Physical Object Understanding with a Physically Controllable World Model

Supplementary Material

Overview of the supplementary

- **Section 1: Additional qualitative results.**
  - Illustrations of parallel motion statistics — probability of motion maps and expected motion maps (Section 1.1).
  - Illustrations of sequential generation of future world states (Section 1.2).
  - Visual Jenga examples (Section 1.3).
  - Illustrations of our movable object discovery algorithm (Section 1.4).
  - Point-prompted movable object segmentation results (Section 1.5).
  - Unprompted movable object discovery results (Section 1.6)
  - 3D object manipulation results (Section 1.7).
- **Section 2: Ablation Studies.**
  - Various ablation studies: **PhyWM** hyperparameter choices and scaling behavior.
- **Section 3: Additional PhyWM architecture & training details**
  - Details on **PhyWM** architecture, quantizer design, and training procedures.
- **Section 4: SpelkeBench dataset details**
  - Dataset collection procedure and more visual examples from the dataset.
- **Section 5: Details about PhyWM's applications.**
  - Further details on procedures that enable **PhyWM**'s applications like computation of parallel motion statistics, segment extraction and 3D object manipulation.
- **Section 6: Baseline evaluation details, and additional qualitative comparisons.**
  - Here we show the settings used for point-prompted movable object segmentation baseline evaluations, and present additional qualitative analysis showing failure modes.
- **Section 7: Statistical significance of results**
  - In this section, we present some analysis of statistical significance of the quantitative results on various object understanding applications in the main paper.

1. Additional Qualitative Results

1.1. Parallel motion statistics

In the main paper, we introduced the use of parallel motion statistics to summarize how different regions of a scene are likely to move under interaction, and we presented qualitative examples in Figure 2. Here, in Figure 2, we provide additional examples spanning a wider range of scenes and objects. These qualitative results further demonstrate the consistency of the predicted motion probabilities and the model's ability to capture coherent motion statistics across diverse visual environments.

1.2. Sequential generation of future world states

In the main paper (Figure 3), we demonstrated the model's ability to generate multiple plausible future world states through sequential autoregressive rollouts. These examples highlighted both the uncertainty in object trajectories and the physical coherence maintained across generations. Here, in Figure 3, we present additional qualitative results spanning a more diverse set of objects, interactions, and scene configurations. These further illustrate the robustness of the model's temporal predictions and its capacity to synthesize physically consistent futures.

1.3. Visual Jenga examples

In the main paper, we applied our world model to the Visual Jenga task and presented qualitative examples in Figure 6b. These results demonstrated the model's ability to identify objects which can be freely moved without disturbing other objects in the scene. We showed that our world model can unstack a stacked structure iteratively while maintaining structural stability in each step. Here, in Figure 4, we provide additional Visual Jenga examples that cover a broader set of tower configurations and intervention types. These results further highlight the model's capacity to reason about physical relationships in complex scenes.

1.4. Illustrations of our movable object discovery algorithm

In Section 4.3 and Figure 4a of the main paper, we outlined our algorithm for discovering movable object segments by simulating virtual pokes and computing correlated motion statistics in the flow responses. Here, in Figure 11, we provide additional examples that visualize the full process—from poke point sampling to segment extraction. These visualizations further demonstrate the robustness and consistency of our flow-based grouping method across a diverse range of objects and scenes.

1.5. Point-prompted movable object segmentation results

In Section 4.3 of the main paper, we evaluated segmentation quality under point-prompted settings on SpelkeBench. Here, we present additional qualitative results comparing our method to several baselines. As shown in Figure 5, our

Table 1. **Ablation studies.** We analyze the effect of modifying the parameters used in the segment discovery algorithm described in Section 4.3 of the main paper, impact of model scaling, SAM2's mask selection strategy, and the importance of incorporating flow tokens for training **PhyWM**. These experiments are conducted on the point-promoted movable object segmentation task on SpelkeBench.

(a) **Parameters used in the object discovery algorithm**

	#seeds=1, #sequential steps=64			
#pokes	1	2	4	8
AR	0.3793	0.3938	0.4673	**0.5251**
mIoU	0.5874	0.5964	0.6419	**0.6786**
	#pokes=1, #sequential steps=64			
#seeds	1	2	4	8
AR	0.3793	0.3987	0.4495	**0.4822**
mIoU	0.5874	0.5943	0.6266	**0.6448**
	#seeds=1, #pokes=8			
#sequential steps	0	64	128	256
AR	0.4622	0.5251	0.5314	**0.5336**
mIoU	0.6413	0.6715	0.6750	**0.6774**

(b) **Scaling behavior**

	#pokes=8, #seeds=1, #sequential steps=64		
	PhyWM (100M)	**PhyWM** (1B)	**PhyWM** (7B)
AR	0.4306	0.5251	**0.5466**
mIoU	0.6166	0.6715	**0.6804**

(c) **SAM2 mask selection ablations**

	Most confident	Random	Least confident
AR	**0.4816**	0.4070	0.3025
mIoU	**0.6225**	0.5900	0.5012

(d) **Importance of using flow tokens**

	CWM	**PhyWM**-RGB	**PhyWM**
AR	0.158	0.412	**0.541**
mIoU	0.334	0.576	**0.681**

method consistently produces cohesive and physically plausible segments, in contrast to alternative approaches that often fragment objects or include extraneous background elements.

1.6. Unprompted movable object segmentation results

Previously, in Figure 4c of the main paper, we evaluated unprompted segmentation on the SpelkeBench benchmark. Here, we present in Figure 7 additional qualitative results comparing our method to baselines such as SAM2 [19], ProMerge [12], and CutLER [23], and FPT [2]. As shown, our method consistently discovers a set of physically plausible segments, whereas the baselines tend to over-segment or under-segment the scene.

1.7. 3D object manipulation results

In Section 4.5 of the main paper, we demonstrated **PhyWM**'s ability to manipulate objects in 3D, and the importance of physically grounded segmentation for object manipulation. Here, in Figure 8 we include further qualitative comparisons of edits generated using our predicted segments versus those from SAM2 [19] and additional comparisons with SOTA 3D editing methods in Figure 9. As illustrated, segments aligned with physical objecthood based on co-movement significantly improve edit realism, spatial coherence, and transformation consistency across multiple 3DEditBench [11] examples.

2. Ablation studies

We conduct a series of ablations to assess the effect of various design choices. All experiments are done on the point-prompted movable object segmentation application described in Section 4.3 of the main paper.

- Table 1a validates the segment extraction framework detailed in Section 4.3 of the main paper: averaging across multiple pokes and seeds improves and stabilizes performance. The table also compares parallel decoding (0 steps) with sequential decoding using varying numbers of autoregressive steps (64-256), a process described in detail at the end of Section 3 of the main paper. Results show that while sequential decoding improves upon parallel mode, performance gains diminish beyond 64 steps, suggesting that most significant causal dependencies can be captured with relatively few steps.
- Table 1b shows that scaling the model is beneficial up to the 7B parameter range.
- Table 1c compares the mask selection strategies for running SAM2 evaluations on SpelkeBench in multimask mode. Specifically, SAM2's multimask mode outputs a set of masks and their associated confidence score. To give SAM2 the fairest chance on the benchmark, we perform an ablation study on mask selection strategy, including most confident, random, and least confident. We find that choosing the most confident mask outperforms random or least confident selection strategies, but still falls short of our approach.
- Table 1d shows the importance of using flow tokens by comparing **PhyWM** to models trained without flow

tokens. We evaluate CWM[3], **PhyWM**, and **PhyWM**-RGB—a variant trained to operate in RGB space rather than flow space (following the same training methodology described in Section 3 of the main paper except without the flow tokens). For **PhyWM**-RGB, we use the same RGB patch motion counterfactual method proposed in CWM for extracting plausible flow fields. While **PhyWM**-RGB outperforms CWM due to its multimodal generative capabilities, **PhyWM** with flow tokens substantially outperforms the RGB-only variant, as flow provides a more intuitive control surface for exercising virtual physical interventions in the scene.

3. Additional PhyWM specifications

Pointer tokens enable random order sequence constructions, making probabilistic models tractable. **PhyWM** adapts the causal autoregressive modeling paradigm for high-dimensional data. Traditional GPT-style transformers predict sequences in a preset, hard-coded order. While this is a natural fit for one-dimensional data such as language, it becomes an unnecessary, and potentially harmful inductive bias when modeling higher-dimensional data. Most autoregressive image modeling approaches simply accept this bias, while we introduce a new token type—the pointer—which allows us to serialize the data in arbitrary order.

Pointer tokens enable us to package random-access traversals over high-dimensional data structures (such as images) into one-dimensional sequences of tokens for efficient causal pretraining, by interleaving pointer tokens among the content tokens, which represent the actual data, as illustrated in Figure 1b in the main paper.

In addition to freeing us from the raster-order generation bias, pointer tokens allow us to condition our model on any subset of the image, learning complex multidirectional conditioning relationships in the data, essentially learning the underlying probabilistic model (PGM) of the world. They also allow for partial patch conditioning, and patch regeneration during inference. All of this functionality can still be simply expressed as a causal autoregressive sequence of tokens, and thus can be modeled and optimized as a standard LLM.

The sequence model formulation transforms the challenge of learning probabilistic models at scale. The key insight is that our sequences represent traversals through the probabilistic model—each pointer-content pair corresponds to visiting and observing a node in the graph. By modeling these traversals autoregressively, we approximate the full joint distribution through conditional factorization. The pointer mechanism ensures we can still query any conditional distribution at spatiotemporal locations, but now through tractable sequential prediction rather than exponentially complex full inference. This recasts the seemingly intractable problem of learning a complete probabilistic model over high-dimensional visual data as a standard GPT-style modeling problem.

