Understanding Physical Dynamics with Counterfactual World Modeling
Supplementary Material

Rahul Venkatesh[1*], Honglin Chen[1*], Kevin Feigelis[1*], Daniel M. Bear[1], Khaled Jedoui[1], Klemen Kotar[1], Felix Binder[3], Wanhee Lee[1], Sherry Liu[1], Kevin A. Smith[2], Judith E. Fan[1], and Daniel L. K. Yamins[1]

[1] Stanford [2] MIT [3] UC San Diego

Table of Contents

1 CWM pre-training

1.1 Architecture details

Figure 1 provides an overview of the predictor architecture. The input video is first divided into non-overlapping spatiotemporal patches of size 8×8. Then a subset of patches is masked, and only the remaining visible patches are passed as inputs into the transformer encoder. We follow the standard ViT architecture. Following MAE [11], each transformer block in the ViT consists of a multi-head self-attention block and an MLP block, both having LayerNorm (LN). The CWM encoder and decoder have different widths, which are matched by a linear projection after the encoder [11]. Finally, the embedded tokens from the encoder and learnable mask tokens are passed as inputs into a shallow decoder to reconstruct the masked patches. Each spatiotemporal patch has a unique sine-cosine positional embedding. Position embeddings are added to both the encoder and decoder inputs. CWM does not use relative position or layer scaling [2, 11].

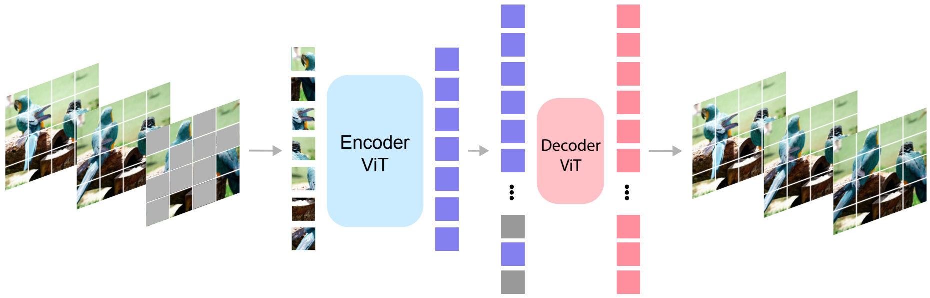


Fig. 1: Architecture of the masked predictor Ψ in the CWM framework.

1.2 Implementation details

While the discussion about CWM has for simplicity of presentation assumed a frame pair as input, for physical prediction problems it is natural to have an additional context frame to allow object initial velocities to be well-defined. More specifically, given 3 consecutive video frames at 150 ms apart during training, we provide full visibility to the first two context frames and only mask the last frame. In common situations where there is no motion in the first two frames, the three-frame model will recover what a two-frame model would have learned. When there is motion, the three-frame model will additionally learn acceleration, which is essential for physical predictions. For extracting keypoint and flow which only require 2 frames as input, we repeat the first frame twice so that the total input length is 3 frames. For extracting segmentation, we are given a single input frame, which we repeat twice and simulate object motions onto the third frame to compute segments.

CWM uses the standard ViT-B and ViT-L architectures with a patch size of 8, which allows structure extraction at a higher resolution. We pre-train CWM on the Kinetics-400 dataset [12], without requiring any specialized sparse operations or temporal downsampling. It takes approximately 6 days to train 1600 epochs on a TPU v4-256 pod.

1.3 Default settings

We show the default pre-training settings in Table 1. CWM does not use color jittering, drop path, or gradient clip. Following ViT's official code, xavier uniform is used to initialize all Transformer blocks. Learnable masked token is initialized as a zero tensor. Following MAE, we use the linear lr scaling rule: $lr = base\_lr \times batch\_size\,/\,256$ [11].

2 Structure extraction details and results

In this section, we discuss implementation details of the counterfactual queries for extracting keypoints, optical flow, and segmentations. We also provide more qualitative results of each structure extracted by CWM.

Table 1: Default pre-training setting of CWM

config	value
optimizer	AdamW [14]
base learning rate	1.5e-4
weight decay	0.05
optimizer momentum	$\beta_1, \beta_2 = 0.9, 0.95$ [6]
accumulative batch size	4096
learning rate schedule	cosine decay [13]
warmup epochs [10]	40
total epochs	1600
flip augmentation	no
augmentation	MultiScaleCrop [26]

2.1 Keypoint

Implementation details CWM queries keypoints iteratively, starting with an intervention initialized as an initial empty mask and adding visible tokens one-by-one. Note that, whereas the counterfactual queries for optical flow and segmentation involve perturbing the visual input to the predictor Ψ, keypoints arise by varying the prediction model's input mask.

