing prior work.

**Study structure.** The study consists of three sections, each corresponding to a different type of anamorphosis: cylindrical mirror, conic mirror, and Niceron's lens. Each section begins with an example image and video demonstrating how the illusion works. Participants are then shown 10 different samples generated from 10 pairs of prompts. These prompts remain the same across all three sections to avoid bias. Additionally, the order in which the methods are presented is randomized for each sample. To ensure a fair comparison, we first display the results in high resolution before asking users to rank them (see Fig. 21).

**Ranking criteria.** Participants are asked to provide a ranking of the five methods from 1 (best) to 5 (worst). At the beginning of the study, they are instructed to evaluate the images based on the following criteria:

- **Match both text prompts:** When viewed directly (resp. through a mirror or lens), the image should correspond to the first (resp. second) prompt.
- **Maintain the specified style:** The image should adhere to the given style prompt (*e.g.* painting, photograph).
- **Be high-quality:** The final image should be sharp, detailed, and visually appealing.

## F. Failed Experiments

**Relative negative prompting.** In Visual Anagrams [18], the authors note that including other views in the negative prompt does not improve substantially the visual quality. Moreover, prompt conflicts can arise. For example, the prompts *"an oil painting of a village"* and *"an oil painting of a horse"* both share the style descriptor *"an oil painting of"*, which should not be included in the negative prompt. Similarly, prompts like *"a cat"* and *"a dog"* share features such as fur. If these shared features appear in both the negative and positive prompts, it can lead to suboptimal results.

We experimented with a variant of this, which we dub *relative negative prompting*. The idea is to put in the negative prompt only the relative direction between the positive prompt (from the current view) and the negative prompt (from the other view). Thus, we subtract the positive prompt embedding from the negative one. Our experiments showed that this removes the need to manually select which part to keep for the negative prompt. In the example of the shared style, our difference vector cancels it out automatically, and the model is thus still able to generate the correct style. However, similar to Geng *et al.* [18], we did not observe substantial improvement using this method.

## G. Choosing Prompts & Failure Cases

The quality of the generation relies on choosing a good style and pair of prompts. Here are some tips we found for generating good anamorphoses:

1. a place or location (*e.g.* jungle, desert, library etc.) gives a lot of freedom to the composition and generally works well for the identity view;
2. the second view is generally seen through some mirror or a lens, which is smaller than the main image. For this view, easily recognizable subjects like animals or faces are good prompts in most cases;
3. artistic styles are more prone to produce good results than photorealistic styles;

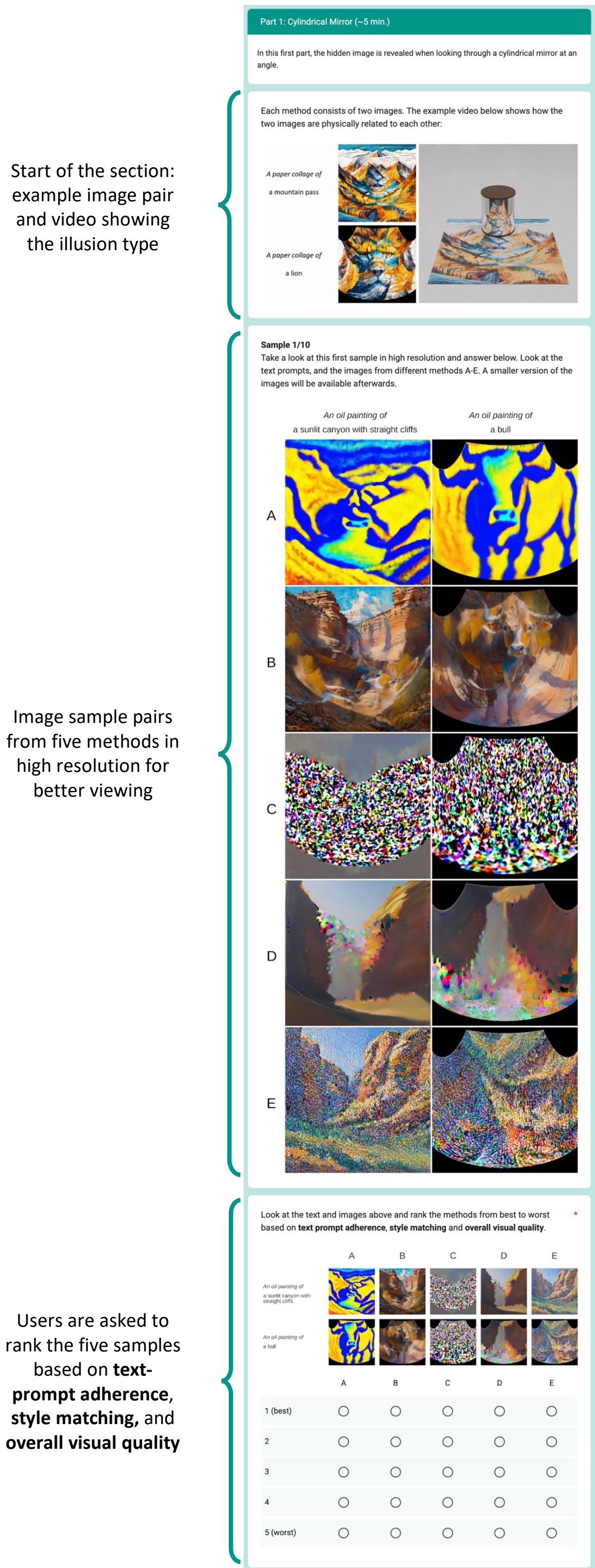



Figure 21. **Sample from the user study.**

4. styles with no colors (*e.g.* sketches, ink, marble sculpture) will generate better results when the two prompts have very different color palettes.

