Physical Simulator In-the-Loop Video Generation (Supplementary Material)

Lin Geng Foo[1,5] Mark He Huang[2,3] Alexandros Lattas[4] Stylianos Moschoglou[4]
Thabo Beeler[4] Christian Theobalt[1,5]

[1]Max Planck Institute for Informatics, Saarland Informatics Campus [2]Singapore University of Technology and Design
[3]A*STAR [4]Google [5]Saarbrücken Research Center for Visual Computing, Interaction and Artificial Intelligence

1. More Method Details

1.1. More Details for Perception Pipeline

Object Scale Optimization After running the perception pipeline described in Section 3.2.1 of the main paper, we have predicted the camera poses and foreground object geometry and rough positions. However, there will sometimes still be a slight mismatch of object sizes of the simulator scene with respect to the template video. Because our final goal is to obtain accurate object motion guidance on the pixel space with the same viewing perspective of the template video, we need to determine accurate object sizes (as accurate as possible) with respect to the scene and align it properly with the camera. Therefore, we run an additional object scale optimization step. Specifically, we estimate the initial object sizes and poses from the 4D bundle adjusted depth maps and refine them via a render-and-align approach. Specifically, we refine the metric scale of each reconstructed object by matching rendered and observed 2D sizes in the first video frame. For each sample, we load camera intrinsics/extrinsics and object from our perception result. Given an initial object mesh and an approximate radius r from initial perception, we generate a set of candidate scales by normalizing the mesh to a cube of side length $s \in [0.5r, 3.5r]$ (in 30 linearly spaced steps). Each candidate is then placed at the estimated 3D centroid via a fixed rigid transform and rendered with Mitsuba [7]. From the RGBA output, we extract the object silhouette by thresholding the alpha channel and compute a tight 2D bounding box. We measure the agreement with each candidate via box IoU. We then fit a low-degree polynomial to the discrete IoU-vs-scale samples and take the maximizer $\hat{s}$ as the optimal side length. The mesh is finally re-normalized with side length $\hat{s}$, transformed, exported. We show an example in Fig. 1 and Fig. 2

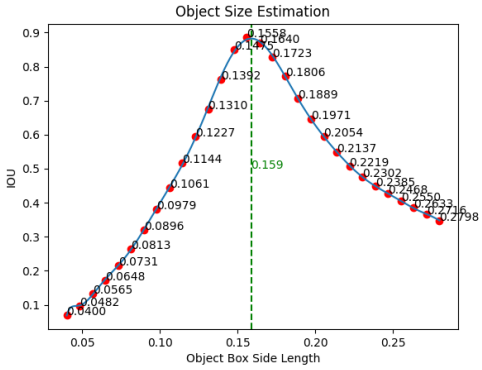

Figure 1. A sample of IoU curve used during our Object Scale Optimization step.

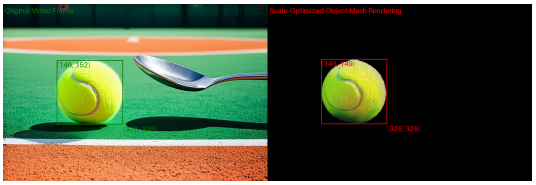

Figure 2. A sample of side-by-side image of the original RGB frame in the template video and our optimized rendering using generated object mesh, estimated camera pose, and refined object scale.

Scene Point Filtering. For the background filtering step discussed in Section 3.2.1 of main paper, we aim to remove unstable "floating" points that are weakly supported by the dominant scene geometry. Given a background point cloud $X \in \mathbb{R}^{N\times 3}$, we assign each point a randomized projection depth score in $[0, 1]$ by projecting X onto n_{dirs} random unit directions and counting in how many projections the point lies inside the central α-mass interval. We then sort these scores to focus on the lowest-scoring tail (at most a small fraction of points), and trim only if we detect a statistically significant gap in this tail (relative to typical score spacings and the discrete resolution $1/n_{\text{dirs}}$). This gap-aware rule removes small detached outlier clusters while leaving unimodal or smoothly varying background structures essentially unchanged.

Rotational Motion Computation. Given that recovering full 3D angular velocity from monocular, generated videos is highly ill-posed and correspondence quality can be degraded by appearance drift and geometric inconsistency. We

intentionally simplify the problem by estimating an object's initial spin from two-frame 2D correspondences (inferred by SuperGlue [22]) and then discretize the angular motion to a single dominant axis. Specifically, The rotational motion is isolated by computing a 2D flow field relative to the centroid of the matched feature points. From this field, we compute three independent scores that capture different modes of rotation. The Roll score corresponds to the mean normalized 2D curl, mean$((xf_y - yf_x)/|v_i|^2)$; the Pitch score is derived from the Pearson correlation between vertical position and vertical flow, corr(y, f_y); and the Yaw score is obtained from the correlation between horizontal position and horizontal flow, corr(x, f_x). The dominant rotational axis is determined by the score with the largest absolute magnitude, thereby distinguishing between swirling (Roll) and divergent (Pitch or Yaw) motion patterns.

1.2. More Details for Physical Simulation

In Section 3.2.2 of the main paper, we present our procedure for prompting a large VLM (GPT-5) for physical property estimation, here we provide more details on the prompting pipeline and parameter mapping. To obtain physical parameters for initializing our MPM-based simulator in the loop, we adopt a two-stage procedure that combines prompting the VLM to reason about the object's physical property with an offline, deterministic mapping from qualitative attributes to numerical values. This two-stage procedure ensures robustness for the pipeline to be fail-safe.

Stage 1: Physical attribute extraction. Given the first frame of the template video, we highlight each object of interest by drawing an bounding box on the frame. We prompt VLM with the image and a curated textual instruction (provided below) to elicit discrete, qualitative descriptors of the object's physical behavior and material. The VLM is instructed to describe the object using only a fixed set of categorical labels, and to avoid outputting any numerical quantities.