At each iteration, we compute the Mean-Squared-Error (MSE) between the next-frame predictions of Ψ and the ground-truth next frame. We sort the MSE and select the top k locations as candidate keypoints. k is set as 4 by default. For each candidate keypoint, we add its patch content to the intervention and re-compute the MSE between the updated predictions of Ψ and the ground-truth next frame. The candidate keypoint with the minimum MSE error, or equivalently maximum error reduction, is selected as the keypoint output at that iteration. The selected keypoint is added to the intervention and we repeat the procedures above to compute the location of the next keypoint.

Additional qualitative results Figure 2 shows additional qualitative results of the keypoints extracted on DAVIS 2016 [19] and Bridge dataset [7]. We extract 5 keypoints for each example. Our procedure extracts dynamical RGB keypoints in the input frame pairs.

2.2 Optical flow

Implementation details As originally proposed in [4], we simultaneously estimate optical flow at all locations in a frame pair $x_1, x_2 \in \mathbb{R}^{3\times H\times W}$ via the Jacobian of Ψ, denoted as $\mathcal{J}\Psi \in \mathbb{R}^{H\times W\times H\times W}$. The Jacobian of Ψ assigns to element (i, j, k, l) the predictor's change in output at location (k, l) in the second frame due to an infinitessimal change at location (i, j) in the first frame. Flow can be computed using $\mathcal{J}\Psi$ as the following:

$$\mathbf{flow}(k,l) = \begin{cases} \mathbf{undefined}\ \text{(disocclusion)} & \text{if } \mathbf{argmax}_{i,j} |\mathcal{J}\Psi(i,j,k,l)| \ll 1 \\ (k,l) - \mathbf{argmax}_{i,j} \mathcal{J}\Psi(i,j,k,l) & \text{otherwise} \end{cases} \quad (1)$$

with results averaged over several choices of visible patches in the masked second frame x_2^β. This is a tensorial operation that enables parallel computation for flow at all pixel locations, implemented practically using Jacobian-vector products available in Pytorch autograd. Note that this method detects disocclusion rather than occlusion, since no perturbation at any location in the first frame will cause a response at a point that becomes disoccluded in the second frame.

Additional qualitative results We show additional qualitative results on two distinct datasets: DAVIS 2016 [20] and a recent synthetic dataset SPRING [15]. The results are shown in Figure 3 and 4 respectively.

2.3 Segmentation

Single Spelke object extraction To extract the Spelke object at a pixel location (i, j) of a static image, we first create an intervention that simulates counterfactual motion by taking the patch at location (i, j) and creating a new frame that is largely blank, but in which the content of the patch has been copied (e.g. translated) to a new location $(i+\epsilon_1, j+\epsilon_2)$, where ϵ_1, ϵ_2 are location offsets randomly sampled within a radius $r > 0$. We set r to a fixed fraction (0.2) of the input image size. In addition, we can optionally create the appearance of stopping the counterfactual motion at a location (i', j') by directly copying the patch at that location to the same location in the intervention without offset (or equivalently $r = 0$). Adding the stop-motion patch allows the counterfactual query to isolate a single object, especially in a cluttered scene with multiple objects adjacent to and stacked on top of one another. In practice, different random choices of motion offset and stop-motion patches could potentially yield different counterfactual motion results. We sample 4 different interventions per pixel location, compute flow for the counterfactual motion, average the flow magnitudes of the different samples, and then threshold the mean flow magnitude map at 0.5 to obtain the binary segmentation map at pixel location (i, j). We can use CWM to estimate optical flows, following procedures described in section 2.2. For faster extraction, we use RAFT [22] to estimate the optical flows of different samples.

We also find that the above procedure can be repeated iteratively to refine the segments further. Once a tentative segmentation map is obtained, we can sample more patches within the segment and add them to the intervention to simulate better counterfactual motion. At each iteration, we sample one patch within the segment and add it to the set of patches that simulates counterfactual motion. We additionally sample one patch outside the segment and add it to the set of stop-motion patches that simulates stopping the counterfactual motion. We set the number of iterations to 3 by default.

While the predictor Ψ is trained with input resolution 224×224, Spelke object can be extracted at a different resolution by simply interpolating the position encoding correspondingly. We extract zero-shot segmentations on COCO training images at resolution 480×480 using the ViT-B/8 CWM model.

