InterDyn: Controllable Interactive Dynamics with Video Diffusion Models
Supplementary Material

A. Ablation study

We use binary hand masks as our control signal due to their widespread availability via methods such as GroundingDINO [52] and SAM2 [65]. However, other control signals—such as skeletons or meshes—might provide richer controllability, since they encode pseudo-3D information and fine-grained correspondences across frames.

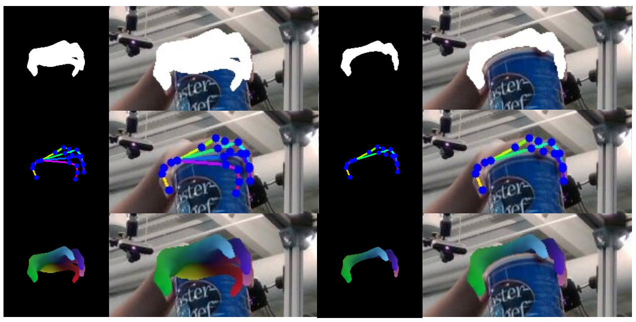

Figure S1. **Evaluated control signals.** From top to bottom: binary mask, joints in the style of OpenPose [9], and colored mesh [61]. Left: w/o object occlusions. Right: w/ object occlusions.

We evaluate the impact of our control signal on performance using the DexYCB dataset [11]. It provides 8,000 videos of hands grasping an object, along with ground-truth 3D hand/object poses/meshes. DexYCB uses the parametric human hand model MANO [67], rendered here as (i) a binary mask (similar to our SSV2 control signal), (ii) joints similar to OpenPose [9], and (iii) a colormap based on [61], see Fig. S1. Additionally, when generating hand masks for SSV2 with SAM2, hand-held objects provide an object contour in the hand mask, inadvertently providing InterDyn with information on the trajectory and shape of the object (a limitation we share with our baseline CosHand [70]). To evaluate its impact, we train separate InterDyn versions on DexYCB, where we render the control signal either with or without the contour of hand-held objects, see Fig. S1.

We present the ablation results in Tab. S1, which indicate that both the type of control signal and contour-leaking effect have minimal impact on image quality, spatio-temporal dynamics, and motion fidelity. These findings softly hint that maintaining hand consistency and driving object interactions does not heavily depend on detailed control signals, rather it does on the video generation model's implicit understanding of interactive dynamics. This is great news, highlighting the potential of using simple, readily available control signals to generate high-quality video outputs.

B. State comparison

For completeness, we also compare against CosHand [70] for the second frame of each video and compare these results with the baselines reported in [70], see Tab. S2. For MCVD, UCG, IPix2Pix, TCG, and CosHand (the first five rows), we adopt the results reported in [70] without retraining. Since CosHand does not provide, nor specify, an exact validation split, these numbers are not directly comparable. We also run CosHand on our own validation set.

Method	SSIM ↑	PSNR ↑	LPIPS ↓
MCVD [70, 75]	0.231	8.75	0.307
UCG [66, 70]	0.340	12.08	0.124
IPix2Pix [7, 70]	0.289	9.53	0.296
TCG [66, 70]	0.234	9.05	0.221
CosHand [70]	0.414	13.72	**0.116**
CosHand [70] (our val-set)	0.698	20.55	0.194
Ours (256×256)	0.785	23.93	0.127
Ours (256×384)	**0.796**	**24.37**	0.122

Table S2. **State comparison on the SSV2 dataset.** We compare InterDyn with CosHand and other static baseline methods for generating a single future frame. We report results for InterDyn at two resolutions: 256×256 (matching CosHand) and 256×384 (matching SVD's prior).

Control	Occlusion	SSIM ↑		PSNR ↑		LPIPS ↓		FVD ↓		Motion Fidelity ↑ [93]	
		256×256	256×384	256×256	256×384	256×256	256×384	256×256	256×384	256×256	256×384
Mask	✗	**0.829**	0.847	24.08	24.75	0.123	0.121	39.94	41.99	0.666	0.670
Joints	✗	0.827	0.846	24.00	24.72	0.124	0.122	40.02	41.17	0.673	0.676
Mesh	✗	0.828	0.847	24.14	24.83	0.122	0.121	41.99	42.26	0.663	0.665
Mask	✓	**0.829**	0.847	**24.15**	24.79	0.122	0.121	**37.64**	41.18	**0.675**	0.672
Joints	✓	0.827	0.846	24.05	24.69	0.124	0.122	44.07	41.41	0.665	0.676
Mesh	✓	**0.829**	**0.848**	**24.15**	**24.86**	**0.121**	**0.119**	40.11	**38.83**	**0.675**	**0.680**

Table S1. **Ablation on control signal.** We train and evaluate InterDyn on DexYCB [11]. We ablate: the type of control signal (mask, joints, and a colored mesh rendering), the presence of object occlusions in the control signal, and two image resolutions (256×256 & 256×384).