Listing 1. Prompt used for acquiring physical descriptor

```
You are a physics-aware vision assistant. You
    are given an image and a highlighted object.
    Your task is to describe the objects
    material and contact behavior using only
    qualitative categories and discrete labels.
Note that:
1. Do not output any numeric physical values
    such as density, Youngs modulus, coefficient
    of restitution, or friction coefficients.
2. Always use the fixed label sets and scales
    provided in the user prompt.
3. Return your answer strictly as a single JSON
    object without additional commentary.

You are given an image that contains a single or
    multiple objects.
The object of interest is the one that is
    highlighted by a bounding box in the image.

Based on visual appearance and common real-world
    knowledge,
estimate the following qualitative physical
    attributes of the object of interest.

1. Material category (choose ONE):
  - 'metal'
  - 'wood'
  - 'hard_plastic'
  - 'soft_plastic_or_rubber'
  - 'glass_or_ceramic'
  - 'fabric_or_textile'
  - 'paper_or_cardboard'
  - 'foam'
  - 'stone_or_concrete'
  - 'other'

2. Solidity / internal structure (choose ONE):
  - 'solid' (mostly solid, compact interior)
  - 'mostly_solid' (small hollow parts but
      mostly solid)
  - 'hollow_thin_shell' (like a thin plastic
      ball, can)
  - 'layered_or_composite' (e.g., book, stacked
      layers)

3. Hardness (choose ONE):
  - 'very_soft' (easily deformable by hand, like
      sponge, pillow)
  - 'soft'
  - 'medium'
  - 'hard'
  - 'very_hard' (rigid, difficult to deform,
      like steel, stone, thick glass)

4. Elasticity / bounce behavior in everyday use:
  Imagine the object is dropped from about 1
      meter onto a hard floor.
  Choose ONE:
  - 'almost_no_bounce' (stays where it lands,
      heavy or very soft)
  - 'low_bounce'  (barely bounces, like a wood
      block)
  - 'medium_bounce' (bounces but quickly loses
      height, like some plastics)
  - 'high_bounce'  (bounces well, like a rubber
      ball)
  - 'very_high_bounce' (extremely lively, like a
      superball)

5. Surface roughness (integer 1-5):
  - 1 = very smooth / polished (glass, glossy
      plastic)
  - 2 = mostly smooth (painted metal, smooth
      wood)
  - 3 = slightly textured (matte plastic,
      unfinished wood)
  - 4 = rough (coarse wood, textured rubber)
  - 5 = very rough (abrasive, heavily textured)

6. Surface friction tendency (choose ONE):
  - 'very_slippery'
  - 'slippery'
  - 'medium'
  - 'grippy'
  - 'very_grippy'
```

```
7. Thickness / size hint (choose ONE):
  This is to help estimate stiffness at the
      scale of the object.
  - 'thin_and_small' (e.g., thin plastic cup,
      small toy)
  - 'thin_and_large' (e.g., large but thin panel)
  - 'thick_and_small' (compact, chunky object)
  - 'thick_and_large' (large and bulky)

8. Short natural-language justification:
  Briefly explain how the texture, appearance,
      and context of the object
  led you to these choices.

OUTPUT FORMAT:
Return ONLY a single JSON object with the
    following keys:
- 'material_class'
- 'solidity'
- 'hardness_level'
- 'bounce_category'
- 'surface_roughness_level' (integer 1-5)
- 'friction_tendency'
- 'size_thickness_hint'
- 'justification'

Example of the required structure (values are
    just illustrative):
{
 'material_class': 'hard_plastic',
 'solidity': 'hollow_thin_shell',
 'hardness_level': 'hard',
 'bounce_category': 'medium_bounce',
 'surface_roughness_level': 2,
 'friction_tendency': 'medium',
 'size_thickness_hint': 'thin_and_small',
 'justification': 'The object looks like a
     hollow plastic toy ball with a smooth,
     glossy surface.'
}
```

We refer to this physical descriptor as

$$s = f_\phi(I)$$

where I is the input frame and f_ϕ denotes the VLM, and s contains the categorical attributes listed above.

Stage 2: Offline mapping to numerical physical parameters. In the second stage, we deterministically map the semantic descriptor s to the numerical parameters used by the physics simulator. This mapping is implemented as a set of lookup tables and multiplicative factors, yielding:

$$p = g(s),$$

where p comprises density ρ, Young's modulus E, coefficient of restitution e, and static / dynamic friction coefficients $(\mu_\text{s}, \mu_\text{d})$:

$$p = (\rho, E, e, \mu_\text{s}, \mu_\text{d}).$$

We first assign a base density ρ_base to each material class (e.g., metals, wood, plastics, foam), chosen to be order-of-magnitude realistic for bulk materials. We then modulate this value using a solidity-dependent factor (e.g., lower effective density for hollow objects):

$$\rho = \alpha_\text{solidity}(\texttt{solidity}) \cdot \rho_\text{base}(\texttt{material\_class}).$$

Similarly, we define a base Young's modulus E_base for each combination of material class and hardness level, capturing the typical stiffness range for that material and hardness category. We will provide the full prompting and mapping script along with our source code.

Discussion. This design separates high-level perception from low-level physics parameterization. The VLM is used only to produce stable, interpretable semantic attributes s, which tend to be much more robust than direct regression of physical scalars from pixels. The final physical parameters p used in simulation are obtained by the deterministic mapping g, which is under full user control. This separation yields more consistent and physically plausible parameters across scenes, while still leveraging the VLM's ability to recognize materials and contact behavior from visual cues.

More Details of Simulation State Initialization After obtaining all scene assets, determined simulation domain, estimated physical properties as introduced in Section 3.2.2 of the main paper, we initialize the simulation state in the simulator. Here, we discuss more details of this initialization. Specifically, we import the background scene geometry as a signed distance function and evaluate a 3D plane model as ground plane inside the simulation domain to serve as efficient colliders. Then, we import object meshes as Material Points using mesh vertices and physical properties estimated from previous steps. We also voxelize the object meshes to infill the internal space with uniform Material Points to prevent object collapse. Lastly, using the estimated object linear velocity and rotations, we compute particle-level initial velocity for all material points in the scene and assign them accordingly. We set constant gravitational pull and air drag to the simulation domain to finish the state initialization.