Our learned local quantizer. Most visual autoregressive models utilize popular off-the-shelf quantizers such as VQ-GAN [7], VQ-VAE [21], or the Cosmos tokenizer [1]. While such standard quantizers achieve strong compression ratios, they do not preserve locality of the patches within the token space, but rather encode the image in its entirety as a global code. While some locality is certainly present in the token representation, swapping any given token can modify the representations associated with patches on the other side of the image. While it is naturally more efficient to compress information globally, this comes at a cost of a less interpretable and less controllable latent (token) space.

Instead of compressing the whole frame into global codes, the **PhyWM** architecture uses a Hierarchical Local Quantizer (HLQ): a convolutional autoencoder whose receptive field never crosses patch boundaries during encoding. Each 16×16 RGB pixel patch is encoded into a sequence of four 16-bit codes using finite scalar quantization [14], yielding a 65,536-token vocabulary for RGB images. A second, similar quantizer is used to quantize optical-flow patches. No information from neighboring patches leaks into a code, preserving strict locality. This lets local downstream interventions—masking, overwriting, or re-ordering individual patches—behave predictably. Additionally, it makes the autoregressive modeling objective better aligned with natural-language modeling assumptions of token independence. A decoder accepts local 16-bit codes at every spatial location, and decodes them jointly into the input image/flow map that produced them.

Dataset preparation. The world model is pre-trained on BVD (Big Video Dataset [11])—a 7,000 hour dataset of diverse Internet videos mixed with standard 3-D vision datasets such as ScanNet++ [26], CO3D [20], RealEstate-10K [27] and standard video datasets such as Kinetics [10], SomethingSomethingv2 [9] and OpenX embodiment [5]. Camera pose information is provided to the model whenever available in the dataset, and optical flow for every frame pair is computed with the SeaRAFT [24] model and quantized. The BVD also consists of internet videos automatically crawled using search queries generated by LLaMA 3 [6]. The queries targeted videos containing rich physical dynamics, diverse environments, and varied objects. Specifically, action categories from Kinetics400 [10] were expanded with additional sports, physical activities, and product review categories. To ensure training relevance, we filtered videos by requiring a minimum level of optical flow and by applying CLIP [18]-based keyword alignment. Positive keywords included *action*, *activity*, *motion*, and *place*, while negative keywords included *animation*, *cartoon*, *face*, *game menu*, *graphic*, *map*, *newscast*, *person*, and *screenshot*. Alignment was quantified by the dot product

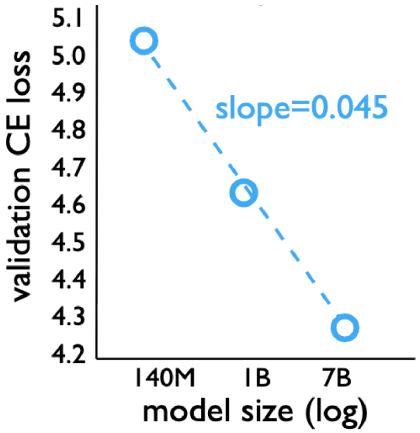

Figure 1. **Scaling laws that show our model obtaining lower loss when given more parameters.**

between CLIP embeddings of keywords and video frames.

Key Training Details. We train an 80M-parameter RGB HLQ on a combination of ImageNet and Open Images, and an 80M Flow HLQ on the BVD dataset. We quantize 512x512 RGB images and flow maps into tokens and train a 7B-parameter Φ model on a dataset of 3 million RGB video clips. This sums up to about 1.4 trillion tokens. We train Φ with causal sequences of 2 frames, spanning up to 1 second of video. Mixed-precision training on 64 H100 GPUs at 65% MFU yields 490 TFLOPS/device ($\sim$31 PFLOPS total) sustained.

All models were trained autoregressively using cross-entropy loss on next-token prediction with a batch size of 512. We first train with only RGB and camera pose tokens for 5×10^5 steps under a Warmup-Stable-Decay (**WSD**) schedule. The learning rate was linearly warmed up over 2,000 iterations to 3×10^{-4}, held constant until the final 1×10^4 steps, and then decayed linearly to zero. Training was continued for an additional 2×10^5 steps with optical flow tokens, in addition to RGB and camera tokens, where the learning rate was re-warmed over 800 iterations to 3×10^{-4}, maintained at this value, and decayed linearly to zero during the last 1×10^4 steps. Each training step takes approximately 3.8s. It has been shown that WSD achieves similar or better performance than cosine schedules of the same length. The use of WSD is particularly important here not because of superior performance, however, but because it enables continual training in the flow tokens integration step we discussed above.

Scaling Properties of `PhyWM`. A critical advantage of the **`PhyWM`** architecture is its predictable scaling behavior, inheriting the well-established scaling laws of language models while extending them to visual domains. We trained **`PhyWM`** models from 100M to 7B parameters and observed consistent improvements in validation loss across three orders of magnitude (Figure 1). The reliable scaling without saturation even at 7B parameters suggests that further scaling would yield continued benefits, validating that **`PhyWM`** successfully transfers the scaling properties of autoregressive language models to structured visual data. We also show scaling properties on downstream applications like point prompted segmentation in our ablation studies here in Section 2.

4. SpelkeBench dataset details

Here we provide more details on SpelkeBench [15] — the evaluation benchmark we use for movable object discovery in Sections 4.3 and 4.4 of the main paper. This benchmark is designed to assess whether segmentation algorithms can identify movable objects — defined as regions that move together — unlike existing datasets such as COCO [13], which emphasize semantic or instance labels. We expand on collection procedures and provide qualitative examples from the dataset.

4.1. Collection procedure

SpelkeBench curates a dataset from two complementary sources: the EntitySeg benchmark [17] and the OpenX-Embodiment robotics dataset [5]. While EntitySeg is designed for high-resolution internet imagery with dense segmentation annotations, OpenX consists of real-world, egocentric robot interactions. Since OpenX does not provide segment labels, they are manually annotated for a subset of 50 images. These annotations reflect the types of movable objects relevant for physical interaction and manipulation tasks that are central to robot learning. For EntitySeg, a high-quality subset of 500 images is extracted using a three-stage filtering pipeline that filters out the annotated segments in the dataset which do not align with the definition of movable objects:

- *Stage 1: Removal of amorphous background regions.* All regions labeled as "stuff"—such as sky, ground, or terrain—based on the standard stuff-vs-things taxonomy [17] are excluded. These regions lack the individuated, cohesive properties associated with movable objects and are typically not physically manipulable entities.
- *Stage 2: Filtering non-movable object categories.* Despite being labeled as "things", certain objects like kitchen sinks, traffic signs, or large fixtures are functionally immovable in real-world settings. These are identified and removed through manual inspection.
- *Stage 3: Final curation of diverse, high-quality scenes.* From the filtered pool, 500 images that contain only movable object-consistent annotations are selected. It is ensured that this set is diverse in terms of object types, spatial arrangements, and scene complexity.

4.2. SpelkeBench qualitative examples

In Figure 10, we present additional examples of images and associated ground-truth movable object annotations in SpelkeBench. These examples further show the failures of existing segmentation benchmarks such as EntitySeg [17] and SA-1B [19], which often contain segments that diverge from the movable object criteria, such as amorphous background regions, or subregions of objects. SpelkeBench's physically grounded segment annotations are therefore better candidates to evaluate the object discovery applications of **PhyWM** presented in the main paper.

5. Details about PhyWM's applications

Here we provide additional details for the algorithms that enable various object understanding applications of **PhyWM**.

5.1. Parallel estimation of motion statistics

In Section 4.1 of the main text, we introduced the expected motion map as the probability weighted average of flow vectors that map to flow tokens. This flow token to flow vector mapping needs to be done as a necessary step before the expectation can be computed. Here we describe the procedure used for doing that.

Flow token epigraphy. **PhyWM** uses a learnt *local patch quantization* to produce flow tokens, but relies on a *global decoder* to generate coherent, high-quality flow fields. As a result, tokens cannot be interpreted by decoding them in isolation—their meaning emerges only in the context of the full sequence. However, since the tokenizer is local, we can find which continuous flow vectors map to it by performing a kind of token space epigraphy—by assigning meaning to discrete flow tokens through statistical aggregation of typical input flow fields that produced them:

$$f_j \mapsto \mathbf{v}_j = \frac{1}{|S_j|} \sum_{\mathbf{u} \in S_j} \mathbf{u},$$
$$\text{where,} \quad S_j = \left\{ \mathbf{u} \in \mathbb{R}^2 \mid \text{tokenizer}(\mathbf{u}) = f_j \right\}.$$

5.2. Movable segment discovery procedure

As discussed in Section 4.3 of the main paper, we use **PhyWM** for movable object discovery by computing the statistical aggregate of pixel-to-pixel correlated motion statistics across a variety of applied virtual pokes and generation seeds. In this section, we explain the intuition behind the process and how it allows for a more expressive definition of movable objects.

We build upon the idea of counterfactual probing introduced in CWM [3], where movable objects are discovered by simulating localized virtual pokes through local patch motion interventions and analyzing the outcome. However, due to its regression-based nature, CWM produces a single deterministic output, which in practice does not accurately represent the physical world where there are multiple physically plausible outcomes for a poke. Consider a simple example of moving a person's hand, the hand may move independently of the body or the entire body may move with the hand–both being valid outcomes. However, because of the aforementioned deterministic nature of CWM, it is forced to average over these distinct possibilities, leading to ambiguous or blurry motion completions that fail to reveal which parts of a scene tend to move together.