Our method is still prone to fail in certain cases. For example, the model can still cheat and put all the views in the image without properly blending them (see Figure 22).

Figure 22. **Failure case.** Similar to Geng *et al.*, our method can cheat and put both views in the image without properly mixing them. In this example, the shark from the cylinder mirror view can be seen in the sky behind the wind turbines.

## H. Concurrent Work

After our initial submission, a preprint titled "*Illusion3D: 3D Multiview Illusion with 2D Diffusion Priors*" [16] appeared on arXiv, addressing a similar problem. While their code is not available at the time of writing, we identified a few key differences between our method and theirs.

First, Illusion3D builds on optimization-based approaches like Burgert et al. [4], whereas our method improves upon feedforward techniques [18, 37], generating images in a single inference pass. Their approach replaces Score Distillation Sampling (SDS) [4] with Variational Score Distillation (VSD), which requires training a LoRA module during optimization. Due to the inherent limitations of score distillation methods, we believe our approach produces higher-quality images while supporting a broader range of styles, from artistic to photorealistic, as demonstrated in Section I.

A key advantage of Illusion3D, however, is its ability to generate full 3D structures, which our method does not support. That said, our approach can still generate 2D textures for mapping onto 3D surfaces, similar to their sphere or cube illusions. Lastly, we expect our method to have significantly lower generation time and memory requirements compared to Illusion3D.

## I. Additional Results

In the next pages, we show additional qualitative results for the three anamorphic views: cylinder mirror, conic mirror, and Nicéron's lens. Please refer to the supplementary videos to see these anamorphoses in action. Table 4 shows additional quantitative evaluations.

| | Method | $\mathcal{S}\uparrow$ | $\mathcal{S}_{0.9}\uparrow$ | $\mathcal{A}\uparrow$ | $\mathcal{A}_{0.9}\uparrow$ | $\mathcal{C}\uparrow$ | $\mathcal{C}_{0.9}\uparrow$ | FID $\downarrow$ | KID $\downarrow$ |
|---|---|---|---|---|---|---|---|---|---|
| Vertical Flip | Geng *et al.* [18] | 0.325 | 0.362 | 0.306 | 0.340 | 0.695 | 0.786 | 149.24 | 0.057 |
| | Tancik SD 3.5 [37] | 0.328 | 0.367 | 0.306 | 0.349 | 0.693 | 0.806 | 132.52 | 0.049 |
| | Burgert *et al.* [4] | 0.303 | 0.347 | 0.281 | 0.324 | 0.679 | 0.778 | 219.84 | 0.115 |
| | SyncTweedies [23] | 0.323 | 0.360 | 0.302 | 0.341 | 0.707 | 0.801 | 132.62 | 0.054 |
| | **LookingGlass (ours)** | **0.320** | **0.358** | **0.297** | **0.338** | **0.680** | **0.779** | **124.67** | **0.049** |
| 135° Rotation | Geng *et al.* [18] | 0.284 | 0.340 | 0.262 | 0.308 | 0.563 | 0.652 | 293.00 | 0.254 |
| | Tancik SD 3.5 [37] | 0.203 | 0.225 | 0.194 | 0.216 | 0.498 | 0.509 | 439.35 | 0.545 |
| | Burgert *et al.* [4] | 0.301 | 0.347 | 0.280 | 0.326 | 0.654 | 0.760 | 223.21 | 0.120 |
| | SyncTweedies [23] | 0.308 | 0.354 | 0.283 | 0.335 | 0.647 | 0.753 | 166.03 | 0.083 |
| | **LookingGlass (ours)** | **0.319** | **0.357** | **0.295** | **0.338** | **0.666** | **0.767** | **129.74** | **0.055** |
| Cylindrical Mirror | Geng *et al.* [18] | 0.190 | 0.228 | 0.171 | 0.198 | 0.506 | 0.546 | 285.23 | 0.216 |
| | Tancik SD 3.5 [37] | 0.189 | 0.225 | 0.171 | 0.198 | 0.505 | 0.547 | 284.97 | 0.215 |
| | Burgert *et al.* [4] | 0.285 | 0.334 | 0.261 | 0.304 | 0.706 | 0.795 | 229.65 | 0.138 |
| | SyncTweedies [23] | 0.285 | 0.348 | 0.241 | 0.284 | 0.673 | 0.763 | 138.69 | 0.082 |
| | **LookingGlass (ours)** | **0.307** | **0.360** | **0.272** | **0.318** | **0.698** | **0.810** | **130.27** | **0.070** |

Table 4. **Additional quantitative comparison.** We additionally assess image-prompt alignment using CLIP similarity score $\mathcal{S}$ on all three transformations evaluated in the main paper. While all methods achieve comparable results for the vertical flip, LookingGlass surpasses previous approaches on more complex transformations, including anamorphoses.