1.3. More Details for TTCO

In Section 3.3.1 of the main paper, we introduced our Test-time Texture-consistency Optimization (TTCO) technique. Here, we discuss more details regarding how we added learnable parameters to the prompt embeddings and text features.

Specifically, for the input T5 [19] text embeddings, we first locate the text embeddings corresponding to the foreground objects. Then, we add a learnable residual to these text embeddings, which is zero-initialized. Therefore, in the first iteration of training, the model and text prompts are exactly the same as their pre-trained versions, and the model learns the modulations such that the foreground objects follow the physical simulator movements more closely. Additionally, we also add similar zero-initialized learnable modulations at each intermediate block of the diffusion transformer (DiT), corresponding to the foreground text embed-

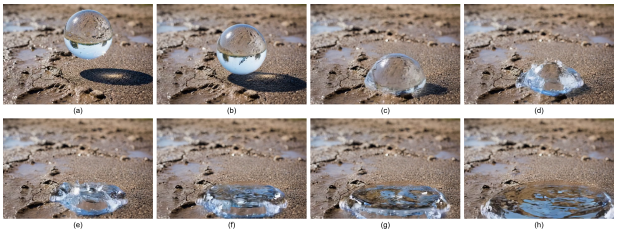

Figure 3. Visualization of our method generating videos of fluid movements.

dings. Such layer-wise modulations better aid the model's convergence towards fitting the movements.

More specifically, let $E = [e_1, \ldots, e_N]$ denote the T5 text embeddings of the input prompt, and let $\mathcal{F} \subseteq \{f_1, \ldots, f_L\}$ be the set of indices corresponding to the foreground object tokens, where there are L foreground object tokens. We introduce learnable residuals r_i for $i \in \mathcal{F}$ (zero-initialized), and define the modulated text embeddings as:

$$\tilde{e}_i = \begin{cases} e_i + r_i, & i \in \mathcal{F}, \\ e_i, & \text{otherwise.} \end{cases} \tag{1}$$

Furthermore, for each DiT block $k = 1, \ldots, K$, we introduce independent foreground-modulation residuals $r_i^{(k)}$ (also zero-initialized) at each layer, such that intermediate text features $x_i^{(k)}$ at the k-th layer and i-th position are modulated to $\tilde{x}_i^{(k)}$ as follows:

$$\tilde{x}_i^{(k)} = \begin{cases} x_i^{(k)} + r_i^{(k)}, & i \in \mathcal{F}, \\ x_i^{(k)}, & \text{otherwise.} \end{cases} \tag{2}$$

2. More Experiment Results

2.1. More Qualitative Results

We show some qualitative results in Figure 5 of the main paper. We show more qualitative results here from Figure 4 to 7. For the single object examples shown, the prompt is to let the various objects drop from mid-air to the ground. For the shown video samples with multiple objects, the prompt instructs the two objects to collide into each other. As observed from the qualitative results, our method is superior in terms of following physical laws and maintaining consistency of textures.

Handling of Complex Materials. Our method can also handle more complex materials such as fluids and sand, by setting the appropriate physical properties (e.g., Young's modulus) in our perception pipeline, and allowing MPM to simulate them. We visualize an example in Fig. 3, where we simulate and generate fluid movements from an input text prompt. This demonstrates the versatility of the simulator-generator integration paradigm in PSIVG.

2.2. Generating Videos of Dropping Objects

We also run experiments on PisaBench dataset [11], which contains videos of objects dropping from mid-air with various backgrounds, often colliding into other objects. This experiment evaluates if we are able to generate videos that closely follow physical principles of gravity and collisions.

Evaluation Metrics. We follow the PisaBench [11] protocol and report three quantitative metrics to assess the physical accuracy of generated videos. *Trajectory L2:* We compute the centroid of the object mask in each frame for both the generated and ground-truth videos, and measure the average L2 distance between corresponding centroids across all frames. This reflects how closely the generated motion follows the real trajectory. *Chamfer Distance (CD):* To evaluate shape fidelity, we calculate the Chamfer Distance between the object masks of the generated and ground-truth frames, capturing discrepancies in object shape and spatial alignment. *Intersection over Union (IoU):* Finally, we report the IoU between the generated and ground-truth object masks, which measures the degree of spatial overlap and indicates the accuracy of object permanence and localization over time.

Baselines. For PisaBench, we follow their paper and report results on four open models: CogVideoX [31] DynamiCrafter [28], Pyramid-Flow [8], and Open-Sora-V1.2 [35], as well as 4 proprietary models: Sora [16], Kling-V1 [10], KlingV1.5 [10], and Runway Gen3 [21]. We also report results on PISA's trained models [11] with Object Reward Optimization (ORO) using Seg, Flow and Depth modalities. Additionally, we also run some controllable video generation methods: MotionClone [13], SG-I2V [15], DragAnything [26], Image Conductor [12], which we implement by applying the simulator's conditioning information.

Results. Results on PisaBench are reported in Tab. 1, where our method achieves the best performance across all three physical accuracy metrics. These results demonstrate that PSIVG generates object motions that more faithfully follow physically plausible trajectories and maintain consistent object shapes and spatial positions throughout the video.

3. More Implementation Details

In Section 4 of the main paper, we discussed the implementation details. Here, we discuss more implementation details.

To compute the optical flow used as input for the flow-conditioned video generation model (GwtF [1]), we derive the foreground flow from the simulator-rendered RGB frames and the background flow from the template video (as discussed in Section 3.3 of the main paper). For the foreground, we use simulator-generated segmentation masks to crop the moving object and composite it onto the static background of the first frame of the template video, creating a video of the simulated foreground over a realistic static

Table 1. Quantitative comparison on PisaBench [11]. Results are reported on the *sim* (unseen) setting.