In contrast, because of the generative nature of **PhyWM** which generates multiple plausible future motions of a scene, we can operationalize movable objects with a more expressive definition as groups of pixels that consistently move together across multiple plausible outcomes of a world model, under different virtual pokes. This requires modeling the distribution of possible responses to external forces. As such, the algorithm defined in Section 4.3 of the main paper is a natural stochastic extension of the original CWM counterfactual procedure as instead of a single prediction, we use **PhyWM** to produce a diverse set of imagined flow completions for various virtual pokes at a candidate spatial location. Diversity arises from two sources of randomness:

1. **Sampling flow tokens** from the learned distribution $\Pr[v|\mathbf{X} \circ p]$ at a pointer p: we draw multiple flows $f_k \sim \Pr[v|\mathbf{X} \circ p]$ to explore the local responses the model deems plausible (e.g. if an object is on a table, we would not sample pokes down into the table).
2. **Varying the decoding order** of spatial indices p_k: Because **PhyWM** is a sequence model, tokens decoded earlier condition those decoded later. Shuffling the order therefore changes how motion propagates through the object—e.g. decoding the torso *before* the leg yields a different global outcome than decoding the leg first.

In our evaluations, we use 8 pokes, and also perform a refinement step by zooming into the object with the initial estimate of the segment, and repeating the extraction procedure.

5.3. Unprompted segmentation procedure

Here we provide additional details about our unprompted segmentation algorithm described in Section 4.4 in the main paper. In many real-world settings, especially in robotics, it is advantageous to automatically discover *every* independently movable object in a scene without requiring manual point-prompting. For example, a household robot tasked with clearing a dining table must infer that a plate and its contents will move as a unit, while a napkin resting on the plate is an independent entity, so it can plan appropriate grasps and avoid unintended collisions.

We now describe a method to extract the full set of movable object segments in a scene automatically. Our ap-

proach consists of two steps. First, as described in Section 4.4 we poke the scene at multiple locations sampled from the probability of motion map. Then from the model's flow responses, we compute a dense pixel-to-pixel affinity matrix that captures the likelihood that a pair of pixels will move together. An iterative clustering algorithm is applied to this matrix to isolate a complete set of independently movable entities.

Computing the affinity matrix. We begin by sampling locations from the motion probability map. These points are where we "poke" to collect flows.

$$\mathcal{K} = \{p_1, p_2, \ldots, p_N\} \subset \mathcal{I}, \qquad \mathbb{P}_{\text{motion}}[p_i] > \tau_p.$$

We then build a motion descriptor for each pixel using the following procedure:

For each $n = 1, \ldots, N$, choose R poke-directions $\{f_n^{(r)}\}_{r=1}^{R}$. For each (n, r) and each of $t = 1, \ldots, T$ random seeds, compute the flow completion given the input image tokens $\mathbf{r}^0$,

$$\hat{\mathbf{f}}_t^{\,(n,r)} \overset{\text{seq}}{\sim} \Phi(\mathbf{r}^0 \circ [c = 0] \circ [p_n, f_n^{(r)}]; \text{flow}; \texttt{seed} = t)$$

Then for each $u \in \mathcal{I}$, where $\mathcal{I}$ is the set of 2D pixel locations, the motion descriptor,

$$\varphi[u] = \left[\hat{\mathbf{f}}_1^{\,(1,1)}, \ldots, \hat{\mathbf{f}}_t^{\,(n,r)}(u)\right] \in \mathbb{R}^{2\,N\,R\,T}.$$

Finally, the affinity matrix can be described as the pairwise dot product of motion descriptors:

$$A[u, v] = \varphi[u]^\top \varphi[v], \quad \forall\, u, v \in \mathcal{I}.$$

For simplicity, we denote $A[u]$ to be the affinity of the pixel u with the rest of the image.

Clustering the affinity matrix to extract segments. Given the precomputed affinity matrix A, we extract segments in an iterative "select–threshold–refine" loop. At each step, we choose the most confident probe center k_{i^*}, defined as the one whose affinity-row $A[k_{i^*}]$ has the highest mean over all pixels—indicative of strong binding to the other pixels that make up the object. We apply Otsu's method to threshold this row, yielding an initial mask $M^{(0)}$. We then gather all remaining poke points k_j that lie within $M^{(0)}$ and average their affinity-rows to form:

$$A_{\text{avg}} = \frac{1}{|\{j : k_j \in M^{(0)}\}|} \sum_{k_j \in M^{(0)}} A[k_j]$$

We threshold A_{avg} via Otsu's method to obtain a refined mask $M^{(t)}$, for $t = 0$. All centers contained in $M^{(t)}$ are then removed from consideration, and the loop repeats on the remaining set of poke points. Once no poke points remain, the algorithm returns the complete set of extracted segments $\{M^{(1)}, \ldots, M^{(T)}\}$. Non-maximum suppression is then used to remove duplicate segments. Figure 12 illustrates this procedure using an example.

5.4. 3D Object Manipulation Procedure

As discussed in Section 4.5 in the main paper, **PhyWM** achieves state-of-the-art object manipulation performance by leveraging 2D optical flow fields that encode 3D transformations. Specifically, to perform 3D object manipulation, we create a flow field where the flow on the surface of the object characterizes the 3D transformation to be performed, with the flow of the background set to 0. We use this to condition the predictor to move the object, but keep the background fixed. To generate these flow fields, we perform the following procedure:

- Step 1: Unproject the depth map of the input image, obtained using off-the-shelf supervised metric depth estimator Depth Anything V2 [25], to obtain a 3D point cloud.
- Step 2: Apply the desired rigid transformation to the object's point cloud while keeping background points unchanged.
- Step 3: Re-project the transformed point cloud and compute the displacement relative to the original pixel positions to generate the 2D flow field.
- Step 4: Finally, **PhyWM** then generates the manipulated image given the computed object-masked flow map and the input image.

6. Baseline evaluation details and additional qualitative comparisons

In Section 4.3 of the main paper, we evaluate various baselines on SpelkeBench using point-prompted segmentation, wherein each method receives a poke point and must output a binary segment. In this section, we expand on how baseline methods generate binary segments from single-point prompts for evaluation, and present additional qualitative analysis showing failure modes of these methods.

- **SAM2 [19]**: We use single-point prompting on SAM2 multimask mode and choose the most confident mask (see ablation in Table 1c).
- **DINOv2 [16]**: We extract features using the ViT-G/14 backbone, compute affinity between the poke point's feature and all spatial features, threshold using Otsu's method, and apply Conditional Random Field refinement to obtain the final binary mask.
- **CutLER [23] & ProMerge [12]**: Both methods generate multiple candidate segments. We select the segment (if any) containing the poke point.
- **ForcePrompting [8], PerceptionAsControl [4]**: For these drag-based approaches, we use the poke point to initialize a trajectory to generate the edited image. We then compute optical flow between original and edited image using RAFT and apply Otsu thresholding to the flow magnitude to extract the binary segment. Some failure modes of these models are illustrated in Figure 13
- **FPT [2]**: FPT produces a flow completion given a virtual

Task	Baseline	Metric	$\Delta \pm$ SE	95% CI	p-value
Point Segmentation	**PhyWM** vs SAM2	mIoU ↑	$+0.058 \pm 0.010$	$[0.038,\ 0.077]$	< 0.001
	Tab. 1a (col 12 vs col 3)	AR ↑	$+0.059 \pm 0.013$	$[0.034,\ 0.084]$	< 0.001
Automatic Segmentation	**PhyWM** vs ProMerge	mIoU ↑	$+0.14 \pm 0.01$	$[0.12,\ 0.16]$	< 0.001
		AP ↑	-0.07 ± 0.01	$[-0.10,\ -0.05]$	< 0.001
	Tab. 1b (col 6 vs col 4)	AR ↑	$+0.12 \pm 0.01$	$[0.09,\ 0.14]$	< 0.001
		F1 ↑	$+0.02 \pm 0.01$	$[0.00,\ 0.04]$	0.082
Object Manipulation	**PhyWM** vs PasC	EA ↑	$+0.096 \pm 0.030$	$[0.044,\ 0.150]$	< 0.001
	Tab. 1d (row 8 vs row 2)	LPIPS ↓	-0.035 ± 0.009	$[-0.041,\ -0.028]$	< 0.001
		SSIM ↑	$+0.064 \pm 0.006$	$[0.049,\ 0.081]$	< 0.001
	PhyWM seg. vs SAM2 seg.	EA ↑	$+0.143 \pm 0.042$	$[0.064,\ 0.223]$	< 0.001
	Tab. 1d (row 8 vs row 9)	LPIPS ↓	-0.022 ± 0.007	$[-0.028,\ -0.017]$	< 0.001
		SSIM ↑	$+0.016 \pm 0.013$	$[-0.015,\ 0.047]$	0.355

Table 2. Statistical analysis of relative improvement (Δ) of **PhyWM** vs the second best model across three tasks.

poke, similar to our method. We can directly apply our algorithm in Section 4.3 of the main paper to this method.

- **CWM [22]**: Similar to drag-based approaches, we apply interventions at the poke point, compute RAFT flow between original image and intervention outcome, and threshold the flow magnitude to obtain the binary segment. CWM merges nearby objects because the model often generates blurry reconstructions, as its RGB pixel regression objective during training does not account for uncertainty. As a result, the flow estimation may produce diffuse or extended motion fields, causing nearby objects to be grouped together, as illustrated in Figure 14

7. Statistical significance of quantitative results

To ensure that our improvements are not only numerically higher but also statistically reliable, we perform a comprehensive significance analysis across all quantitative evaluations. In particular, we focus on the tasks reported in Table 1a, 1b, and 1d of the main paper. Specifically, we report the relative improvement (Δ), along with standard errors (SE), confidence intervals (CIs), and paired t-tests. Across nearly all metrics in Table 2, we observe three consistent trends: (1) the standard errors are small, indicating low variance across samples; (2) the confidence intervals are tight and do not cross zero, suggesting that the observed improvements are consistently positive; and (3) paired t-tests confirm statistical significance with $p \leq 0.001$. Taken together, these results demonstrate that our improvements reflect stable and statistically significant gains over prior methods.

