Fast Sprite Decomposition from Animated Graphics (Supplementary Material)

Tomoyuki Suzuki[1], Kotaro Kikuchi[1], and Kota Yamaguchi[1]

CyberAgent
{suzuki_tomoyuki,kikuchi_kotaro_xa,yamaguchi_kota}@cyberagent.co.jp

A Dataset details

Tab. 3 summarizes the description and the number of each animation type in the Crello Animation dataset. Each sprite has one of the animation types or no animation. All animation types can be represented by affine transformation and opacity changes. In addition to the animation type, each sprite has a delay parameter, which specifies the start time of the animation. We set the duration of all videos to 5 seconds and adjusted the speed of each animation accordingly, as in the actual rendering engine[1]. We set the frame rate to 10 for our experiments, but it can be set to any value as the original animations are continuous functions of time.

Fig. 8 shows the histogram of the number of sprites in each video and the aspect ratio. Though the frame resolution can be set to any value by changing the target size of the affine matrices, we set the short edge to 128 pixels for our experiments while keeping the original aspect ratio.

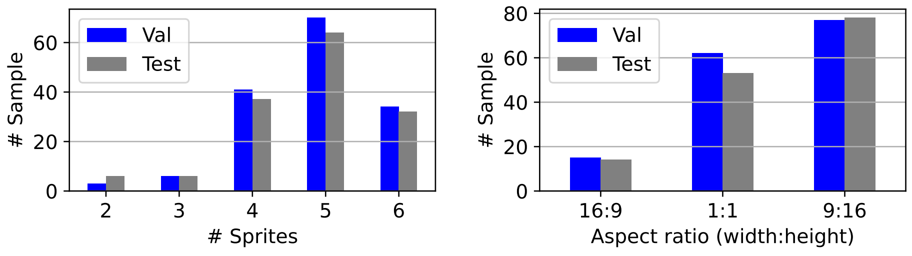

Fig. 8: Statistics in Crello Animation.

B Additional results

We provide additional quantitative results on the validation split in Tab. 4. As in the results on the test split in the main paper, our method achieves sprite errors

[1] https://create.vista.com

Table 3: Animation types in the Crello Animation dataset. "None" indicates that the sprite has no animation.

Type	# Sprites (Val.)	# Sprites (Test)	Description
Slide	244	242	Translation changes continuously. There are three types: *Slide-in*, where translation changes from the start position outside the frame to the base position; *Slide-out*, where translation changes inversely; and *Slide-both*, where both Slide-in and Slide-out occur sequentially.
Scale	165	150	Opacity and scale change continuously. There are three types: *Scale-in*, where opacity and scale change from 0 to 1; *Scale-out*, where opacity and scale change inversely; and *Scale-both*, where both Scale-in and Scale-out occur sequentially.
Fade	91	92	Opacity changes continuously. There are three types: *Fade-in*, where opacity changes from 0 to 1, *Fade-out*, where opacity changes inversely, and *Fade-both*, where both Fade-in and Fade-out occur sequentially.
Zoom	58	43	Scale changes continuously. There are three types: *Zoom-in*, where scale changes from 0 to 1; *Zoom-out*, where scale changes inversely; and *Zoom-both*, where both Zoom-in and Zoom-out occur sequentially.
Shake	26	12	Translation oscillates horizontally or vertically.
Spin	6	6	The object rotates around the center axis in the horizontal or vertical direction.
Flash	4	5	Opacity oscillates between 0 and 1.
None	210	194	–

comparable to the converged other methods even in 10 minutes, and achieves even lower sprite errors after convergence.

We also provide additional qualitative results. Figs. 9 to 12 show the comparison between Layered Neural Atlases (LNA) [1], Deformable sprites (DS) [2] and our method. As described in the main paper, our method consistently decomposes sprites with higher quality than LNA and DS, especially for sprites with complex contours such as text. Figs. 13 and 14 show more examples of the decomposition results with textures by our method. The output textures and ground truth textures may differ in the degrees of freedom of the affine transformation, but the output animations are adjusted accordingly so they are correctly reproduced as sprites.