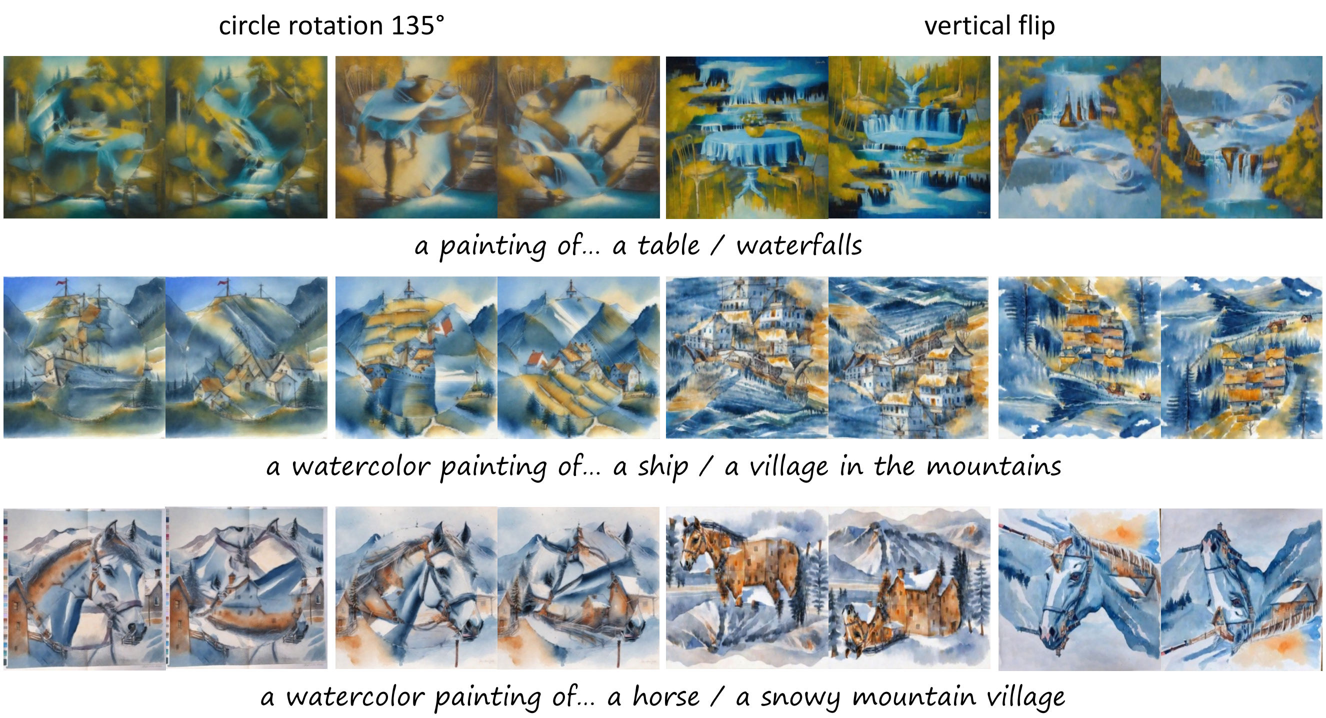



Figure 23. **2D transform results.** Here are some generated results for the two 2D transforms: vertical flip, and 135° rotation (not supported by Geng *et al.* [18]).

*a cinematic rendering of*
rolling hills in golden light / turtle

*an oil painting of*
sunlit canyon with straight cliffs / bull

*an oil painting of*
desert dunes at sunset / jellyfish

*a clay sculpture of*
cobblestone street / cheetah

*a charcoal drawing of*
flower garden with rows of tulips / fox

*a paper collage of*
mountain pass / lion

Figure 24. **Cylinder mirror anamorphosis.** In this figure and the two following ones, we show additional results for the cylinder mirror example. Each example contains the identity view, the mirror view as predicted by the flow model, and a rendering of the actual physical setting to validate our examples. Kindly refer to the supplementary videos to see these results in action.