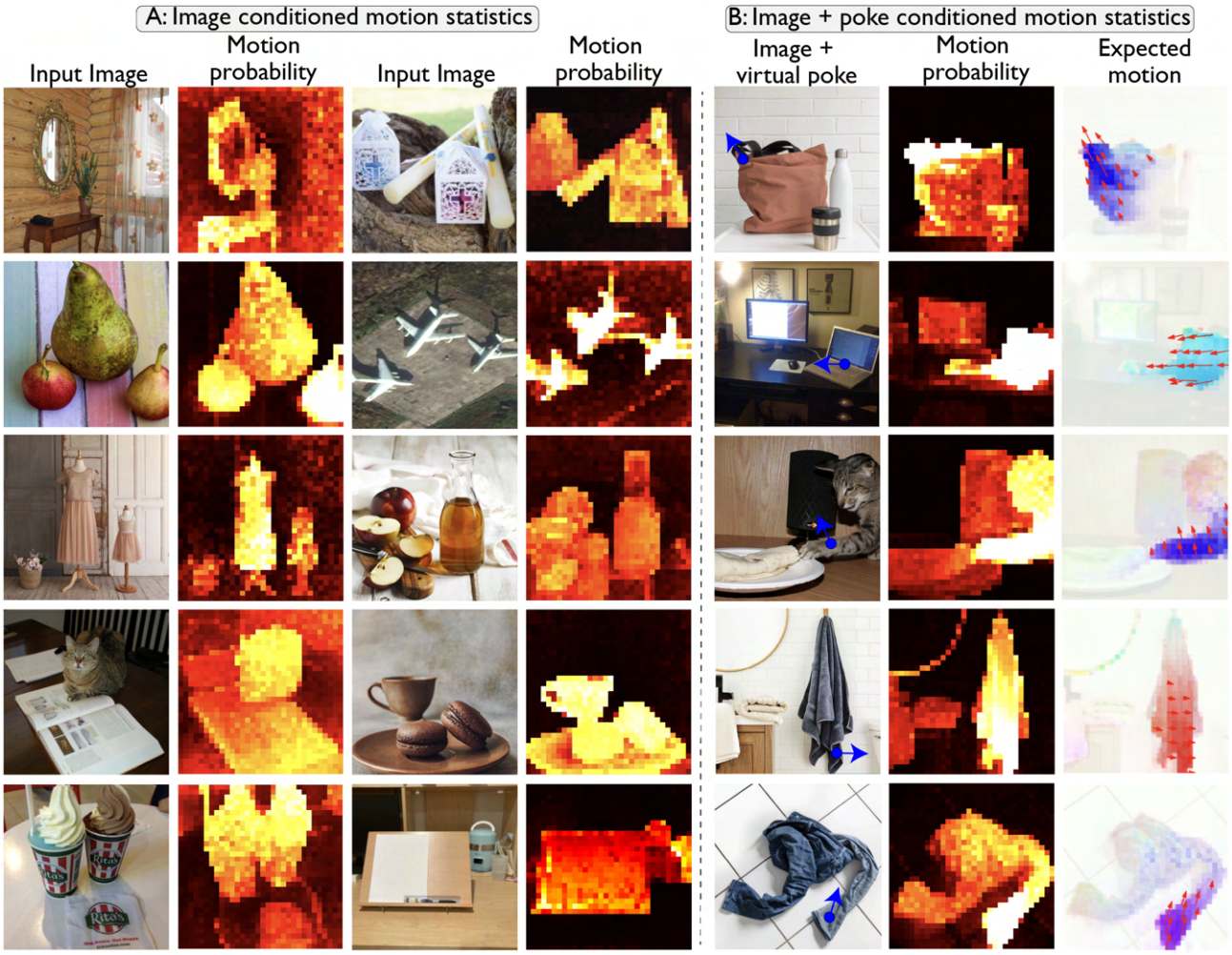

Figure 2. **Motion statistics computed in parallel.** In **Figure A**, we show motion probability computed given the input image and camera stop conditioning. They clearly highlight the parts of the scene that are likely to move. In **Figure B**, we show probability of motion and expected motion maps conditioned on an input virtual poke.

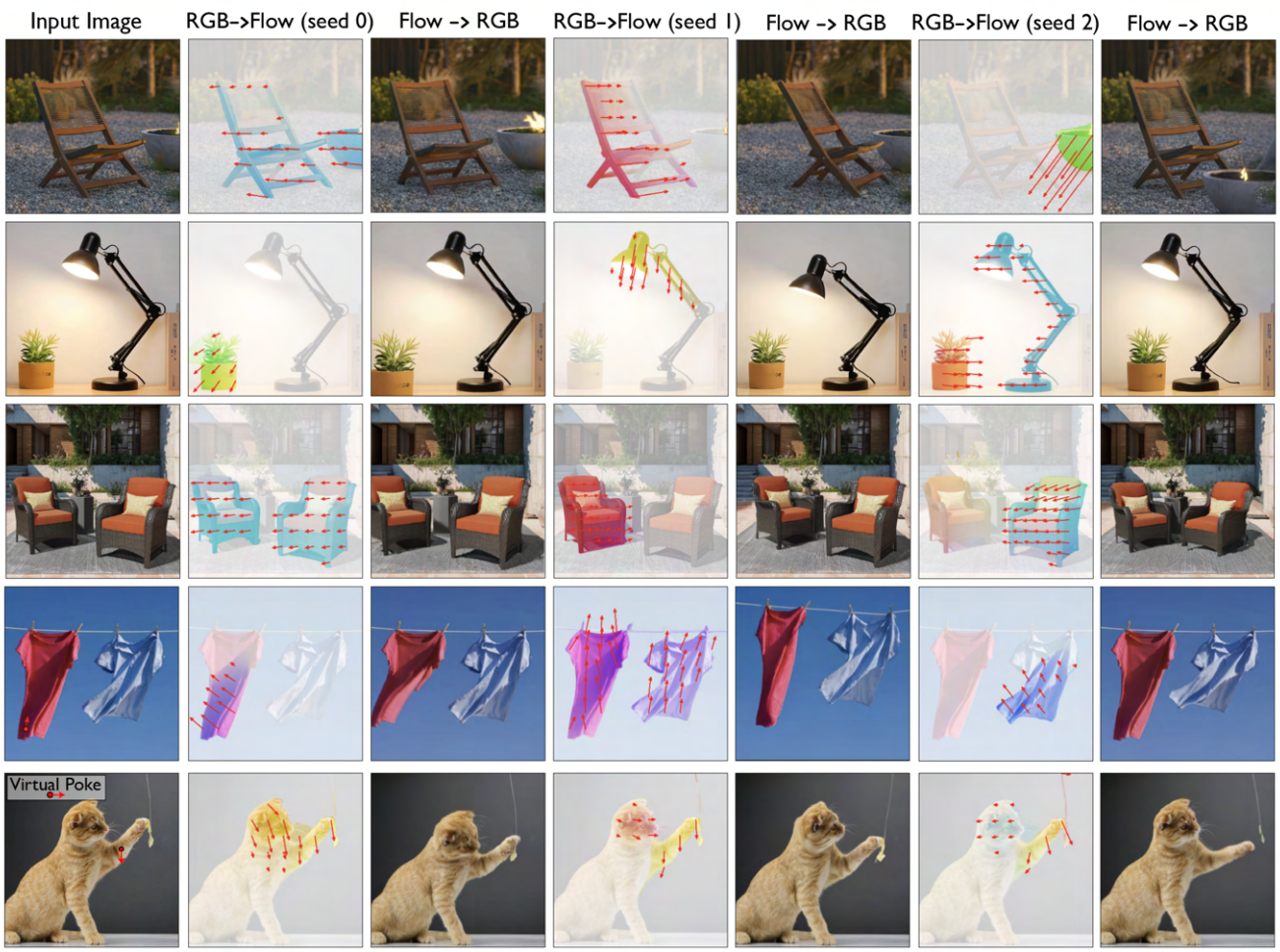

Figure 3. **Sequential generation of plausible object dynamics and appearance.** We show that our model can generate multiple physically plausible scene motions and render them into future appearance states – capturing the true dynamics of the physical world for complex objects. In rows 1–4, the model infers plausible motion patterns directly from a single input image. In the last row, specifying a motion for a part of the object (such as the hand of the cat) generates diverse, yet physically consistent responses for the rest of the body.