We also show failure cases of our method. In the first example in Fig. 15, the close sprites with similar animations are difficult to decompose. In the second

Table 4: Quantitative comparison with prior works on the validation split. All values are averages of the samples. The best and the second best result for each metric are highlighted in **bold** and underlined, respectively. * indicates the results after optimization convergence.

Method	# Iter.	Time (min.)	Frame error ↓ L1	Frame error ↓ LPIPS	Sprite error ↓ RGB L1	Sprite error ↓ Alpha L1
LNA	3k	10.8	0.0308	0.2584	0.2422	0.0271
DS	9k	10.3	0.2771	0.4720	0.3497	0.0369
Ours	11k	10.5	0.0123	0.0510	0.1095	0.0116
LNA*	11k	40.6	0.0145	0.1151	0.2003	0.0214
DS*	16k	24.4	**0.0052**	**0.0167**	0.1068	0.0146
Ours*	91k	97.9	0.0090	0.0338	**0.0926**	**0.0094**

example in Fig. 16, the sprite with (almost) no animation tends to be absorbed into the background. These failure cases are challenging because they result in small reconstruction errors. Our initialization should function as a prior to avoid these failures, but further consideration of priors may be necessary.

C Baseline details

We describe the details of the comparison baselines, Layered Neural Atlases (LNA) [1] and Deformable Sprites (DS) [2].

C.1 Layered Neural Atlases

Based on the official code (updated version)[2], we make a minor modification and tune hyperparameters. For a fair comparison, we utilize the predicted foreground segmentation masks, which we also used in our method. Specifically, we add a binary cross-entropy loss to match the predicted alpha with the segmentation mask for each sprite, as the alpha bootstrapping loss in the original paper. We adopt the original paper's setting except for the weight of the flow alpha loss ($\beta_{f-\alpha}$ in their paper) set to 49 and the weight of the rigidity loss (β_r in their paper) set to 1. We set the weight of the additional binary cross entropy loss to $10,000$.

C.2 Deformable Sprites

We adopted the official implementation[3] to our problem and adjusted several hyperparameters for a fair comparison. DS uses an image prior model to represent texture images, similar to our method ($4.1). We apply the same strategy to initialize the textures as we do ($4.3). We also simplify the deformation as

[2] `https://github.com/thiagoambiel/NeuralAtlases`
[3] `https://github.com/vye16/deformable-sprites`

the affine transformation and initialize its corresponding parameters in a manner similar to our method. To incorporate the given segmentation masks, we employ the binary cross-entropy loss used in LNA to guide the predicted alpha masks. With a schedule ratio of 1:10 for warm start and main optimization, we set the weights of the added alpha loss and the dynamic grouping loss ($\mathcal{L}_{\text{dynamic}}$ in their paper) to 1.0 for the warm start; we set the weights of the reconstruction loss ($\mathcal{L}_{\text{recon}}$ in their paper) to 1.0, and the alpha and grouping losses to 0.01 for the main optimization. We omit other losses, such as optical flow consistency losses, because they do not work effectively in our data domain/problem setting.

References

1. Kasten, Y., Ofri, D., Wang, O., Dekel, T.: Layered neural atlases for consistent video editing. TOG (2021)
2. Ye, V., Li, Z., Tucker, R., Kanazawa, A., Snavely, N.: Deformable sprites for unsupervised video decomposition. In: CVPR (2022)