*an ink wash drawing of* straight ski tracks

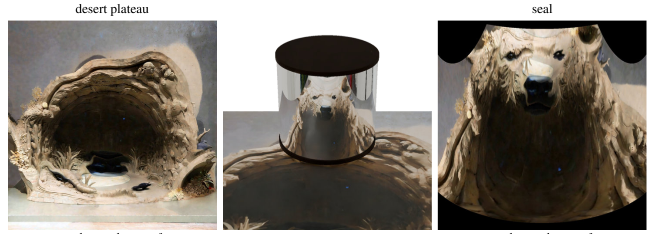
*an ink wash drawing of* iguana

*a watercolor painting of* desert plateau

*a watercolor painting of* seal

*a LEGO model of* cobblestone street

*a LEGO model of* cheetah



*a clay sculpture of* aquarium tunnel

*a clay sculpture of* polar bear

*a fresco painting of* aquarium tunnel

*a fresco painting of* knight's helmet

*an oil painting of* straight levee dividing water

*an oil painting of* otter

*a diorama of* dense tropical rainforest

*a diorama of* panther

*a low-poly model of* sunlit rocky outcrop

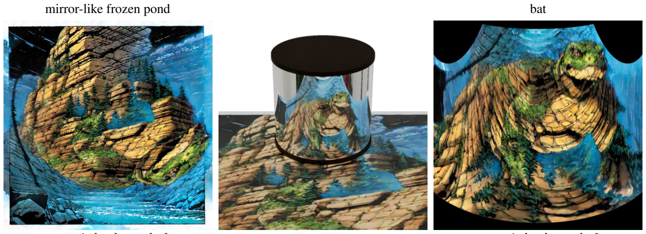
*a low-poly model of* fox

*a fresco painting of* mirror-like frozen pond

*a fresco painting of* bat

*a clay sculpture of* flower meadow at sunrise

*a clay sculpture of* iguana



*a comic book panel of* sunlit rocky outcrop

*a comic book panel of* turtle

*a lithograph of* frozen waterfall

*a lithograph of* cougar

*an oil painting of* highland moor with stone walls

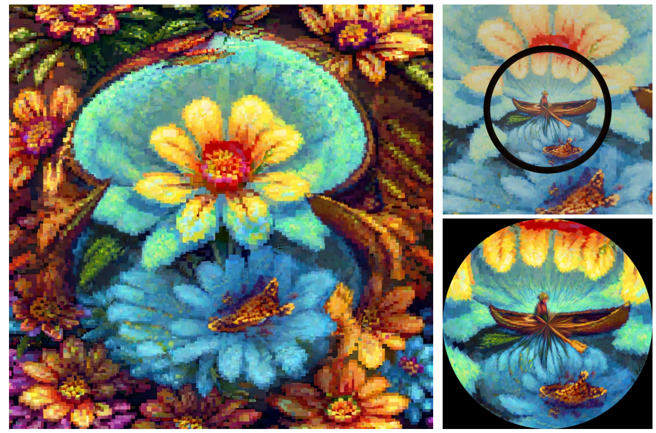
*an oil painting of* hedgehog

*a hyperrealistic sculpture of* windy desert

*a hyperrealistic sculpture of* walrus

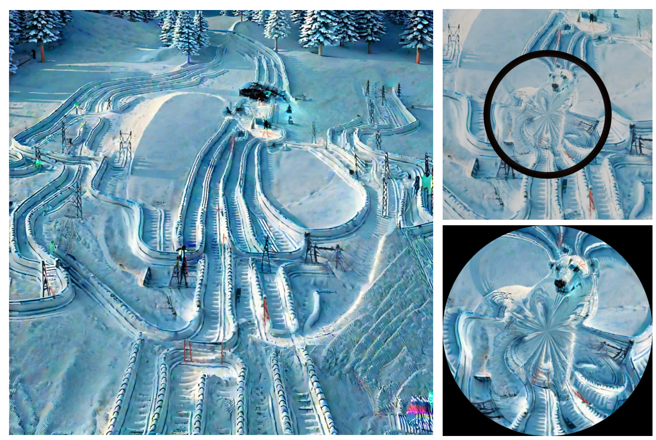

*an ink wash drawing of* straight ski tracks
*an ink wash drawing of* iguana
*a watercolor painting of* snow-covered green trees
*a watercolor painting of* hedgehog
*a watercolor painting of* desert plateau
*a watercolor painting of* seal
*a LEGO model of* cobblestone street
*a LEGO model of* cheetah
*a photorealistic painting of* desert oasis
*a photorealistic painting of* bat
*a clay sculpture of* aquarium tunnel
*a clay sculpture of* polar bear
*a fresco painting of* aquarium tunnel
*a fresco painting of* knight's helmet
*a LEGO model of* medieval castle gate
*a LEGO model of* ant
*a charcoal drawing of* sunlit rocky outcrop
*a charcoal drawing of* seal
*a charcoal drawing of* sunlit rocky outcrop
*a charcoal drawing of* seal
*an oil painting of* sunlit canyon with straight cliffs
*an oil painting of* bull
*an oil painting of* straight levee dividing water
*an oil painting of* otter
*an oil painting of* desert dunes at sunset
*an oil painting of* jellyfish
*a diorama of* dense tropical rainforest
*a diorama of* panther
*a clay sculpture of* cobblestone street
*a clay sculpture of* cheetah
*an ink wash drawing of* horizon of a wheat field
*an ink wash drawing of* frog
*a low-poly model of* sunlit rocky outcrop
*a low-poly model of* fox
*a charcoal drawing of* sunlit canyon with cliffs
*a charcoal drawing of* panther
*a charcoal drawing of* sunlit canyon with cliffs
*a charcoal drawing of* panther
*a clay sculpture of* flower meadow at sunrise
*a clay sculpture of* iguana
*a charcoal drawing of* flower garden with tulips
*a charcoal drawing of* fox
*a line drawing of* desert dunes at sunset
*a line drawing of* horse
*a cubist interpretation of* desert canyon floor
*a cubist interpretation of* eagle
*a paper collage of* mountain pass
*a paper collage of* lion
*a lithograph of* frozen waterfall
*a lithograph of* cougar
*a bronze statue of* mangrove forest
*a bronze statue of* cougar
*a metal engraving of* rooftops of a dense city
*a metal engraving of* fox
*a metal engraving of* rooftops of a dense city
*a metal engraving of* fox
*oil painting of* highland moor with stone walls
*oil painting of* hedgehog
*a hyperrealistic sculpture of* windy desert
*a hyperrealistic sculpture of* walrus