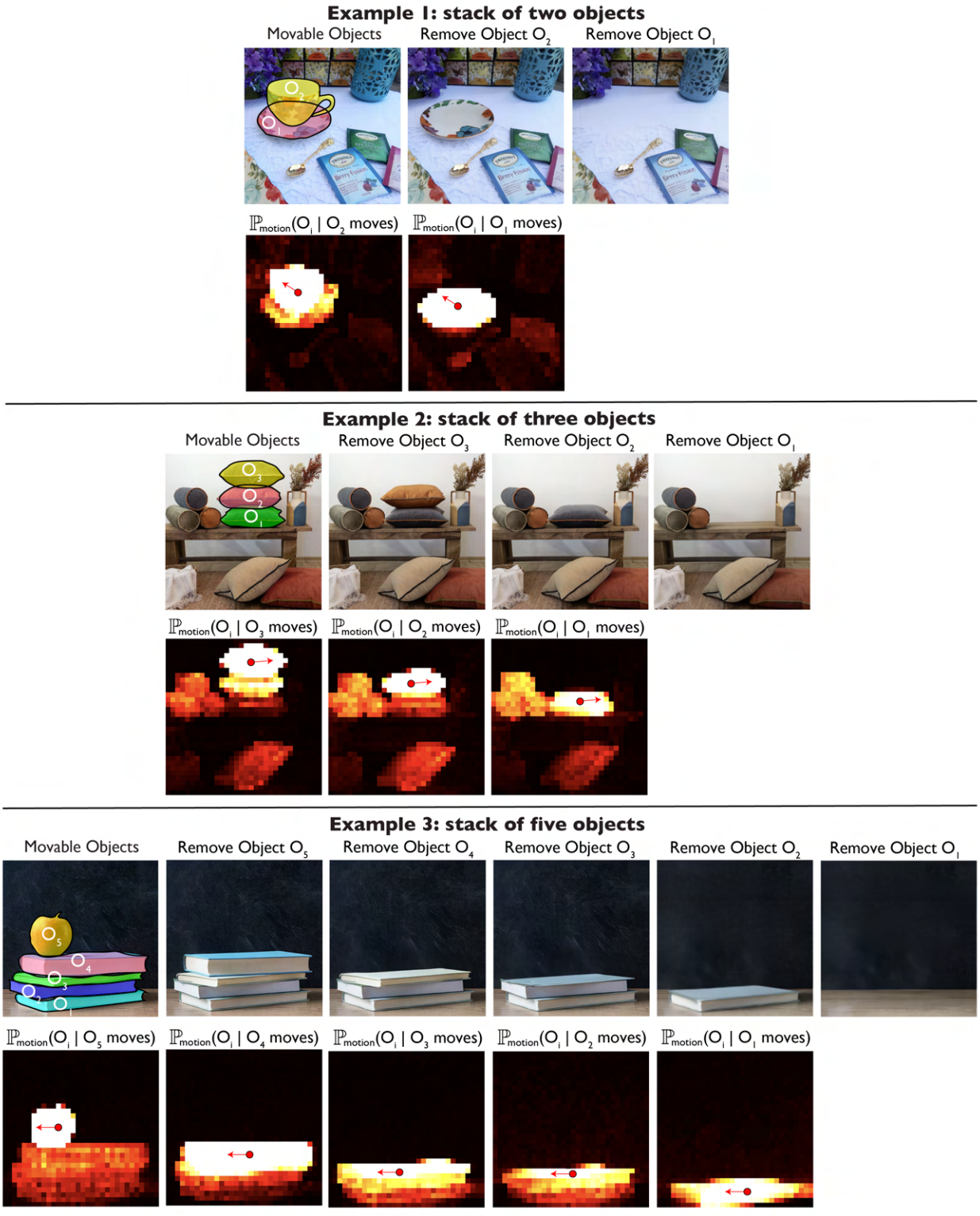

Figure 4. **Reasoning about physical relationships between objects.** Here we demonstrate how probability of motion maps can be used to probe physical dependencies in scenes, enabling applications like visual jenga on three progressively challenging real world examples.

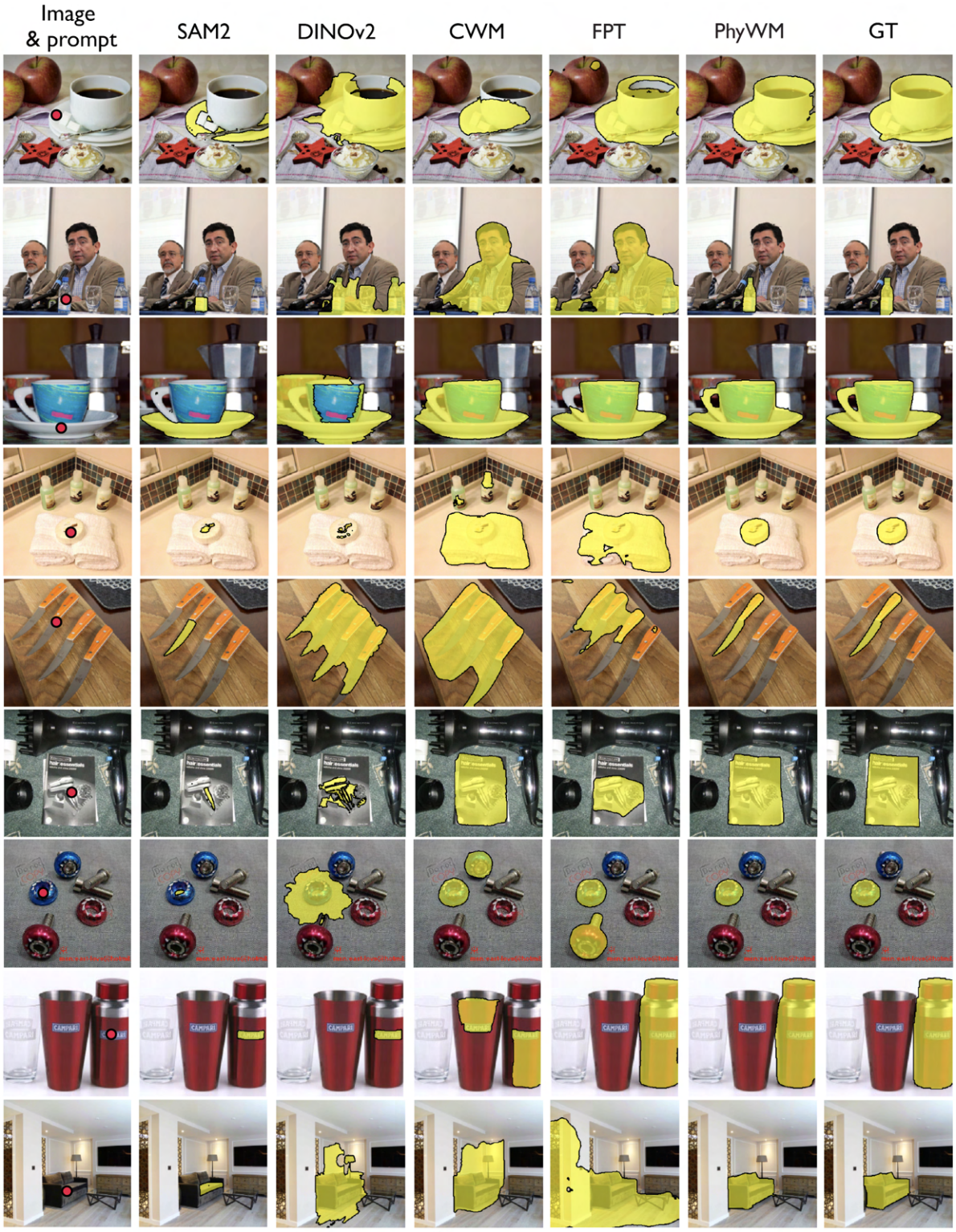

Figure 5. **Additional qualitative results for point-promoted segmentation across models.** **PhyWM** yields sharper segments which are more aligned with the notion of an object as a movable entity as compared to SAM2, DINO, CWM and FPT.

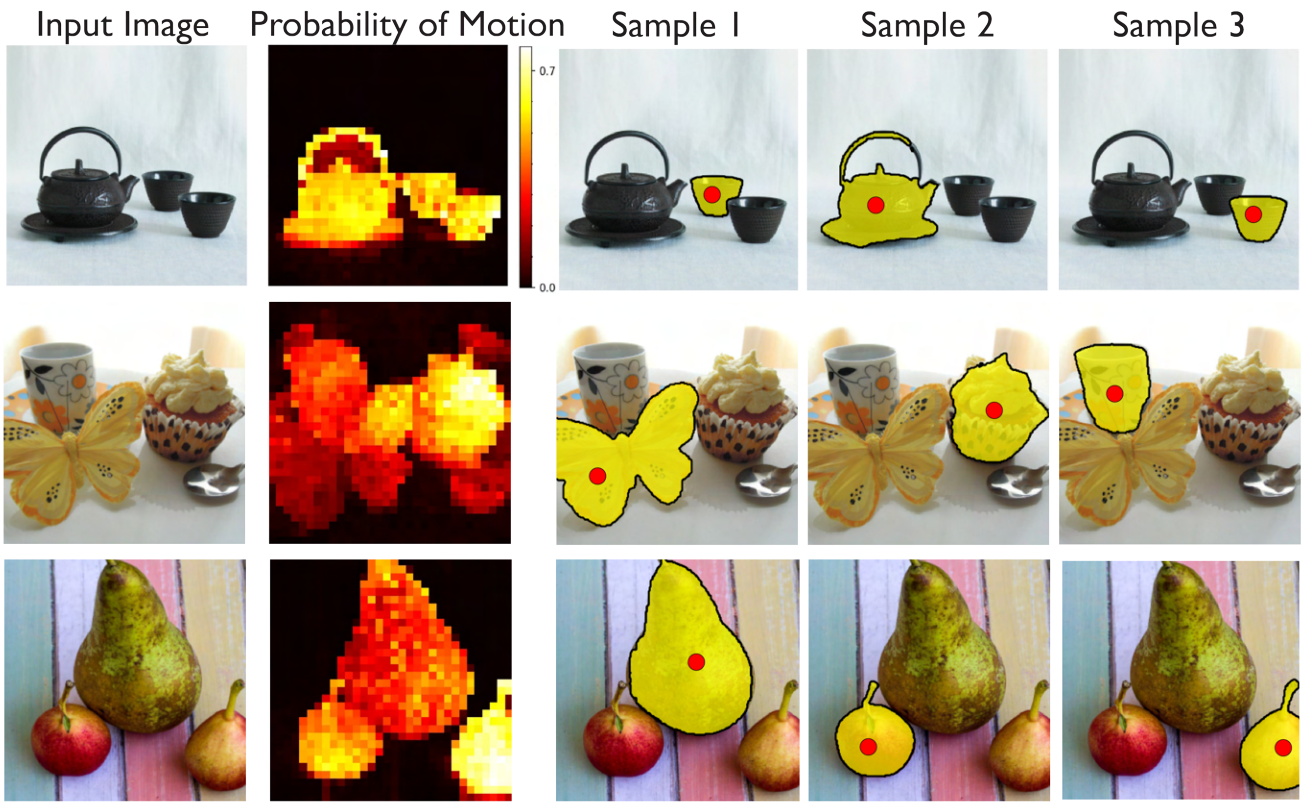

Figure 6. **Illustration of unprompted movable object segment discovery using `PhyWM`.** The corresponding discovered segments are highlighted, demonstrating the ability of **`PhyWM`** to automatically identify every movable object in the scene without manual prompts.