	Method	L2 ↓	CD ↓	IoU ↑
Proprietary	[16] Sora	0.140	0.419	0.031
	[10] Kling-V1	0.145	0.437	0.028
	[10] Kling-V1.5	0.132	0.405	0.029
	[21] Runway Gen3	0.149	0.460	0.038
Open-Source T2V	[31] CogVideoX	0.101	0.290	0.020
	[28] DynamiCrafter	0.136	0.430	0.033
	[8] Pyramid-Flow	0.130	0.381	0.048
	[35] Open-Sora	0.130	0.368	0.034
	[11] PISA-Seg	0.032	0.063	0.145
	[11] PISA-Flow	0.022	0.045	0.071
	[11] PISA-Depth	0.022	0.046	0.096
Controllable	[13] MotionClone	0.085	0.246	0.063
	[15] SG-I2V	0.113	0.344	0.034
	[26] DragAnything	0.176	0.523	0.033
	[12] Image Conductor-Object	0.148	0.405	0.033
	Ours (PSIVG)	**0.012**	**0.018**	**0.397**

scene. This is because the simulator backgrounds tend to produce inaccurate RAFT flow, likely due to being out of distribution for the RAFT model [24]. For the background, we directly compute RAFT optical flow on the template video and obtain background regions using a segmentation model. To generate segmentation masks for the template video, we apply GroundingDINO [14] with the foreground prompts and propagate the masks across frames using SAM 2 [20]. After obtaining the background RAFT optical flow, we further apply a threshold on the background flow to remove the very large "flows", which are often the result of abrupt jumping of objects or flickering in the template video. Finally, the foreground and background flows are fused using the segmentation masks to form a hybrid flow, which serves as the input to GwtF.

We make several modifications to adapt our text-to-video generation pipeline for the PisaBench dataset [11] in Section 2.2. One key challenge is that the resolution of the input image (1024×1024) is significantly larger than the frame resolution of GwtF (480×720). Yet, simply downscaling the input image and padding the sides causes objects to become excessively small, leading to degraded performance. This issue is common among video latent diffusion models, which apply heavy spatial and temporal downsampling in the encoder (e.g., CogVideoX [31] and GwtF [1] use $32\times$ spatial and $4\times$ temporal downsampling). As a result, small objects such as shoes or tape which are often only 20-30 pixels wide originally, can vanish entirely from the generated video after downsizing and encoding. To address this, we spatially and temporally crop each video into multiple smaller clips and generate them separately. We also perform augmentation of the prompts using a VLM to describe the scene better. We adapt the text prompts based on the objects visible in each clip, to avoid hallucinations when certain objects mentioned in the prompt are absent from the frame. Additionally, we design a specific prompt used after collisions occur, to prevent objects from unnaturally colliding again when the model tries to performing the "dropping action" again. Finally, because the PisaBench simulator backgrounds are synthetic and fall outside the training distribution of typical video models, we find that GwtF often hallucinates or distorts these background regions. To mitigate this, we introduce a background loss during PisaBench experiments to enforce consistency with the initial frame, ensuring stable and realistic background appearance throughout the video.

4. More Experiment Details

4.1. More Details for Text-to-Video Generation Experiments

In Section 5.1 we discussed our text-to-video generation results from various prompts. Here, we provide more details on evaluation metrics.

Evaluation Metrics. We assess our method from two perspectives: **motion controllability** and general **video generation quality**. Motion Controllability: To quantify how well object motions adhere to the simulated trajectories, we evaluate: (1) the *mean Intersection-over-Union (SAM mIoU)* between SAM2 [20] tracking boxes in the generated videos and those from the physical simulator renders, and (2) the *pixel-level mean squared error (Corr. Pixel MSE)* between the pixels of consecutive frames in the generated videos according to the frame-to-frame pixel correspondences, measuring how well the generated frames follow the simulated trajectory. These metrics together measure how faithfully the generated objects follow simulated dynamics in both translation and rotation. General Video Generation Quality: We further assess the overall realism, text alignment, and temporal stability of generated videos. For text alignment, we compute the cosine similarity between the CLIP [18] text embedding of the prompt and the CLIP image embeddings of generated frames, averaged over time (*CLIP Text*). For temporal consistency, we evaluate two sets of measures: (1) the mean cosine similarity between CLIP image embeddings of consecutive frames (*CLIP Img*), and (2) four temporal quality metrics from VBench [6], including *subject consistency*, *background consistency*, *motion smoothness*, and *temporal flickering*.

4.2. More Details on User Study

In Section 5.2 of the main paper, we conducted a user study involving 32 participants to assess the physical consistency of our generated videos, and here we provide more details. Specifically the user study consisted of 25 multiple choice questions, where each question featured our generated video along with 5 strong baselines (CogVideoX [30], HunyuanVideo [9], PISA-Seg [11], SG-I2V [15], MotionClone [13]). These 5 baselines were selected because they

generally showed the best quantitative results in Table 1 of the main paper. All 6 videos were put side-by-side in GIF format, and in a random order for each question. For each multiple choice question, the respondents were asked to answer the following question: "Which of these videos is the most physically accurate?". For certain videos with camera movements, the question included more information, and becomes instead: "These videos are supposed to have camera movements. With this in mind, which of these videos is the most physically accurate?" Overall, as shown in Table 2 of the main paper, our method was preferred in 82.3% of the comparisons across all 6 options, substantially outperforming all baseline models. This provides strong evidence that our generated videos were significantly more physically consistent and natural.

5. Limitations and Future Work

Despite its strengths, our method has several limitations. First, its applicability is bounded by the capabilities of the underlying physical simulator, the Material Point Method (MPM) [23]. As a result, we inherit known challenges in modeling complex agents (e.g., humans and vehicles) and articulated objects (e.g., laptops), since MPM typically requires careful particle-level material initialization and parameterization to represent such structures faithfully. Additionally, our framework struggles with extremely small or thin objects – such as cloth or paper – that involve fine-grained dynamics. This stems largely from limitations within the GwtF video generation model [1], which often fails to accurately synthesize the movements and structural details of small or thin objects.