*a clay sculpture of*
aquarium tunnel / polar bear



*a pixel art version of*
flower petals close-up / canoe

*a stained glass depiction of*
straight coastline / lion

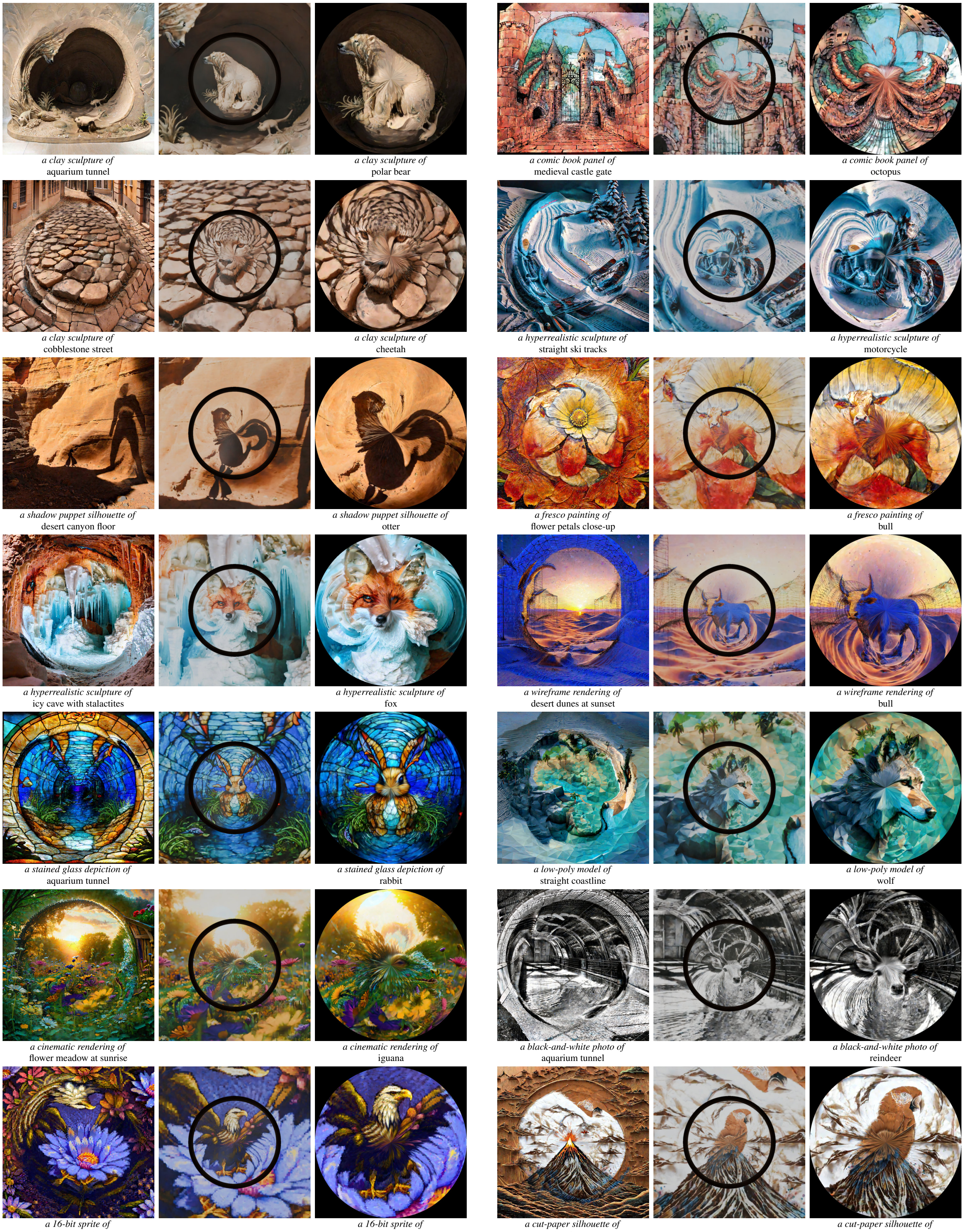


*a 3D rendering of*
straight ski tracks / polar bear

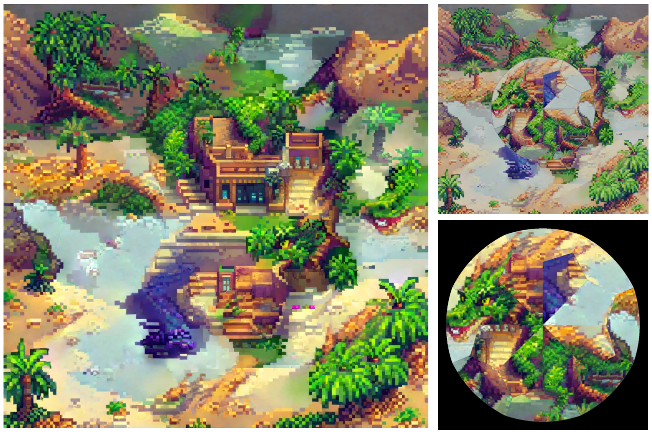
*a cinematic rendering of*
icy cave with stalactites / parrot

*a pointillism painting of*
straight canal lined with trees / butterfly

Figure 27. **Conic mirror anamorphosis.** In this figure and the two following ones, we show additional results for the conic mirror example. Each example contains the identity view, the mirror view as predicted by the flow model, and a rendering of the actual physical setting from the top to validate our examples. Kindly refer to the supplementary videos to see these results in action.


*a clay sculpture of*
aquarium tunnel
*a clay sculpture of*
polar bear
*a comic book panel of*
medieval castle gate
*a comic book panel of*
octopus
*a clay sculpture of*
cobblestone street
*a clay sculpture of*
cheetah
*a hyperrealistic sculpture of*
straight ski tracks
*a hyperrealistic sculpture of*
motorcycle
*a shadow puppet silhouette of*
desert canyon floor
*a shadow puppet silhouette of*
otter
*a fresco painting of*
flower petals close-up
*a fresco painting of*
bull
*a hyperrealistic sculpture of*
icy cave with stalactites
*a hyperrealistic sculpture of*
fox
*a wireframe rendering of*
desert dunes at sunset
*a wireframe rendering of*
bull
*a stained glass depiction of*
aquarium tunnel
*a stained glass depiction of*
rabbit
*a low-poly model of*
straight coastline
*a low-poly model of*
wolf
*a cinematic rendering of*
flower meadow at sunrise
*a cinematic rendering of*
iguana
*a black-and-white photo of*
aquarium tunnel
*a black-and-white photo of*
reindeer
*a 16-bit sprite of*
flower petals close-up
*a 16-bit sprite of*
eagle
*a cut-paper silhouette of*
volcanic crater
*a cut-paper silhouette of*
macaw