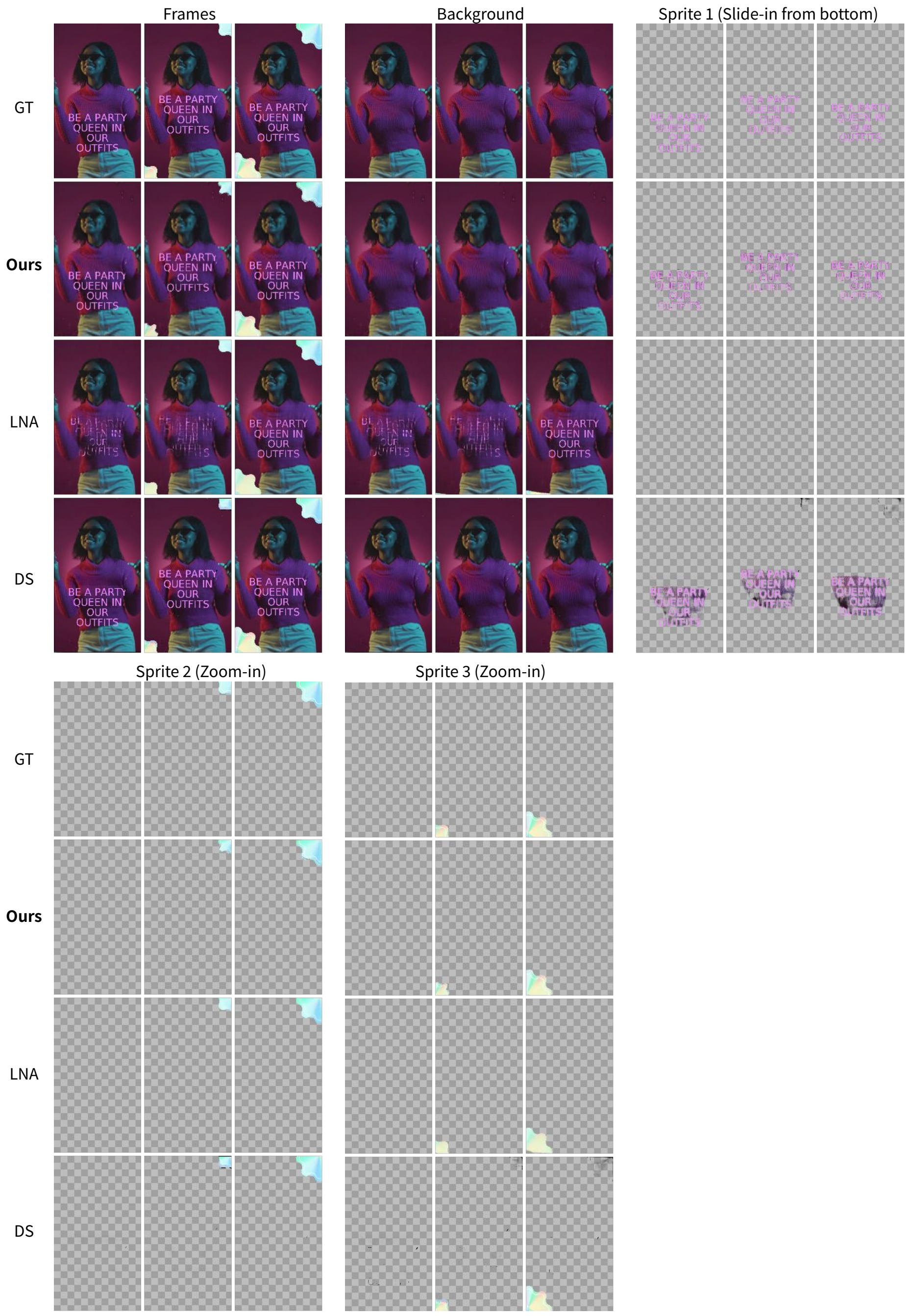

Fig. 9: Qualitative comparison between Layered Neural Atlases (LNA) [1], Deformable sprites (DS) [2], and our method. We put the description of the animation above each sprite. Best viewed with zoom and color.