Figure 7. **Additional qualitative results for automatic segmentation across models.** **`PhyWM`** produces a set of physical object segments more consistent with physical co-movement as compared to SAM2, ProMerge, CutLER and FPT. Red regions denote predicted segments that are not matched to GT labels.

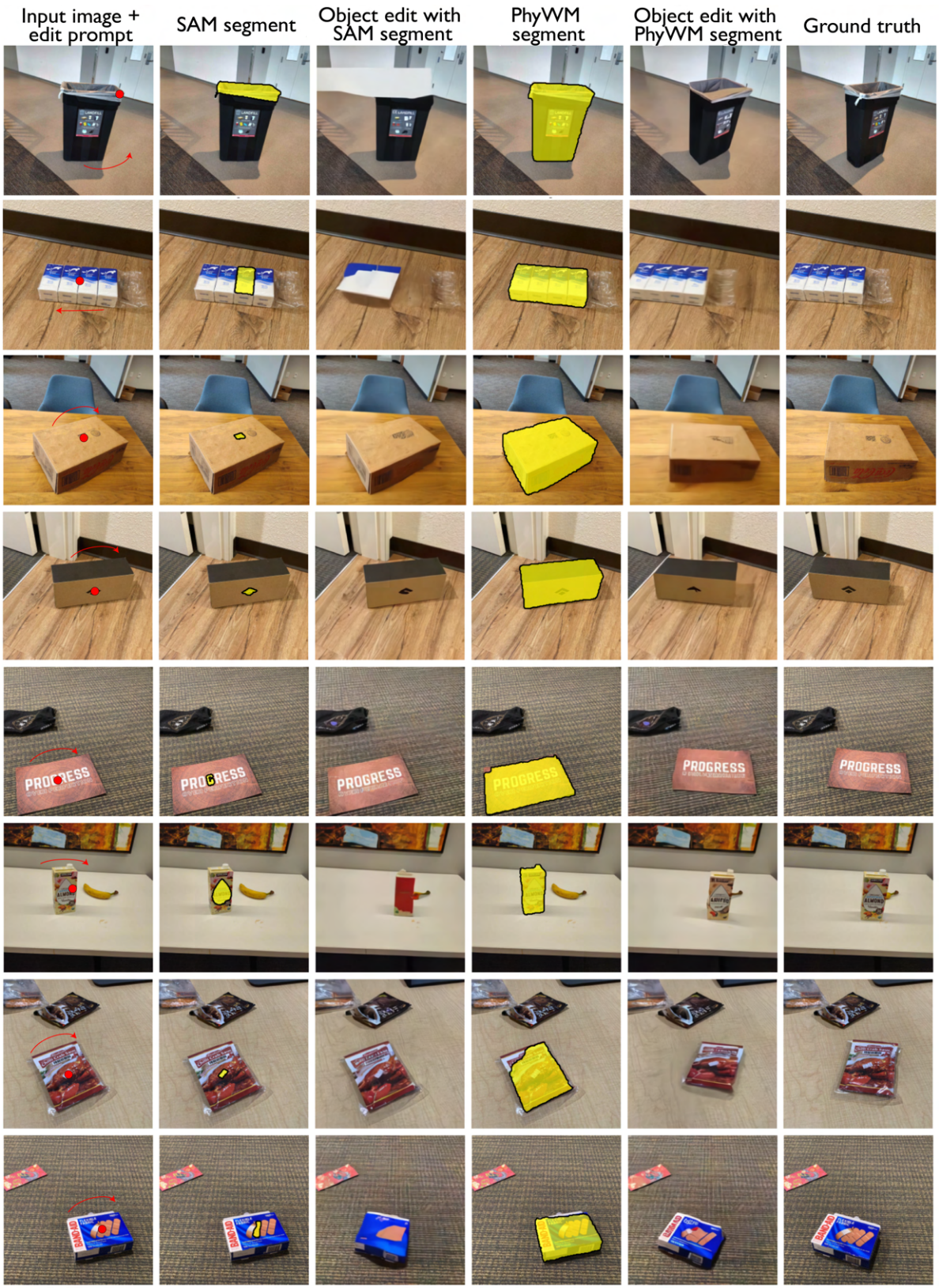

Figure 8. **Additional qualitative comparisons of scene edits using SAM masks versus `PhyWM` segments.** Each row shows the original image, the user click location, and the resulting edited image using different segmentation methods.

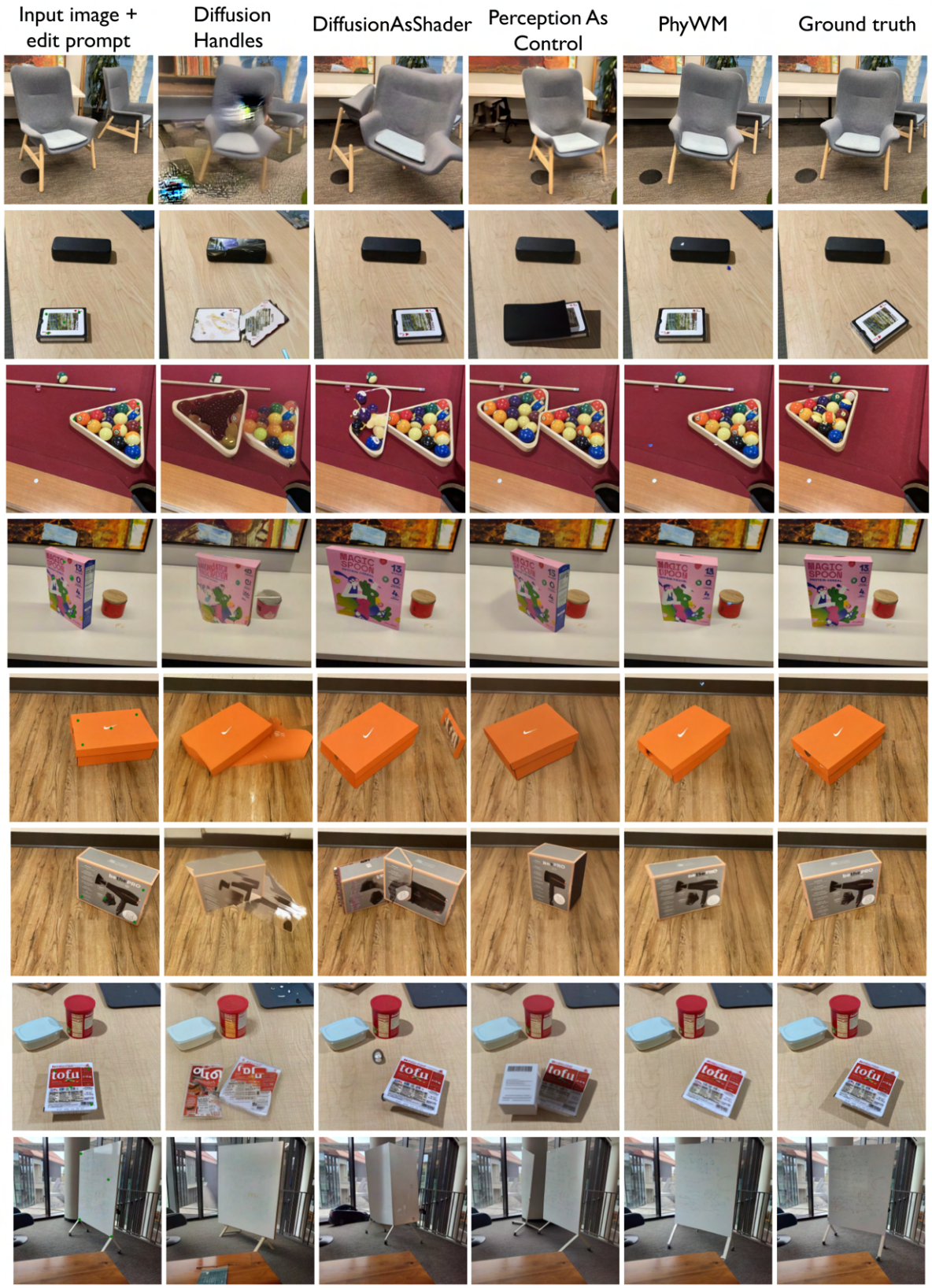

Figure 9. **Additional qualitative comparisons of object manipulation comparisons with SOTA methods.** We compare **PhyWM** to various object-centric image editing methods and show that **PhyWM** enables more physically plausible image edits.