a stained glass depiction of spiral staircase
a stained glass depiction of hermit crab
a chalkboard drawing of open-pit mine
a chalkboard drawing of bee
a diorama of icebergs in the ocean
a diorama of shark
a comic book panel of sunlit rocky outcrop
a comic book panel of turtle
a marble carving of city skyline at night
a marble carving of horse
a ceramic tile mural of windy desert
a ceramic tile mural of knight's helmet
a stained glass depiction of shipyard with cranes
a stained glass depiction of lantern
a stained glass depiction of shipyard with cranes
a stained glass depiction of lantern
a low-poly model of meandering river in valley
a low-poly model of hermit crab
a diorama of medieval castle gate
a diorama of knight's helmet
a 3D rendering of volcanic crater
a 3D rendering of polar bear
a 3D rendering of volcanic crater
a 3D rendering of polar bear
a steampunk illustration of straight ski tracks
a steampunk illustration of komodo dragon
low-poly model of beach, shoreline & waves
low-poly model of fountain statue
an oil painting of icy cave with stalactites
an oil painting of butterfly
a digital illustration of aquarium tunnel
a digital illustration of canoe
a minimalist illustration of industrial pipeline
a minimalist illustration of polar bear
a holographic image of sunlit rocky outcrop
a holographic image of jellyfish
a fresco painting of flower petals close-up
a fresco painting of gorilla
a minimalist illustration of mountain pass
a minimalist illustration of zebra
a low-poly model of open-pit mine
a low-poly model of octopus
a comic book panel of icy cave with stalactites
a comic book panel of wolf
a cinematic rendering of ocean waves
a cinematic rendering of horse
a marble carving of desert oasis
a marble carving of komodo dragon
a marble carving of desert oasis
a marble carving of komodo dragon
a surrealist painting of mountain ridge
a surrealist painting of octopus
a futuristic concept art of bamboo forest
a futuristic concept art of fox
a futuristic concept art of bamboo forest
a futuristic concept art of fox
a 16-bit sprite of mountain pass
a 16-bit sprite of gecko
a line drawing of mirror-like frozen pond
a line drawing of lizard



*a 16-bit sprite of*
desert oasis / dragon

*a lithograph of*
horizon of a wheat field / rabbit

*a vintage poster of*
dense tropical rainforest / deer

*a hyperrealistic sculpture of*
straight ski tracks / motorcycle

*a hyperrealistic sculpture of*
windy desert / walrus

*a pencil sketch of*
harbor pier / lion

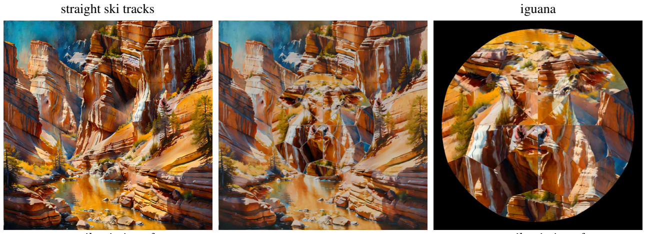
Figure 30. **Nicéron's lens anamorphosis.** In this figure and the two following ones, we show additional results for the lens example. Each example contains the identity view, the lens view as predicted by the flow model, and a rendering of the actual image through the lens to validate our examples. Kindly refer to the supplementary videos to see these results in action.

*an ink wash drawing of*
straight ski tracks

*an ink wash drawing of*
iguana

*a LEGO model of*
cobblestone street

*a LEGO model of*
cheetah

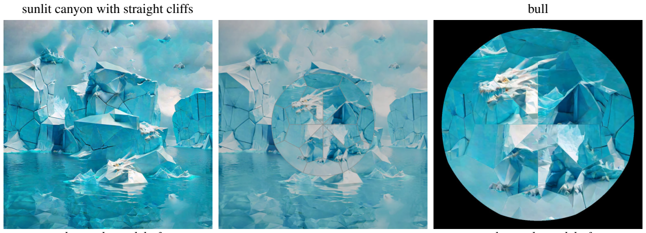


*an oil painting of*
sunlit canyon with straight cliffs

*an oil painting of*
bull

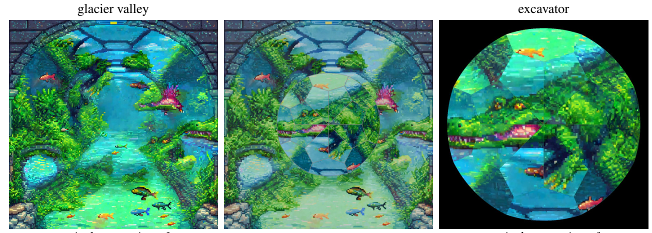
*a photo of*
glacier valley

*a photo of*
excavator



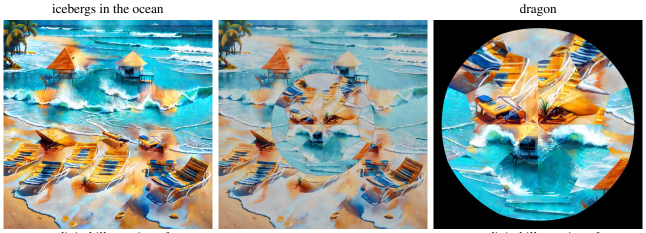
*a low-poly model of*
icebergs in the ocean

*a low-poly model of*
dragon



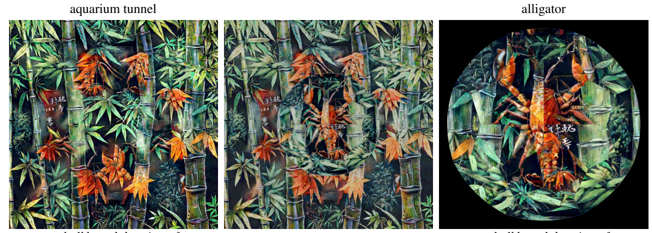
*a pixel art version of*
aquarium tunnel

*a pixel art version of*
alligator



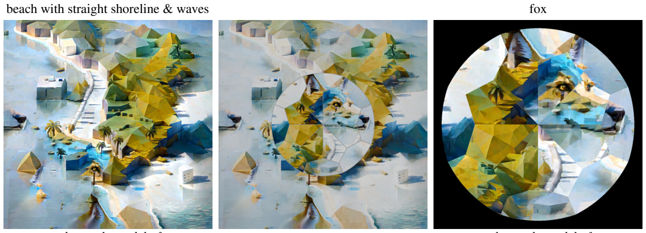
*a digital illustration of*
beach with straight shoreline & waves