Several promising avenues exist for extending our work. First, future research could explore integrating richer conditioning signals from the simulator, such as 3D point tracks, to better align generated videos with simulated trajectories [4, 25]. Second, the reconstruction and perception pipeline could be strengthened by incorporating more recent image-to-3D [2, 27] or video-based reconstruction methods [5], which might particularly improve performance in scenarios involving initial-frame occlusions [2]. In an orthogonal direction, leveraging text-prompting techniques for physical consistency [29], other prompt optimization techniques [17, 32], or agentic AI frameworks [34] could enhance video diffusion prompts in a training-free manner. Finally, our proposed TTCO technique also holds potential as a general-purpose test-time refinement tool for other fields such as controllable video editing [3, 33], where it could significantly boost visual quality based on user-provided inputs and computed tracks.

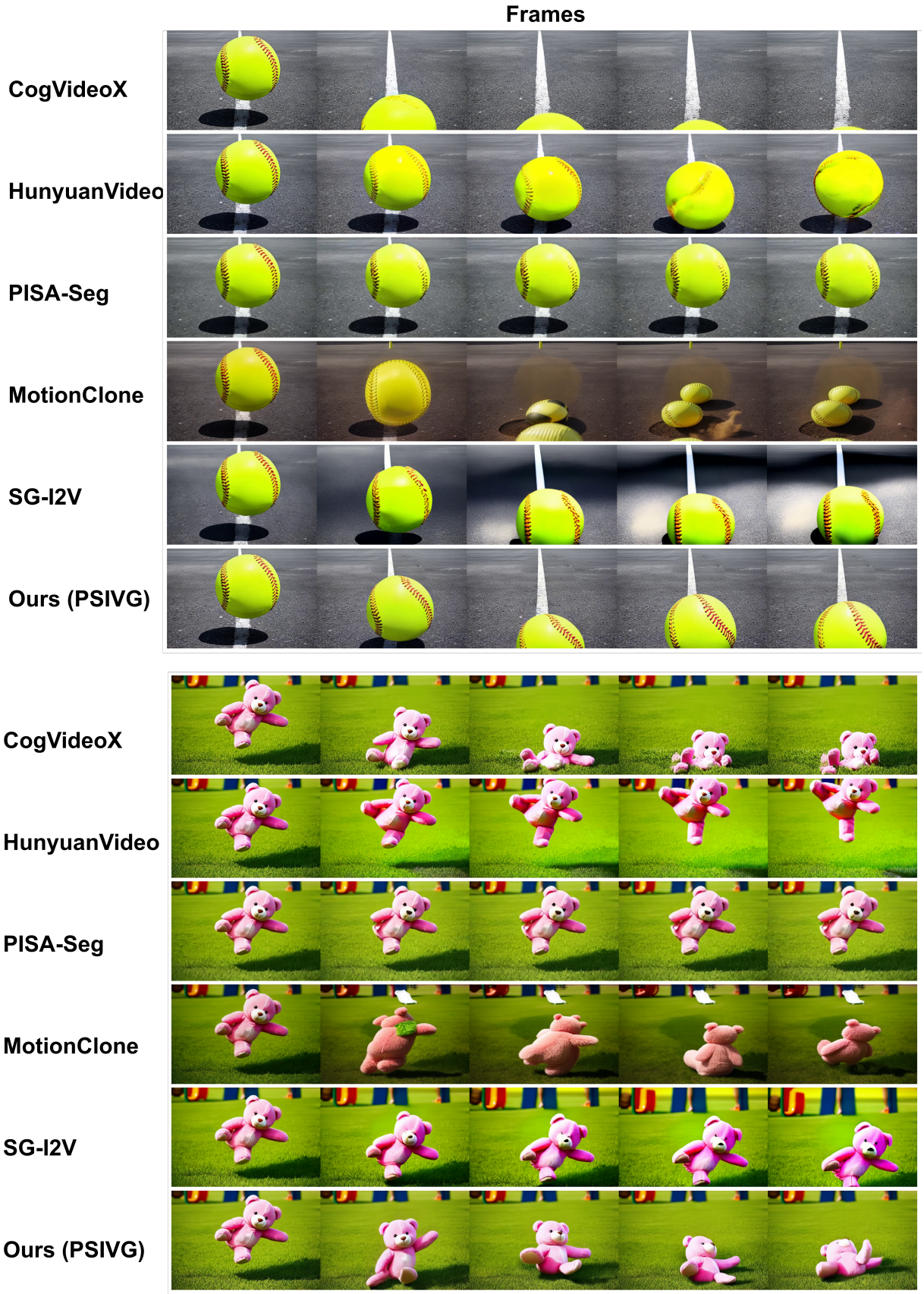

Figure 4. More qualitative comparisons with baselines.