C. Limitations

InterDyn performs best on translations relative to the camera; dropping objects, moving objects toward or away from the camera, and picking up objects. InterDyn struggles with complex non-translational interactions (e.g. throwing one object at another, burying an object, or poking a tower of stacked objects). It also underperforms in scenarios where depth is ambiguous in the input image, see Fig. S2.

We report the 20 video classes on which InterDyn 256×384 performs best and worst in terms of motion fidelity, alongside the number of videos for each class in the validation set and the average motion fidelity score for that class, see Tab. S3. We generally notice that InterDyn performs less effectively on underrepresented classes within the dataset, while at the same time, many of these underrepresented classes involve complex dynamics, such as spinning, burying, or folding objects.

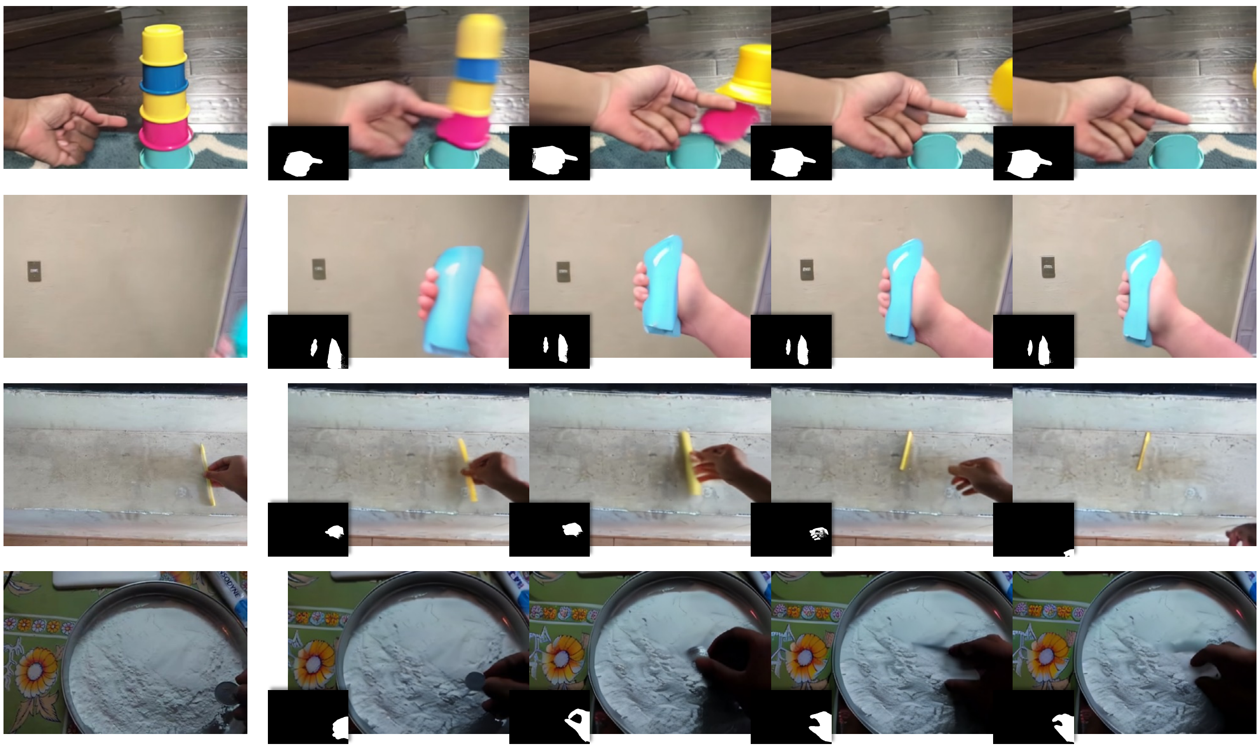

Figure S2. **Limitations of InterDyn.** We show challenging scenarios in which InterDyn underperforms, such as (from top to bottom): object consistency in highly dynamic scenarios, no object in the first frame, depth ambiguity, and burying an object. **Zoom in** for details.