*a digital illustration of*
fox



*a chalkboard drawing of*
bamboo forest

*a chalkboard drawing of*
lobster

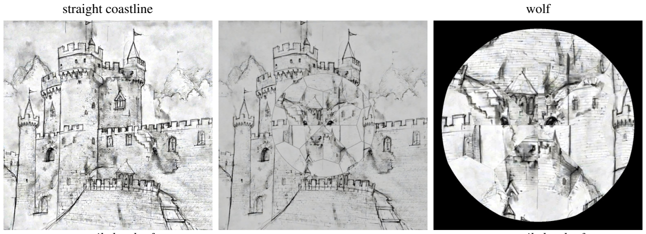


*a low-poly model of*
straight coastline

*a low-poly model of*
wolf

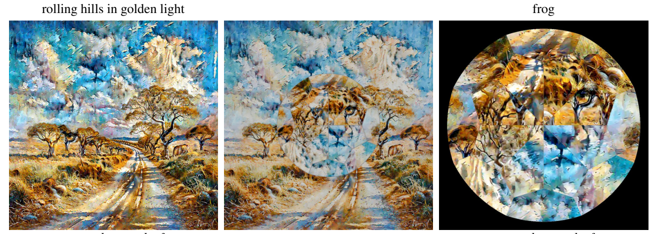
*an embroidered version of*
rolling hills in golden light

*an embroidered version of*
frog



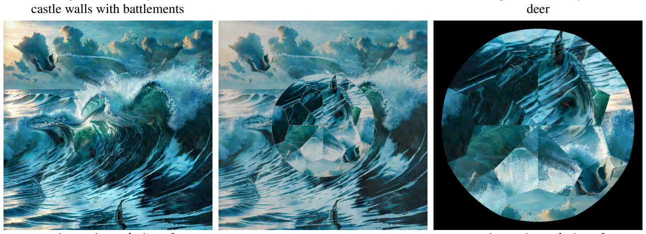
*a pencil sketch of*
castle walls with battlements

*a pencil sketch of*
deer



*a pastel artwork of*
straight dirt road through a savannah

*a pastel artwork of*
cheetah



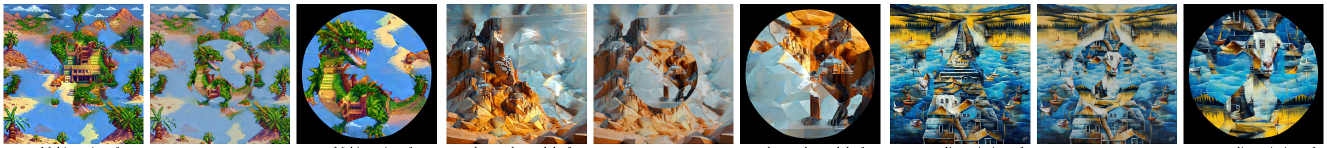
*a cinematic rendering of*
ocean waves

*a cinematic rendering of*
horse

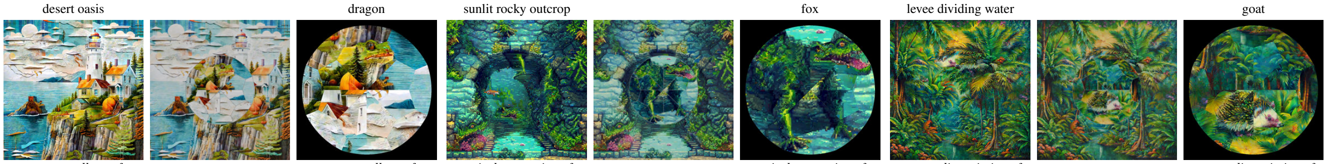
*a marble carving of*
horizon of a wheat field

*a marble carving of*
cougar

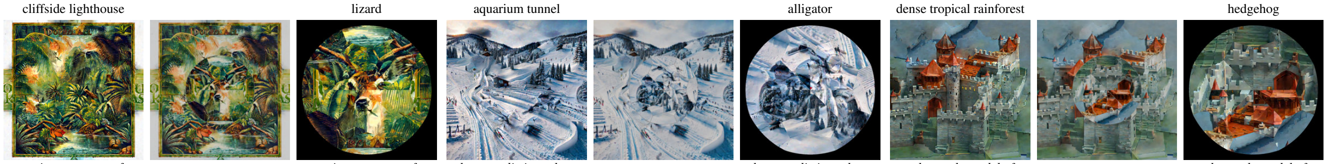
*an ink wash drawing of* straight ski tracks
*an ink wash drawing of* iguana
*a cinematic rendering of* windy desert
*a cinematic rendering of* wagon
*a pixel art version of* medieval castle gate
*a pixel art version of* wine bottle



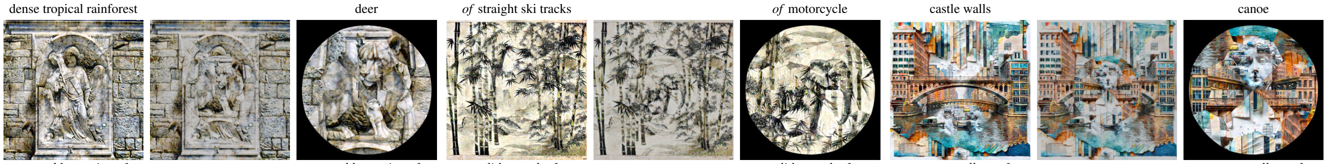
*a 16-bit sprite of* desert oasis
*a 16-bit sprite of* dragon
*a low-poly model of* sunlit rocky outcrop
*a low-poly model of* fox
*a surrealist painting of* levee dividing water
*a surrealist painting of* goat