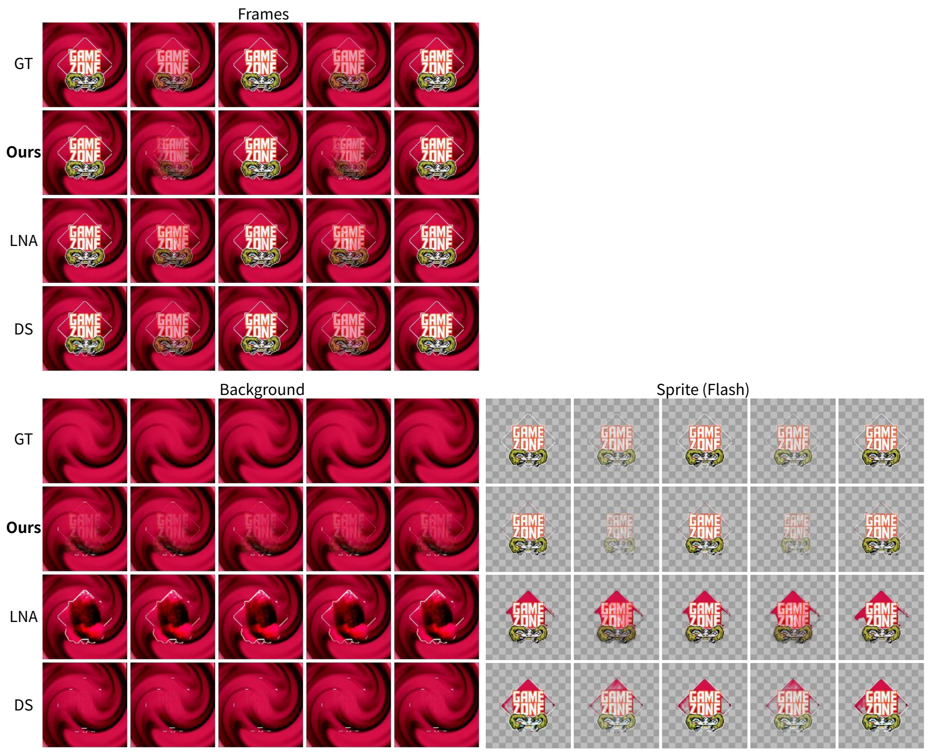

Fig. 10: Qualitative comparison between Layered Neural Atlases (LNA) [1], Deformable sprites (DS) [2], and our method. We put the description of the animation above each sprite. Best viewed with zoom and color.