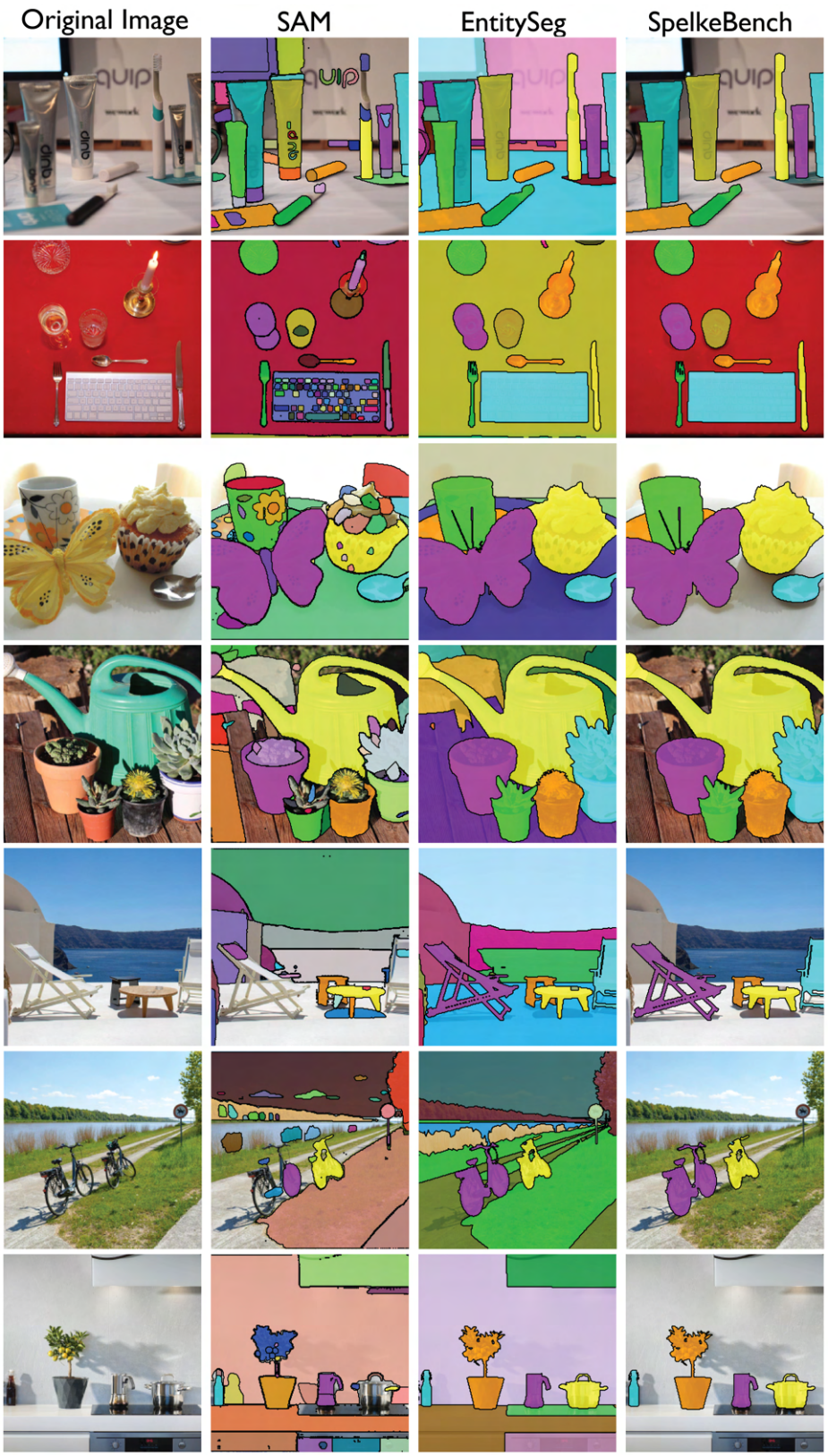

Figure 10. **Qualitative comparison of SpelkeBench vs other datasets.** The visualization demonstrates characteristic differences across datasets: SAM's dataset tends to oversegment objects into constituent parts (i.e., bottle labels, cup designs), EntitySeg frequently includes ill-defined background regions (i.e., ground, wall), whereas SpelkeBench contains segments that better align with the notion of movable objects as units that move together, serving as an appropriate benchmark for the object discovery applications of our world model.

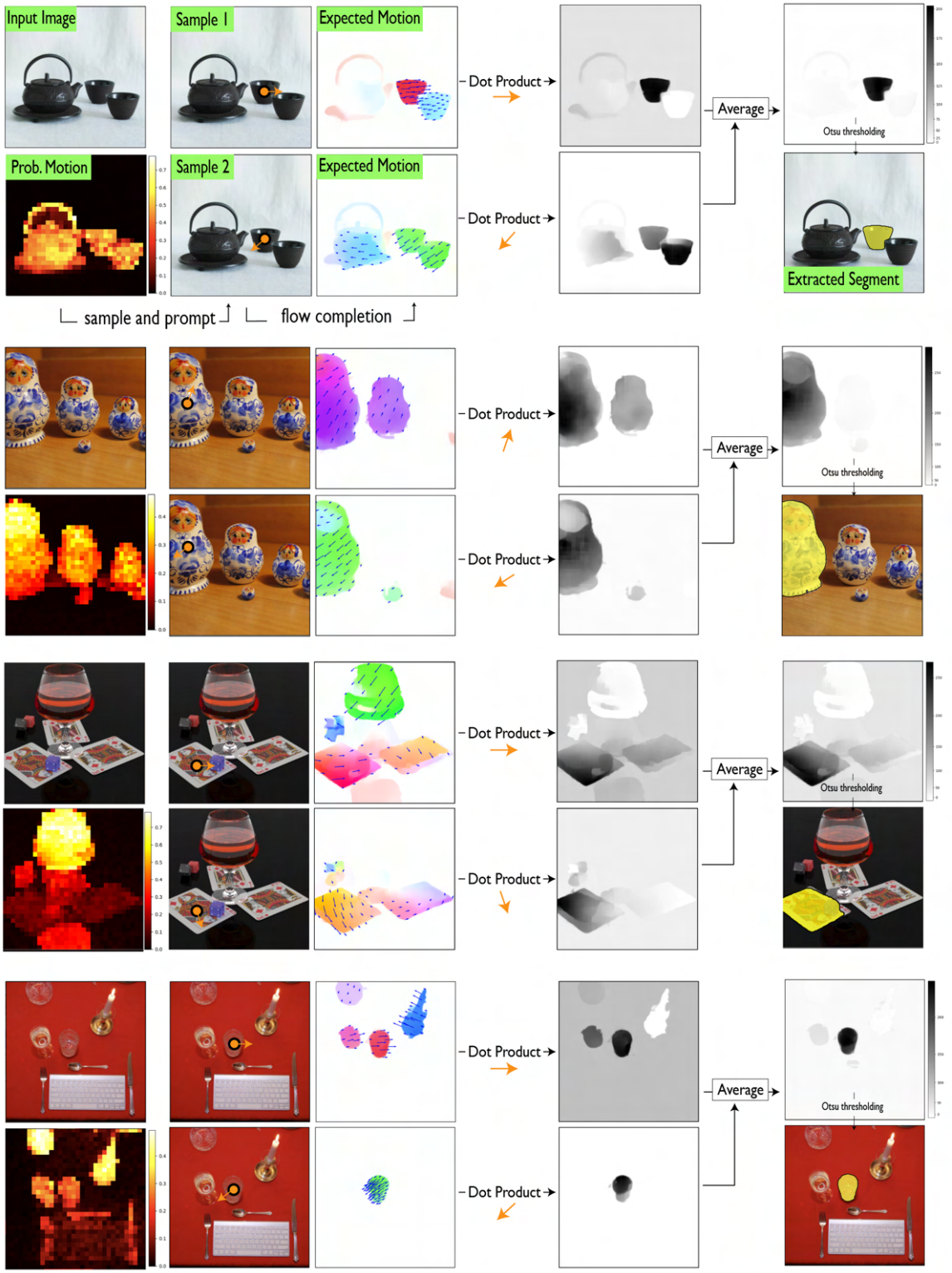

Figure 11. **More illustrations of our movable object discovery algorithm described in Section 4.3 of the main paper**. To discover movable objects, we apply multiple virtual pokes at locations sampled from the $\mathbb{P}_{\text{motion}}$ map (column 2). While the model consistently propagates flow across the poked object (column 3), it also generates unprompted flow on other objects. However, since this unprompted flow varies across pokes and typically diverges in direction from the input poke, it gets suppressed when averaging the dot product (column 4) and helps us isolate independently movable entities as shown in the last column. Note that we average across 8 pokes, but only show two rows here for brevity.

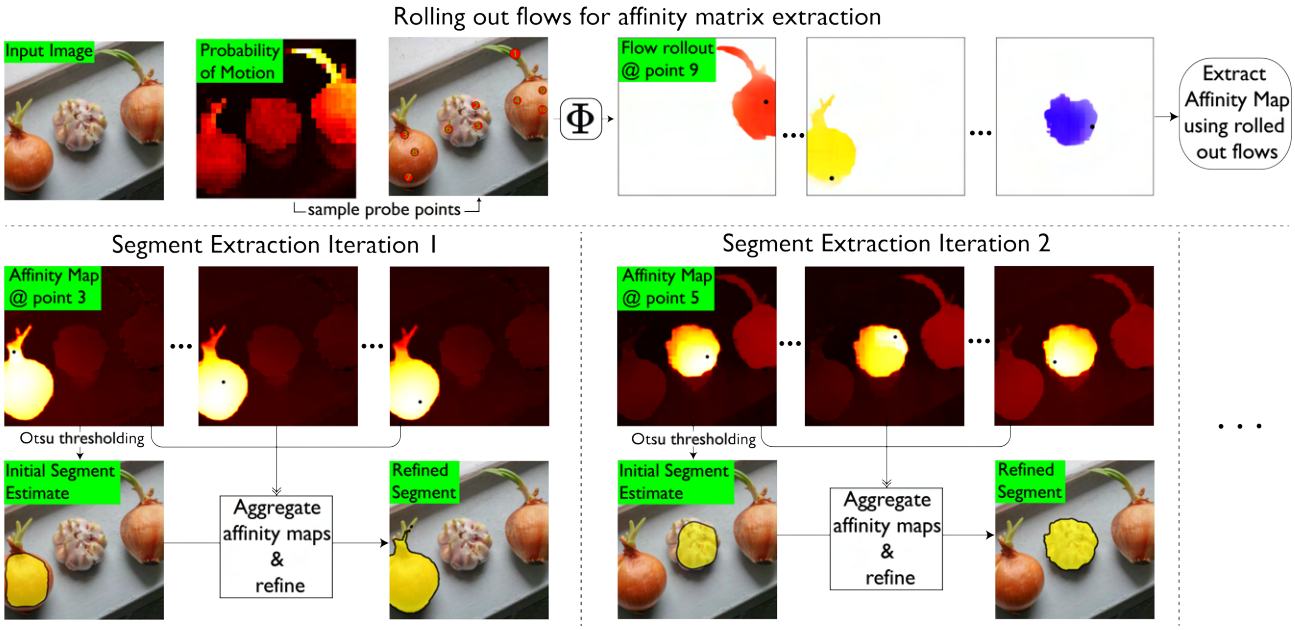

Figure 12. **Unprompted discovery of movable object segments**. We extract probability of motion maps from an image, and use it to sample candidate poke points **(top left)**. We apply a poke to the image at the sampled points and obtain dense flow fields conditioned on the poke **(top right)** which are used to compute affinity maps. As shown in the **bottom** panel, these maps enable the extraction of segments using iterative clustering (see Section 5.3).