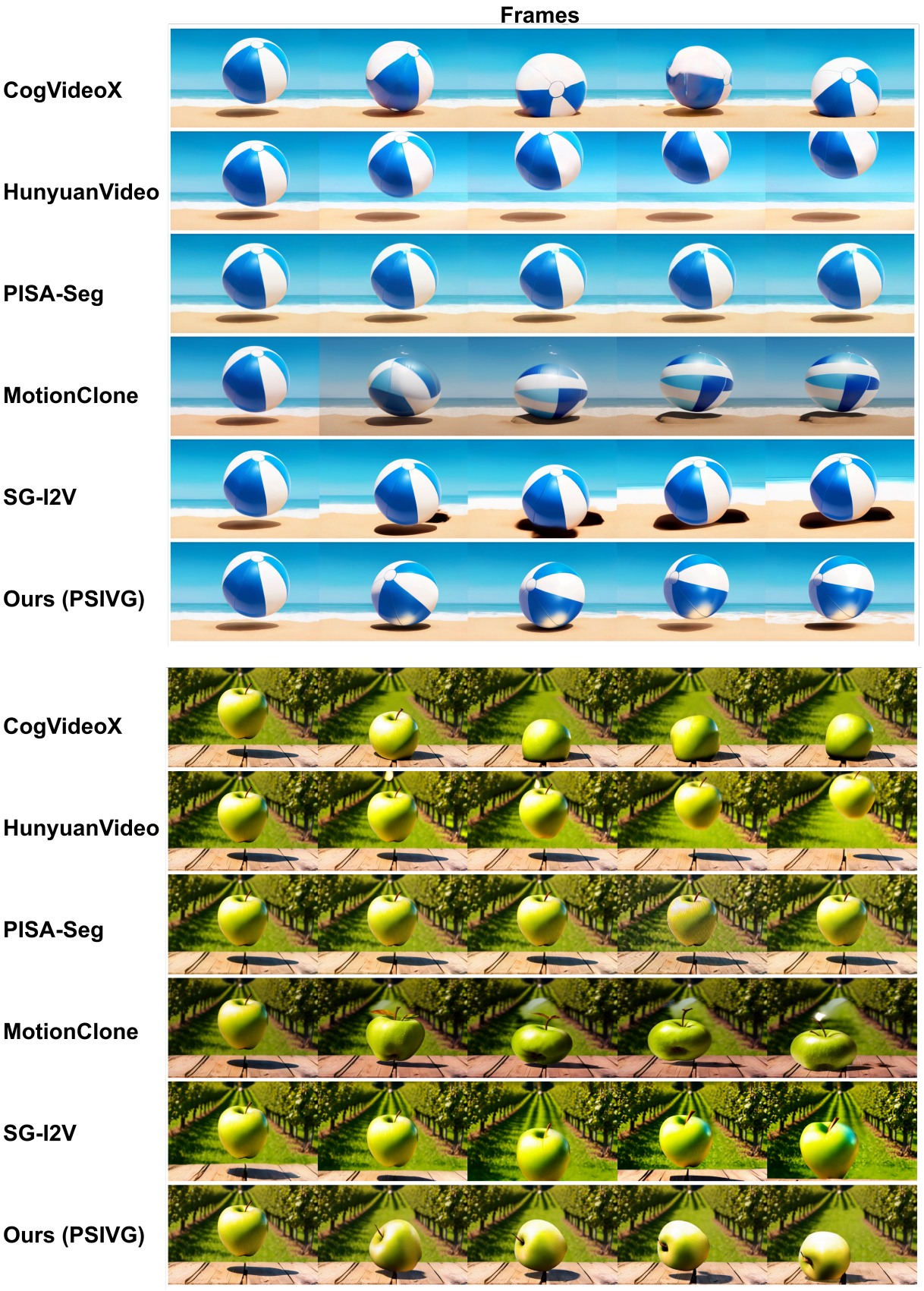

Figure 5. More qualitative comparisons with baselines.