Multiple Spelke object extraction To automatically discover multiple Spelke objects in a single image, we choose interventions at pixel locations based on a sampling probability distribution α, which has high probability at pixel locations belonging to Spelke objects and low otherwise. Once a segment is discovered, we mask out the probability distribution values using the segments and repeat the process to discover the next object.

We find two choices of sampling distribution work well. One is a movability distribution computed by sampling a few random interventions and averaging the motion responses across multiple predictions. The second choice is the prominence map computed by applying normalized cut to the patch-wise feature similarity matrix as proposed by CutLER [28]. In practice, the second approach yields slightly better qualitative segmentation results. Therefore, we choose the second approach as the default for computing the sampling probability distribution.

Distillation We follow the same procedure in the previous work CutLER [28] to distill segmentations extracted from a large task-agnostic pre-trained model into a smaller instance segmenter for faster and more robust segmentation. The extracted segmentations are used as pseudo annotations to train a downstream instance segmenter in a self-supervised manner.

Additional qualitative results Figure 5a and 5b show more qualitative segmentation results of Spelke objects extracted by CWM on COCO training images, and compare them to those of other baseline methods FreeSOLO [27] and CutLER [28]. In each image, we set the maximum number of Spelke objects to be extracted as 3. Figure 6a and 6b show unsupervised segmentation results from the distilled instance segmenter. We show the results on COCO validation images.

3 Dynamics understanding experiments

3.1 Physion benchmark

Dataset details As discussed in the main text, we use the Physion v1.5 benchmark to evaluate CWM and baseline models on physical dynamics understanding. Physion v1.5 has several key improvements over Physion v1, which are illustrated in Figure 7. More specifically, Physion v1.5 introduces another indoor environment, called the "craft room", in addition to the two environments featured in Physion v1 (see Figure 7d). Furthermore, v1.5 enhances the diversity of lighting conditions by employing a collection of 8 unique HDRI skyboxes, specifically designed to simulate various environmental lighting scenarios. This enhancement allows for dynamic time-of-day simulations in the room by adjusting the orientation of the skyboxes and directional lighting (see Figure 7a). Physion v1.5 also has improved rendering quality and photorealism in comparison to v1 (see Figure 7b and 7c). The physics simulations and rendering are done using the ThreeDWorld simulation platform [9].

Physion v1.5 comprises seven distinct physical scenarios, including collide, drop, dominoes, contain, roll, support, and link. This version comprehensively demonstrates various aspects and challenges of rigid body physics (see Figure 8 for examples of each scenario). We train and test the linear classifier model (as outlined in Section 4.1 of the main text) on all seven scenarios.

Per-scenario OCP and OCD results. In the main text, we presented the OCP and OCD scores averaged across all seven scenarios. Now, we provide a detailed breakdown of performance for each specific scenario, as shown in Table 4 and Table 5.

Qualitative comparisons on OCP and OCD task. We supplement our quantitative results on the Physion v1.5 tasks with qualitative visualizations in Figures 8, 9, 10 and 11. We show several example inputs for each scenario, along with the classification results of a linear probe on top of the CWM model, compared with those of leading baselines in each model category outlined in Table 1 in the main text.

Integrating vision structures for OCP and OCD tasks. In the main text, we demonstrate how zero-shot vision structures extracted from CWM improve OCP and OCD performance on Physion v1.5. This section describes the details of how these structures are integrated prior to linear probing on downstream tasks. Keypoint information is integrated by incorporating patch features at the keypoint locations. Optical flow information is integrated by providing 8×8 patches of optical flow value at the keypoint locations. Finally, segment information is integrated by using the segments to pool the feature map and incorporate the aggregated features for linear probing.

The integration process for segments is detailed as follows: Let $F \in \mathbb{R}^{H \times W \times D}$ be the feature map of an input frame from the last layer of the ViT encoder, where H, W, and D represent height, width, and channel dimension. Suppose we have binary segmentation masks S with dimension $N \times H \times W$, where N denotes the number of masks. We compute the set of aggregated feature vectors for each of the segments,

$$F_{agg} = \left\{ \frac{\sum_{i,j} S_{ij}^{n} F_{ij}}{\sum_{i,j} S_{ij}^{n}} \mid 1 \leq n \leq N \right\} \tag{2}$$

where S^n is the n-th segmentation mask and F_{ij} is the feature vector at location (i, j). These aggregated feature vectors, along with the keypoint patch

Table 2: GPT-4V performance on the Physion v1.5 tasks using two different promoting strategies: a) with RGB frames only and b) RGB frames with ground truth segment map overlayed on the objects of interest.