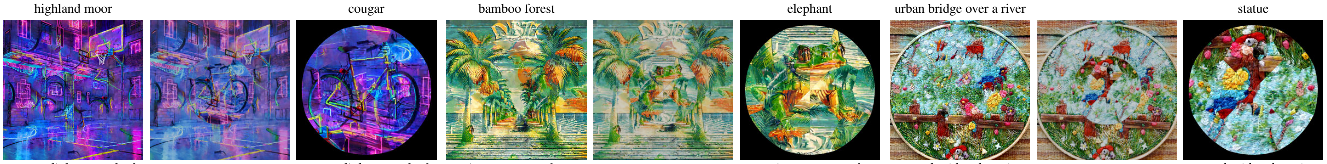
*a paper collage of* cliffside lighthouse
*a paper collage of* lizard
*a pixel art version of* aquarium tunnel
*a pixel art version of* alligator
*a surrealist painting of* dense tropical rainforest
*a surrealist painting of* hedgehog



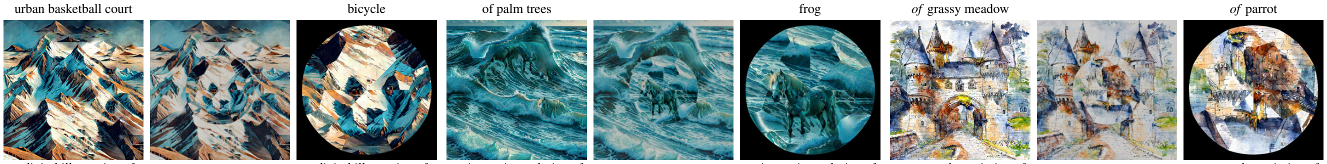
*a vintage poster of* dense tropical rainforest
*a vintage poster of* deer
*a hyperrealistic sculpture of* straight ski tracks
*a hyperrealistic sculpture of* motorcycle
*a low-poly model of* castle walls
*a low-poly model of* canoe



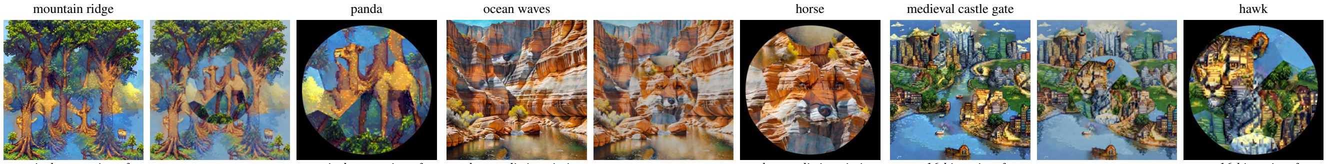
*a marble carving of* highland moor
*a marble carving of* cougar
*a lithograph of* bamboo forest
*a lithograph of* elephant
*a paper collage of* urban bridge over a river
*a paper collage of* statue



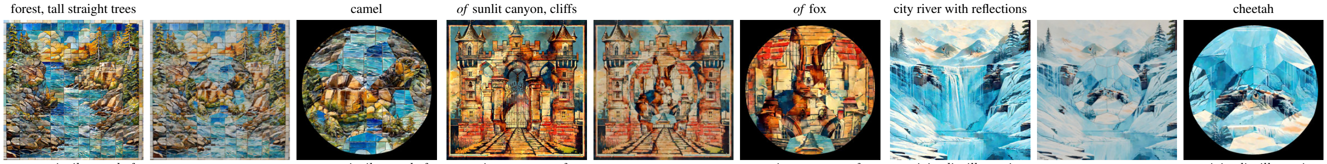
*a neon light artwork of* urban basketball court
*a neon light artwork of* bicycle
*a vintage poster of* rows of palm trees
*a vintage poster of* frog
*an embroidered version of* grassy meadow
*an embroidered version of* parrot



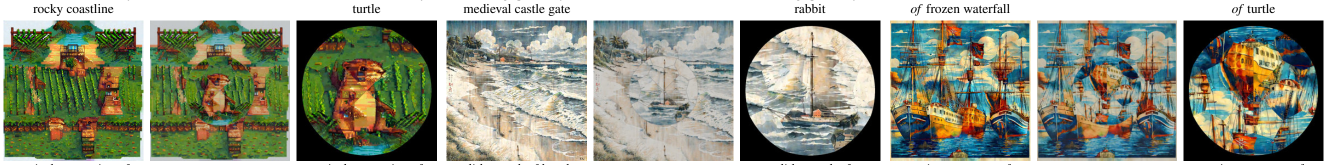
*a digital illustration of* mountain ridge
*a digital illustration of* panda
*a cinematic rendering of* ocean waves
*a cinematic rendering of* horse
*a watercolor painting of* medieval castle gate
*a watercolor painting of* hawk



*a pixel art version of* forest, tall straight trees
*a pixel art version of* camel
*a photorealistic painting of* sunlit canyon, cliffs
*a photorealistic painting of* fox
*a 16-bit sprite of* city river with reflections
*a 16-bit sprite of* cheetah



*a ceramic tile mural of* rocky coastline
*a ceramic tile mural of* turtle
*a vintage poster of* medieval castle gate
*a vintage poster of* rabbit
*a minimalist illustration of* frozen waterfall
*a minimalist illustration of* turtle



*a pixel art version of* vineyard, straight trellises
*a pixel art version of* otter
*a lithograph of* beach, shoreline & waves
*a lithograph of* sailboat
*a vintage poster of* harbor with docked ships
*a vintage poster of* hot air balloon