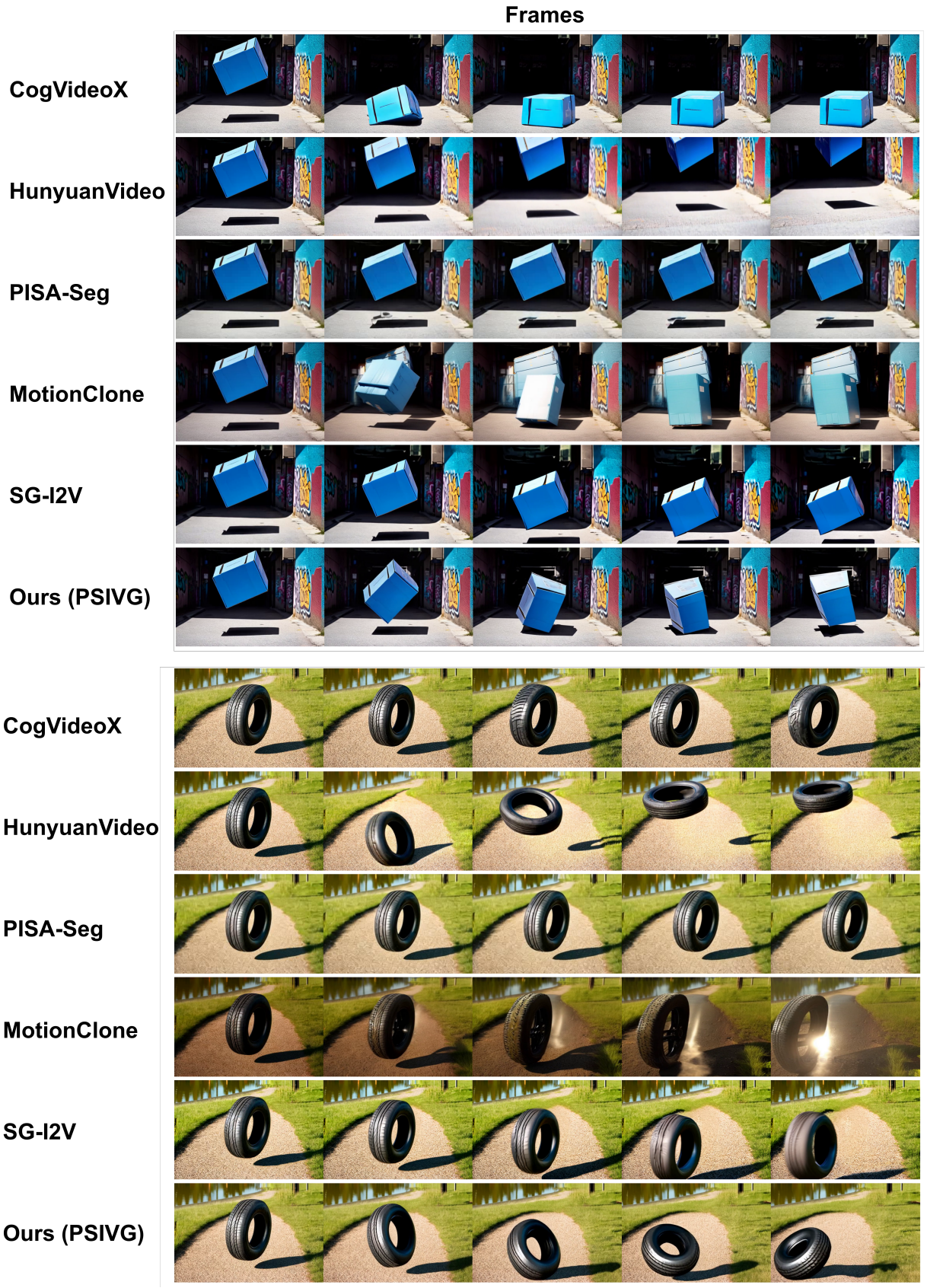

Figure 6. More qualitative comparisons with baselines.

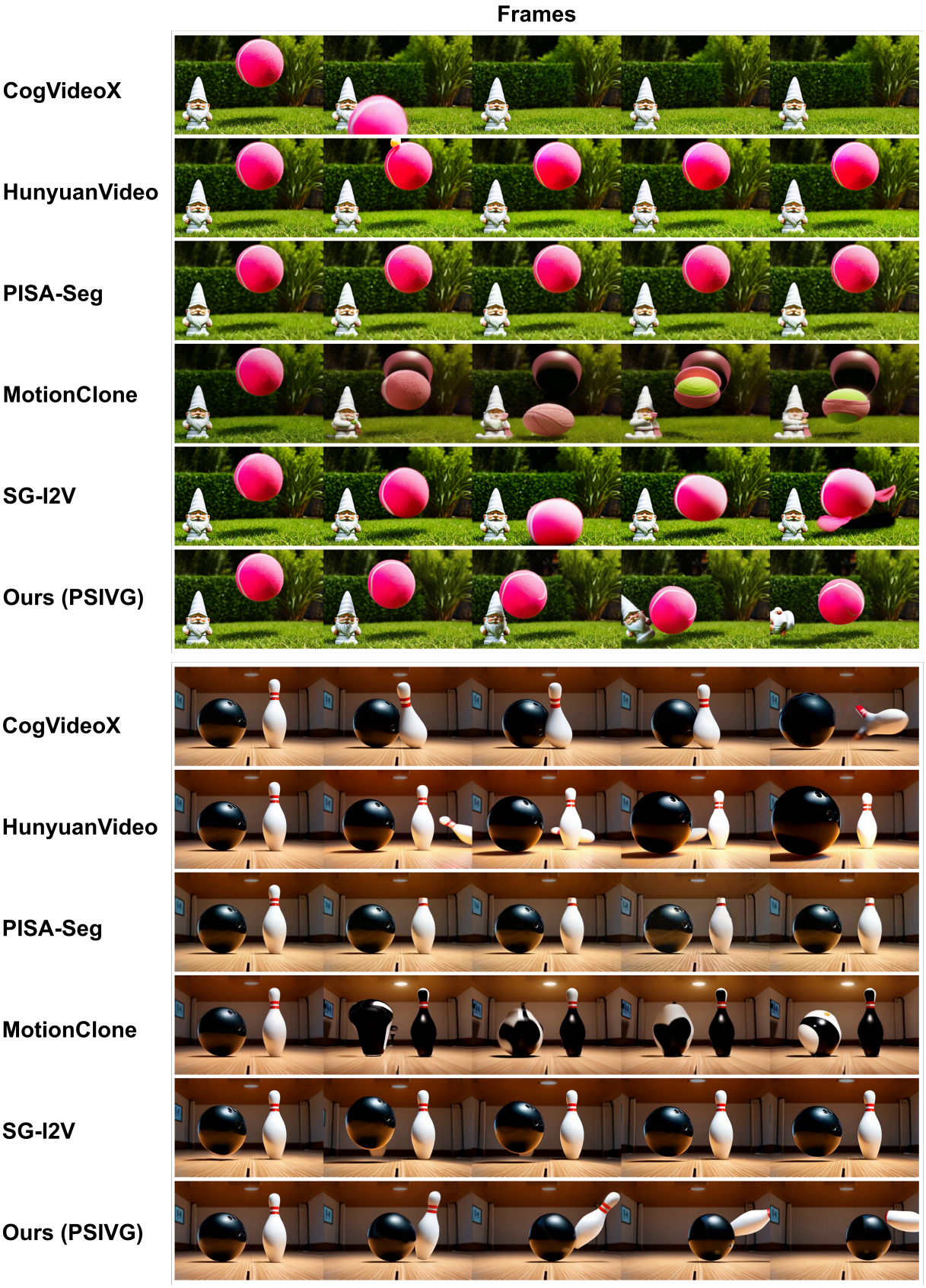

Figure 7. More qualitative comparisons with baselines.

References

[1] Ryan Burgert, Yuancheng Xu, Wenqi Xian, Oliver Pilarski, Pascal Clausen, Mingming He, Li Ma, Yitong Deng, Lingxiao Li, Mohsen Mousavi, et al. Go-with-the-flow: Motion-controllable video diffusion models using real-time warped noise. In Proceedings of the Computer Vision and Pattern Recognition Conference, pages 13–23, 2025.

[2] Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, et al. Sam 3d: 3dfy anything in images. arXiv preprint arXiv:2511.16624, 2025.

[3] Xiang Fan, Anand Bhattad, and Ranjay Krishna. Videoshop: Localized semantic video editing with noise-extrapolated diffusion inversion. In European conference on computer vision, pages 232–250. Springer, 2024.