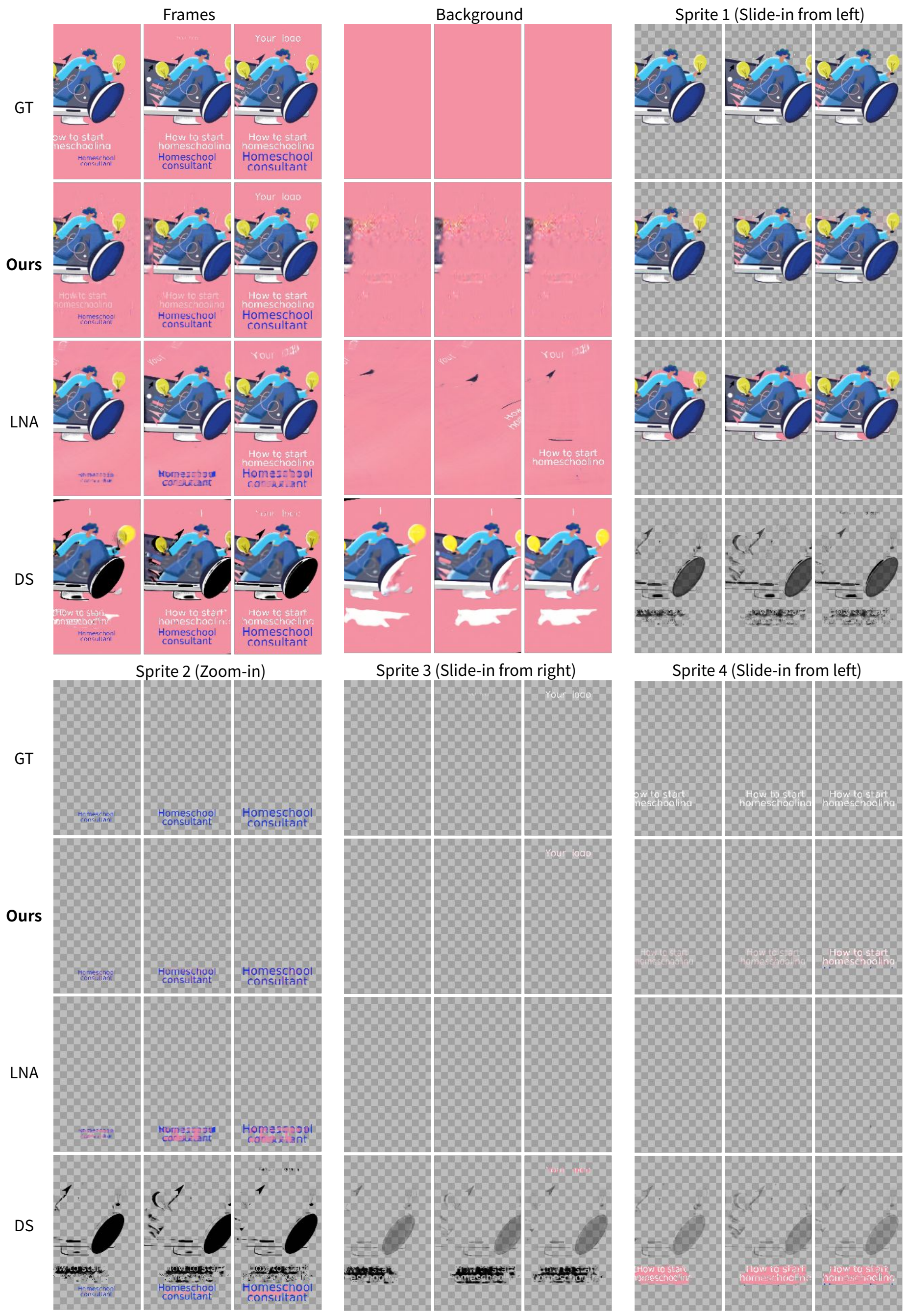
Frames
Background
Sprite 1 (Slide-in from left)
Sprite 2 (Zoom-in)
Sprite 3 (Slide-in from right)
Sprite 4 (Slide-in from left)
GT
Ours
LNA
DS
How to start homeschooling
Homeschool consultant
Your logo

Fig. 11: Qualitative comparison between Layered Neural Atlases (LNA) [1], Deformable sprites (DS) [2], and our method. We put the description of the animation above each sprite. Best viewed with zoom and color.