Label	Count	MF ↑ (avg.)
Moving something down	182	0.86
Pulling something from right to left	57	0.84
Moving something up	197	0.82
Pulling something from left to right	83	0.82
Holding something over something	165	0.80
Holding something	103	0.80
Moving something across a surface without it falling down	26	0.79
Pushing something from left to right	123	0.79
Holding something in front of something	138	0.78
Pushing something from right to left	122	0.77
Putting something on a surface	85	0.77
Moving something across a surface until it falls down	28	0.77
Lifting something with something on it	369	0.77
Squeezing something	216	0.77
Lifting something up completely without letting it drop down	66	0.75
Throwing something in the air and letting it fall	6	0.75
Moving something closer to something	105	0.75
Holding something next to something	135	0.75
Putting something that can't roll onto a slanted surface, so it stays where it is	15	0.75
Trying to bend something unbendable so nothing happens	74	0.74

(a) **Top 20 categories.** Contains many translation dynamics with respect to the camera, such as moving something up or from left to right.

Label	Count	MF ↑ (avg.)
Spinning something so it continues spinning	51	0.47
Poking something so that it falls over	42	0.46
Pulling something out of something	33	0.46
Folding something	187	0.46
Poking something so it slightly moves	71	0.45
Spinning something that quickly stops spinning	47	0.45
Taking something out of something	66	0.45
Unfolding something	122	0.44
Putting something, something, and something on the table	60	0.44
Piling something up	27	0.43
Something being deflected from something	10	0.41
Poking something so lightly that it doesn't or almost doesn't move	83	0.41
Burying something in something	4	0.41
Showing something next to something	19	0.40
Pushing something so it spins	23	0.39
Poking something so that it spins around	7	0.39
Putting number of something onto something	5	0.37
Poking a stack of something so the stack collapses	8	0.34
Showing something on top of something	14	0.34
Wiping something off of something	9	0.32

(b) **Bottom 20 categories.** Contains very complex dynamics such as spinning, burying, or showing an object from behind something.

Table S3. **Motion fidelity for different action classes on the Something-Something-v2 dataset.** The table shows the top and bottom 20 categories, together with the number of samples for that category in the validation set.