[4] Zekai Gu, Rui Yan, Jiahao Lu, Peng Li, Zhiyang Dou, Chenyang Si, Zhen Dong, Qifeng Liu, Cheng Lin, Ziwei Liu, et al. Diffusion as shader: 3d-aware video diffusion for versatile video generation control. In Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers, pages 1–12, 2025.

[5] Mark He Huang, Lin Geng Foo, Christian Theobalt, Ying Sun, and De Wen Soh. Onlinesplatter: Pose-free online 3d reconstruction for free-moving objects. In The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2025.

[6] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 21807–21818, 2024.

[7] Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. Dr. jit: A just-in-time compiler for differentiable rendering. ACM Transactions on Graphics (TOG), 41(4): 1–19, 2022.

[8] Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong MU, and Zhouchen Lin. Pyramidal flow matching for efficient video generative modeling. In The Thirteenth International Conference on Learning Representations, 2025.

[9] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. arXiv preprint arXiv:2412.03603, 2024.

[10] Kuaishou. Kling. https://kling.kuaishou.com/, 2025. Accessed: 2025-10-15.

[11] Chenyu Li, Oscar Michel, Xichen Pan, Sainan Liu, Mike Roberts, and Saining Xie. Pisa experiments: Exploring physics post-training for video diffusion models by watching stuff drop. In Forty-second International Conference on Machine Learning, 2025.

[12] Yaowei Li, Xintao Wang, Zhaoyang Zhang, Zhouxia Wang, Ziyang Yuan, Liangbin Xie, Ying Shan, and Yuexian Zou. Image conductor: Precision control for interactive video synthesis. In Proceedings of the AAAI Conference on Artificial Intelligence, pages 5031–5038, 2025.

[13] Pengyang Ling, Jiazi Bu, Pan Zhang, Xiaoyi Dong, Yuhang Zang, Tong Wu, Huaian Chen, Jiaqi Wang, and Yi Jin. Motionclone: Training-free motion cloning for controllable video generation. In The Thirteenth International Conference on Learning Representations, 2025.

[14] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In European conference on computer vision, pages 38–55. Springer, 2024.

[15] Koichi Namekata, Sherwin Bahmani, Ziyi Wu, Yash Kant, Igor Gilitschenski, and David B. Lindell. SG-i2v: Self-guided trajectory control in image-to-video generation. In The Thirteenth International Conference on Learning Representations, 2025.

[16] OpenAI. Creating video from text. https://openai.com/index/sora//, 2025. Accessed: 2025-10-15.

[17] Duo Peng, Zhengbo Zhang, Ping Hu, Qiuhong Ke, David KY Yau, and Jun Liu. Harnessing text-to-image diffusion models for category-agnostic pose estimation. In European Conference on Computer Vision, pages 342–360. Springer, 2024.

[18] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In International conference on machine learning, pages 8748–8763. PmLR, 2021.

[19] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of machine learning research, 21(140):1–67, 2020.

[20] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollar, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In The Thirteenth International Conference on Learning Representations, 2025.

[21] Runway. Introducing gen-3 alpha: A new frontier for video generation. https://runwayml.com/research/introducing-gen-3-alpha/, 2025. Accessed: 2025-10-15.

[22] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 4938–4947, 2020.

[23] Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. A material point method for snow simulation. ACM Transactions on Graphics (TOG), 32(4): 1–10, 2013.

[24] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field

transforms for optical flow. In European conference on computer vision, pages 402–419. Springer, 2020.

[25] Chen Wang, Chuhao Chen, Yiming Huang, Zhiyang Dou, Yuan Liu, Jiatao Gu, and Lingjie Liu. Physctrl: Generative physics for controllable and physics-grounded video generation. In The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2025.

[26] Weijia Wu, Zhuang Li, Yuchao Gu, Rui Zhao, Yefei He, David Junhao Zhang, Mike Zheng Shou, Yan Li, Tingting Gao, and Di Zhang. Draganything: Motion control for anything using entity representation. In European Conference on Computer Vision, pages 331–348. Springer, 2024.

[27] Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, and Jiaolong Yang. Native and compact structured latents for 3d generation. Tech report, 2025.

[28] Jinbo Xing, Menghan Xia, Yong Zhang, Haoxin Chen, Wangbo Yu, Hanyuan Liu, Gongye Liu, Xintao Wang, Ying Shan, and Tien-Tsin Wong. Dynamicrafter: Animating open-domain images with video diffusion priors. In European Conference on Computer Vision, pages 399–417. Springer, 2024.

[29] Qiyao Xue, Xiangyu Yin, Boyuan Yang, and Wei Gao. Phyt2v: Llm-guided iterative self-refinement for physics-grounded text-to-video generation. In Proceedings of the Computer Vision and Pattern Recognition Conference, pages 18826–18836, 2025.

[30] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. arXiv preprint arXiv:2408.06072, 2024.

[31] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Yuxuan.Zhang, Weihan Wang, Yean Cheng, Bin Xu, Xiaotao Gu, Yuxiao Dong, and Jie Tang. Cogvideox: Text-to-video diffusion models with an expert transformer. In The Thirteenth International Conference on Learning Representations, 2025.

[32] Zhengbo Zhang, Li Xu, Duo Peng, Hossein Rahmani, and Jun Liu. Diff-tracker: text-to-image diffusion models are unsupervised trackers. In European Conference on Computer Vision, pages 319–337. Springer, 2024.

[33] Zhengbo Zhang, Yuxi Zhou, Duo Peng, Joo-Hwee Lim, Zhigang Tu, De Wen Soh, and Lin Geng Foo. Visual prompting for one-shot controllable video editing without inversion. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7784–7794, 2025.

[34] Bingxi Zhao, Lin Geng Foo, Ping Hu, Christian Theobalt, Hossein Rahmani, and Jun Liu. Llm-based agentic reasoning frameworks: A survey from methods to scenarios. arXiv preprint arXiv:2508.17692, 2025.

[35] Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all. arXiv preprint arXiv:2412.20404, 2024.