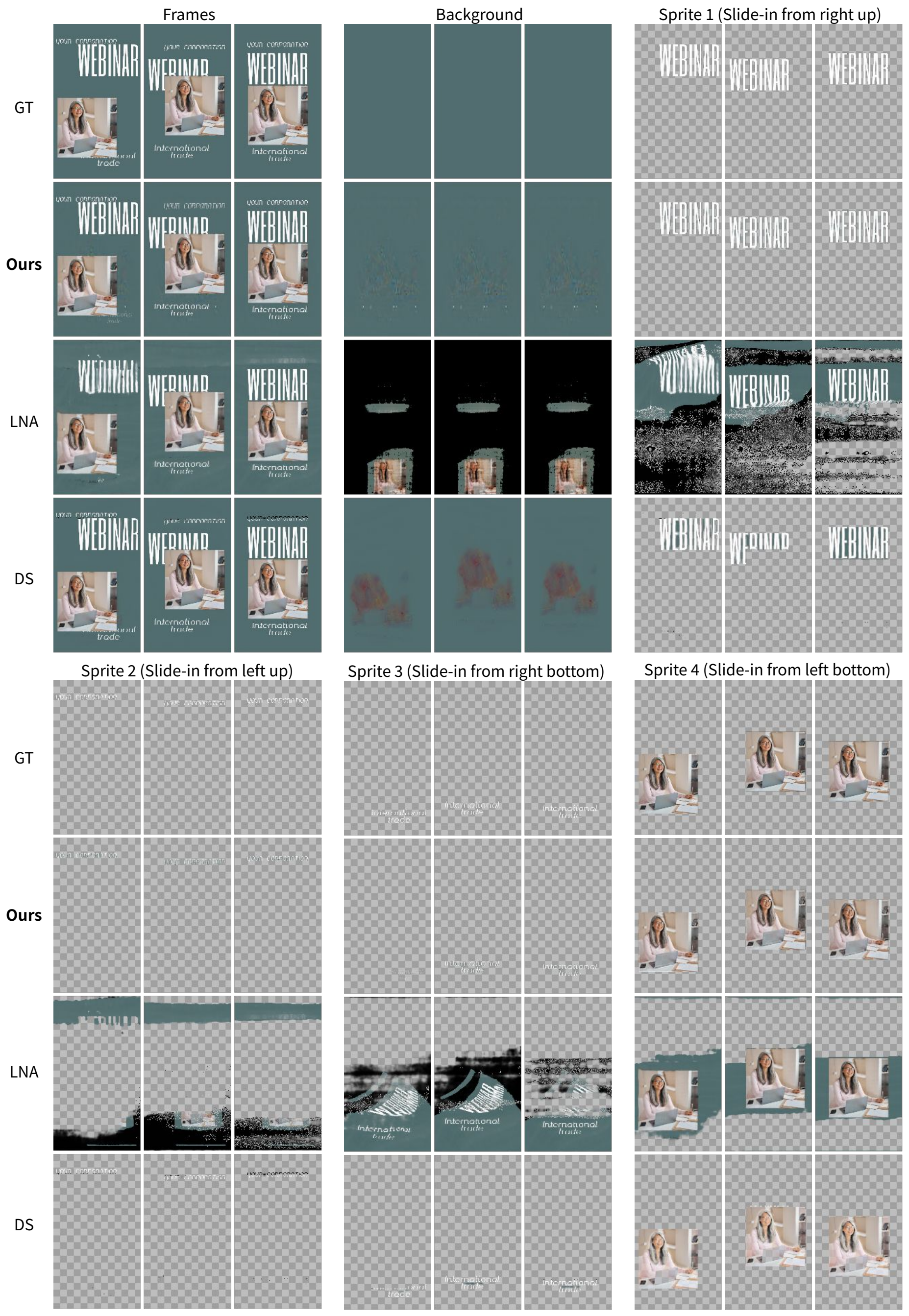
Frames
Background
Sprite 1 (Slide-in from right up)
GT
Ours
LNA
DS
Sprite 2 (Slide-in from left up)
Sprite 3 (Slide-in from right bottom)
Sprite 4 (Slide-in from left bottom)
WEBINAR
International trade

Fig. 12: Qualitative comparison between Layered Neural Atlases (LNA) [1], Deformable sprites (DS) [2], and our method. We put the description of the animation above each sprite. Best viewed with zoom and color.