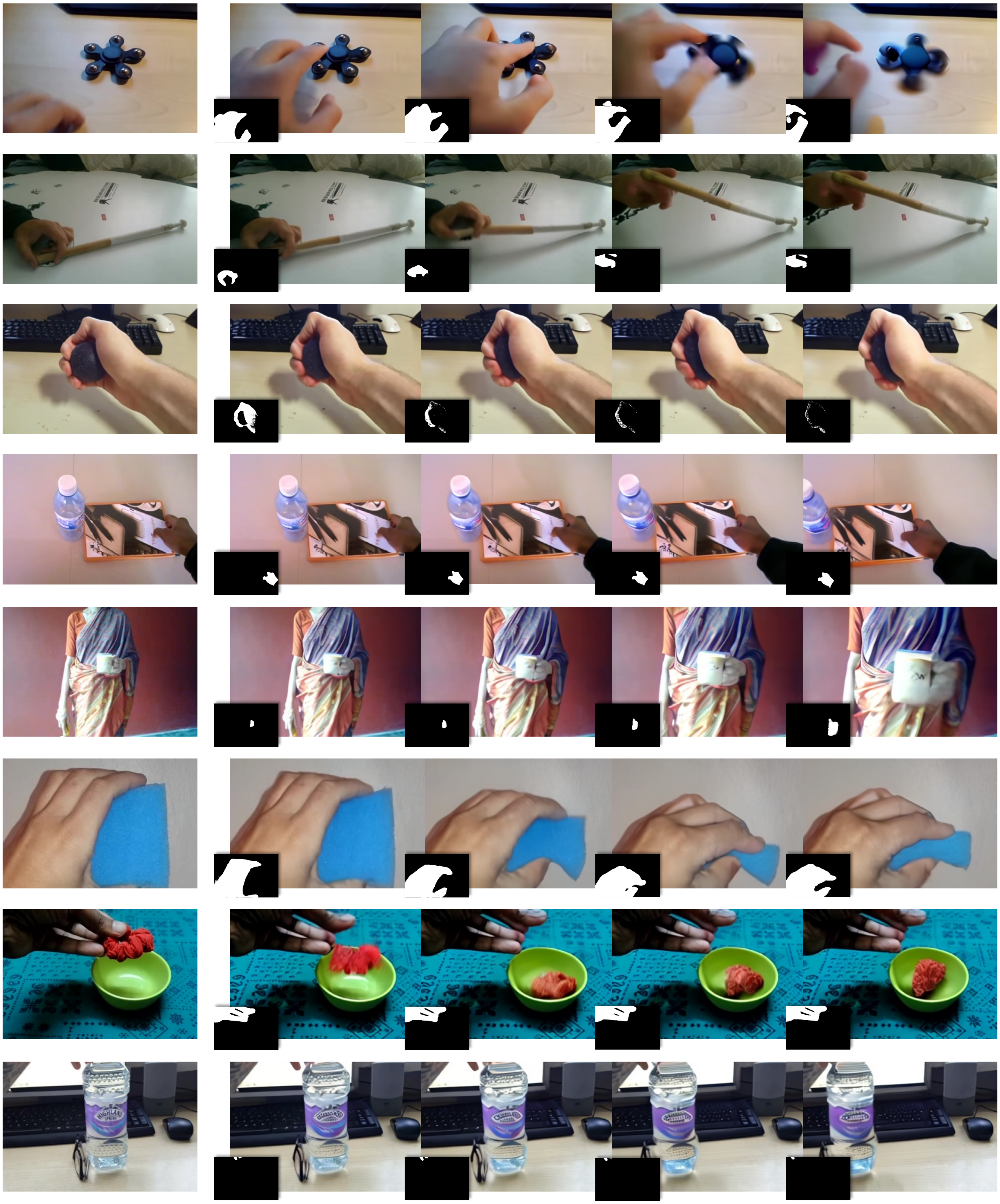

Figure S3. **Additional qualitative results on SSV2.** We show multiple challenging examples, such as (from top to bottom): spinning a fidget spinner, tilting a sleek ridged object, squeezing a ball despite receiving an incomplete control signal, hand object-object interaction, zooming in, squeezing a sponge, dropping a hairband, or hand object-object interaction despite receiving a sparse control signal.

D. Additional Results

We show additional qualitative examples in Fig. S3.

E. Pretending class

Similar to our baseline CosHand [70], we removed the "pretending" class from SSV2 for training and validation, to avoid introducing ambiguous training signals. To compare its performance on this class to our results in Tab. 1, we run InterDyn on the "pretending" class in the validation split (828 samples for 256×384 and 1156 for 256×256), see Tab. S4. While the generations stay consistent in terms of image quality, we notice that motion fidelity is lower. Unfortunately, since FID, KID, FVD, and KVD compare distributions and are heavily dependent on the number of data samples, we cannot directly compare these metrics to those reported in Tab. 1.

Method	SSIM ↑	PSNR ↑	LPIPS ↓	FID ↓	KID ↓	FVD ↓	KVD ↓	Motion Fidelity ↑ [93]
Seer [23]	0.357	9.42	0.657	74.86	0.060	640.06	147.07	—
DynamiCrafter [89] †	—	—	—	34.96	0.016	314.05	34.22	—
CosHand-Independent [70]	0.620	16.79	0.310	**9.85**	**0.003**	123.59	15.97	0.396
CosHand-Autoregressive [70]	0.534	14.80	0.410	24.10	0.012	139.07	11.18	0.512
Ours 256×256	<u>0.666</u>	<u>18.52</u>	<u>0.256</u>	<u>14.29</u>	<u>0.004</u>	**49.02**	<u>-0.131</u>	<u>0.573</u>
Ours 256×384	**0.683**	**18.99**	**0.249**	17.39	<u>0.004</u>	<u>64.18</u>	**-0.450**	**0.572**

Table S4. **Quantitative comparison on the "pretending" class of SSV2.** We compare against Seer [23], DynamiCrafter [89] and two video extensions of our baseline CosHand [70]. Methods denoted with † do not use SSV2 as their training dataset.