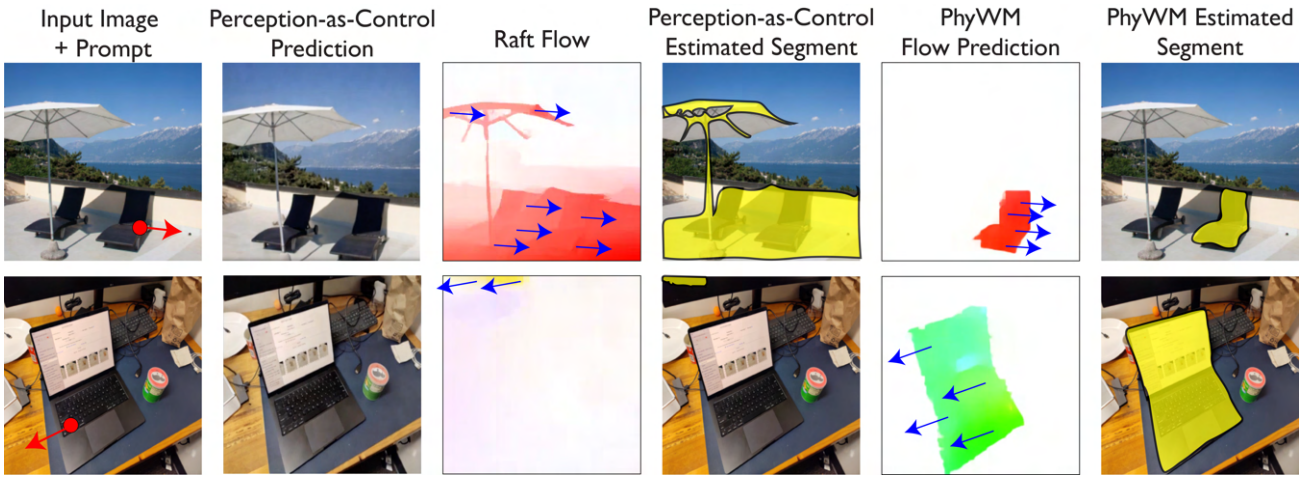

Figure 13. **Perception-as-Control [4] failure modes in complex scenes.** The first column shows the input image with prompt, the second column displays the motion-controlled prediction from the Perception-as-Control, and the third column shows the RAFT-predicted flow field between the input and predicted image. The final column presents the resulting segment obtained by thresholding the flow magnitude. Compared to **`PhyWM`**, Perception-as-Control often produces amorphous flow fields due to imprecise predictions based on the sparse motion control signals.

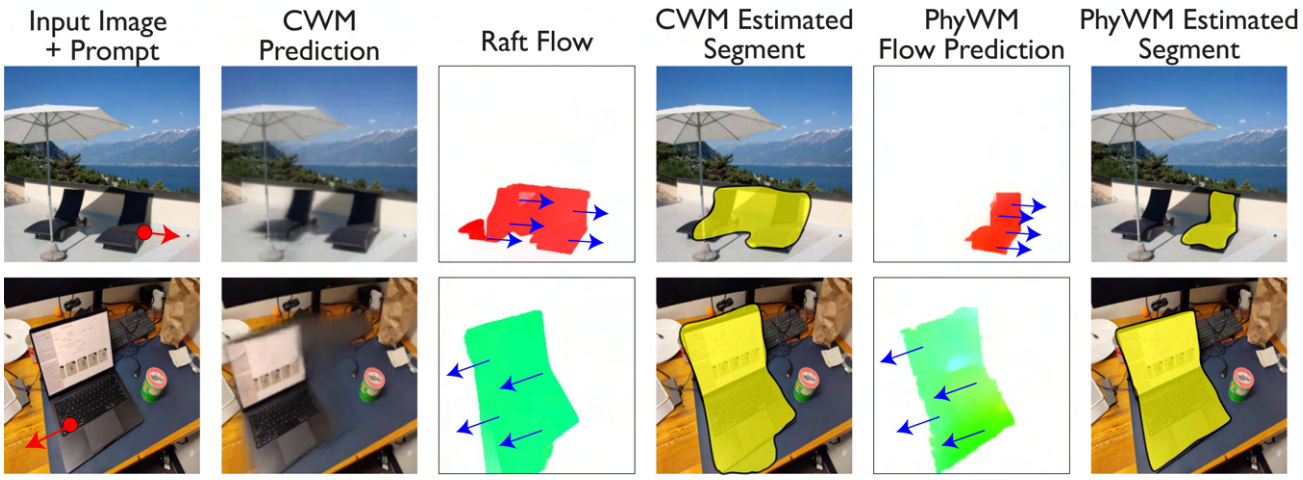

Figure 14. **CWM segmentation failure modes in complex scenes.** Each row shows a challenging example where CWM struggles. The first column shows the input image with the patch motion prompt (red arrow). The second column displays the counterfactual prediction generated by CWM. The third column shows the RAFT-predicted flow field between the input and counterfactual image. The final column presents the resulting segment obtained by thresholding the flow magnitude. Compared to **PhyWM**, CWM often produces diffuse motion fields due to blurry RGB reconstruction and inaccurate object boundaries.

References

[1] Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

[2] Stefan Andreas Baumann, Nick Stracke, Timy Phan, and Björn Ommer. What if: Understanding motion through sparse interactions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10286–10296, 2025.

[3] Daniel M. Bear, Kevin Feigelis, Honglin Chen, Wanhee Lee, Rahul Venkatesh, Klemen Kotar, Alex Durango, and Daniel L. K. Yamins. Unifying (machine) vision via counterfactual world modeling, 2023.

[4] Yingjie Chen, Yifang Men, Yuan Yao, Miaomiao Cui, and Liefeng Bo. Perception-as-control: Fine-grained controllable image animation with 3d-aware motion representation. *arXiv preprint arXiv:2501.05020*, 2025.

[5] Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, and et al. Open X-Embodiment: Robotic learning datasets and RT-X models. `https://arxiv.org/abs/2310.08864`, 2023.

[6] Abhimanyu Dubey, Abhinav Jauhri, and Abhinav Pandey et.al. The llama 3 herd of models. *ArXiv*, abs/2407.21783, 2024.

[7] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

[8] Nate Gillman, Charles Herrmann, Michael Freeman, Daksh Aggarwal, Evan Luo, Deqing Sun, and Chen Sun. Force prompting: Video generation models can learn and generalize physics-based control signals. *arXiv preprint arXiv:2505.19386*, 2025.

[9] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz MuellerFreitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "Something Something" Video Database for Learning and Evaluating Visual Common Sense. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5843–5851, 2017. Dataset: 20BN–Something–Something V2.

[10] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. Dataset: Kinetics-400.

[11] Wanhee Lee, Klemen Kotar, Rahul Mysore Venkatesh, Jared Watrous, Honglin Chen, Khai Loong Aw, and Daniel LK Yamins. 3d scene understanding through local random access sequence modeling. *arXiv preprint arXiv:2504.03875*, 2025.

[12] Dylan Li and Gyungin Shin. Promerge: Prompt and merge for unsupervised instance segmentation. In *European Conference on Computer Vision (ECCV)*, 2024.

[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[14] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: Vq-vae made simple. *arXiv preprint arXiv:2309.15505*, 2023.

[15] NeuroAILab. Spelkebench. `https://github.com/neuroailab/SpelkeBench`, 2024.

[16] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023.

[17] Lu Qi, Jason Kuen, Tiancheng Shen, Jiuxiang Gu, Weidong Guo, Jiaya Jia, Zhe Lin, and Ming-Hsuan Yang. High-quality entity segmentation. In *International Conference on Computer Vision (ICCV)*, 2023.

[18] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[19] 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 Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[20] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. Dataset: CO3D.

[21] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in neural information processing systems*, 2017.

[22] Rahul Venkatesh, Honglin Chen, Kevin Feigelis, Daniel M Bear, Khaled Jedoui, Klemen Kotar, Felix Binder, Wanhee Lee, Sherry Liu, Kevin A Smith, et al. Understanding physical dynamics with counterfactual world modeling. In *European Conference on Computer Vision*, pages 368–387. Springer, 2024.

[23] Xudong Wang, Rohit Girdhar, Stella X Yu, and Ishan Misra. Cut and learn for unsupervised object detection and instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3124–3134, 2023.

[24] Yihan Wang, Lahav Lipson, and Jia Deng. SEA-RAFT: Simple, efficient, accurate raft for optical flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

[25] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2, 2024.

[26] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. ScanNet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. Dataset: ScanNet++.

[27] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *ACM SIGGRAPH Conference Proceedings*, 2018. Dataset: RealEstate-10K.