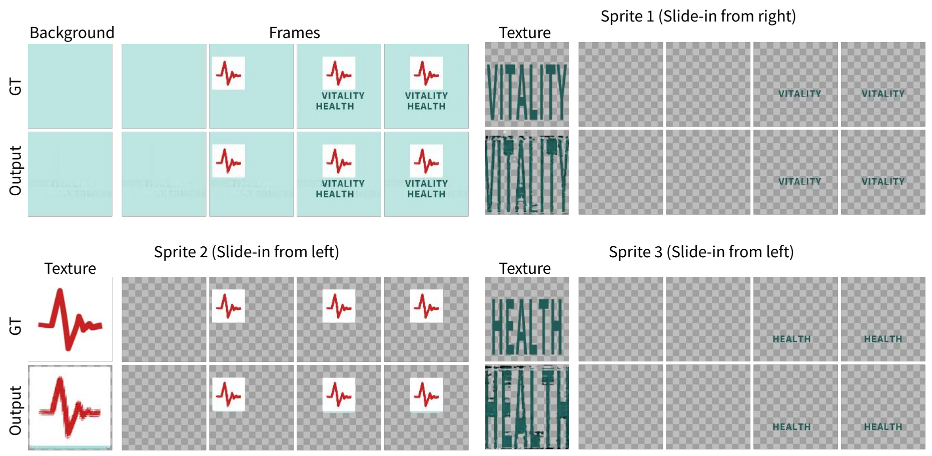
Background
Frames
Texture
Sprite 1 (Slide-in from right)
GT
Output
VITALITY
HEALTH
Texture
Sprite 2 (Slide-in from left)
Texture
Sprite 3 (Slide-in from left)

Fig. 13: Output example of our method. The top-left group shows the background texture and the reconstructed frame and the others show the foreground sprites. We put the description of the animation above each sprite. Best viewed with zoom and color.

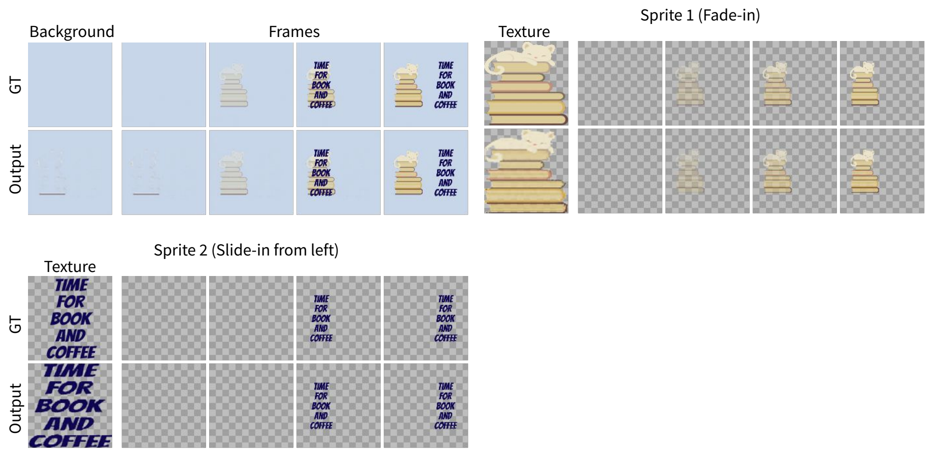

Fig. 14: Output example of our method. The top-left group shows the background texture and the reconstructed frame, and the others show the foreground sprites. We put the description of the animation above each sprite. Best viewed with zoom and color.

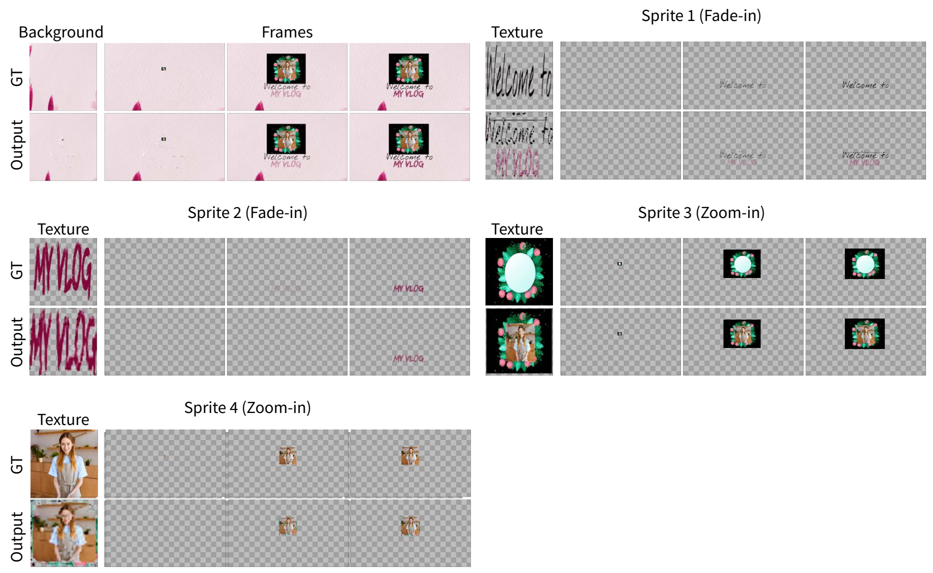

Fig. 15: A failure case of our method. Sprites with similar animations and close distances are difficult to decompose (as shown in Sprite 1 and Sprite 3). The top-left group shows the background texture and the reconstructed frame, and the others show the foreground sprites. We put the description of the animation above each sprite. Best viewed with zoom and color.

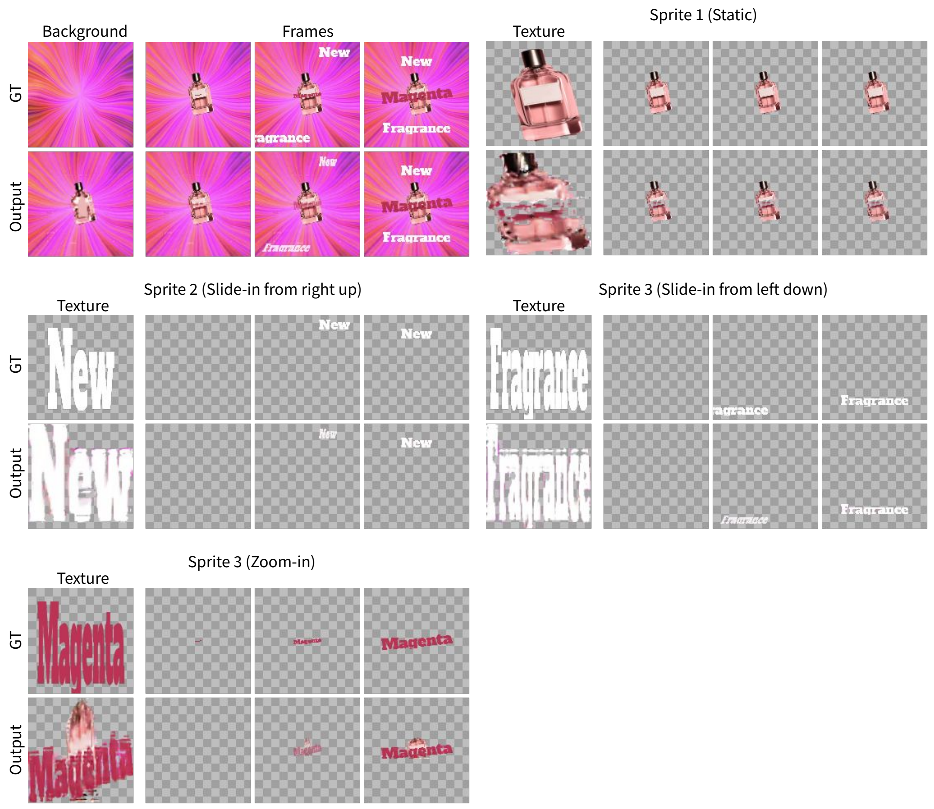

Fig. 16: A failure case of our method. Sprites with (almost) no animation tend to be absorbed into the background (as shown in Background and Sprite 1). The top-left group shows the background texture and the reconstructed frame and the others show the foreground sprites. We put the description of the animation above each sprite. Best viewed with zoom and color.