Prompting Method	OCP ↑	OCD ↑
RGB frames	52.9	54.7
RGB frames + GT segment overlay	**58.3**	**67.5**

features and flow patches, are concatenated with the original feature map F before being fed into the linear classifier.

3.2 GPT4-Vision prompting

GPT-4V prompting methodology and results. In Figures 12, 13, 14 and 15 we report a few results from testing GPT-4V on the OCP and OCD tasks in Physion v1.5. In each figure, the prompt image is shown on the left and the prompt text along with the GT label and response is shown on the right. To construct the image prompt, we tile four successive video frames (sampled at a frame gap of 150ms) into a 2×2 image with four panels. Each panel contains an RGB frame titled with its timestamp. Following the methodology used for evaluating vision-only models, the objects of interest for the contact-related queries are rendered with red and yellow textures to provide visual cues for the model. For OCP, the text prompt used was "These are 4 images taken sequentially from a video. If the video were to continue, would the red object touch the yellow surface? Explain your thinking and end with True or False only". For OCD, the prompt used was "These are 4 images taken sequentially from a video. Does the red object touch the yellow surface at any point in the video? Explain your thinking and end with True or False only". GPT4-V achieves 52.9% accuracy for OCP and 54.7% accuracy for OCD. For OCP, the model predicted "contact" 78.7% of the time, while for OCD the model predicted "no contact" 81.5% of the time.

Alternate querying methods. Additionally, we experiment with an alternative querying method to explore the limit of GPT4-V in dynamics understanding. In addition to rendering the objects of interest in red and yellow texture, we apply a bright red and yellow ground-truth segmentation overlay on them to focus the model's attention on these objects (see Figure 12 for visualizations). As shown in Table 2, we find that this querying strategy improves the OCP from 52.9% to 58.3% and the OCD from 54.7% to 67.5%. In the main text, we have also demonstrated a parallel phenomenon with CWM, where integrating segment information led to enhanced performance in related tasks (Table 3 in main text). These observations highlight the importance of vision structures such as segmentation in downstream tasks associated with physical dynamics understanding.

Table 3: Evaluation on additional benchmarks. (a) Activity recognition on Something-Something V2, (b) IntPhys intuitive physics benchmark.

Model	frames	accuracy ↑
VideoMAE	16	**54.7**
VideoMAE*	3	51.3
CWM	3	54.2

(a) Activity recognition on SSv2

Method	B1↓	B2↓	B3↓
VideoMAE	0.40	0.23	0.30
VideoMAE*	0.46	0.30	0.30
VideoMAEv2	0.36	0.30	0.36
CWM	**0.36**	**0.20**	**0.26**

(b) IntPhys

Implementation details. We employ four GPT-4V accounts and retrieve the results using Selenium [8], a browser automation tool. Our methodology adheres to the code framework available at[1].

4 Evaluating CWM on additional benchmarks

4.1 Activity Recognition

We evaluate CWM on the Something-Something V2 benchmark for activity recognition and report the results in Table 3a. To obtain model predictions we train an attentive probe on the feature representation similar to the setup used in V-JEPA [3]. We find that CWM outperforms VideoMAE trained with the same number of input frames (i.e VideoMAE*) by a fair margin and is comparable to the standard VideoMAE trained on a longer context window of 16 frames.

4.2 IntPhys

In the main text we show evaluations on Physion as it is by far the most challenging and comprehensive benchmark in the literature, consisting of realistic 3D simulations from diverse physical scenarios. Here, we evaluate on IntPhys [21] which is a complementary benchmark to Physion with photorealistic simulations of various intuitive physics tasks. Table 3b reports the relative error metric in IntPhys [21] evaluation on the validation split. For all models, we use the future frame reconstruction error to obtain a plausibility score for a given video. The evaluation comprises of three tasks: B1 tests for object permanence, B2 tests for shape constancy, and B3 tests for spatio-temporal continuity. We find that CWM, despite trained with less number of frames, outperforms VideoMAE. CWM (86M) outperforms VideoMAEv2 (1.1B) despite having fewer parameters. For a fair comparison, we further evaluate VideoMAE* which is trained on the same number of frames as CWM but with tube masking instead of temporally factored masking. The performance gain from CWM further validates the benefit of the temporally-factored masking policy.

[1] https://github.com/Michelangelo27/chatgpt_selenium_automation

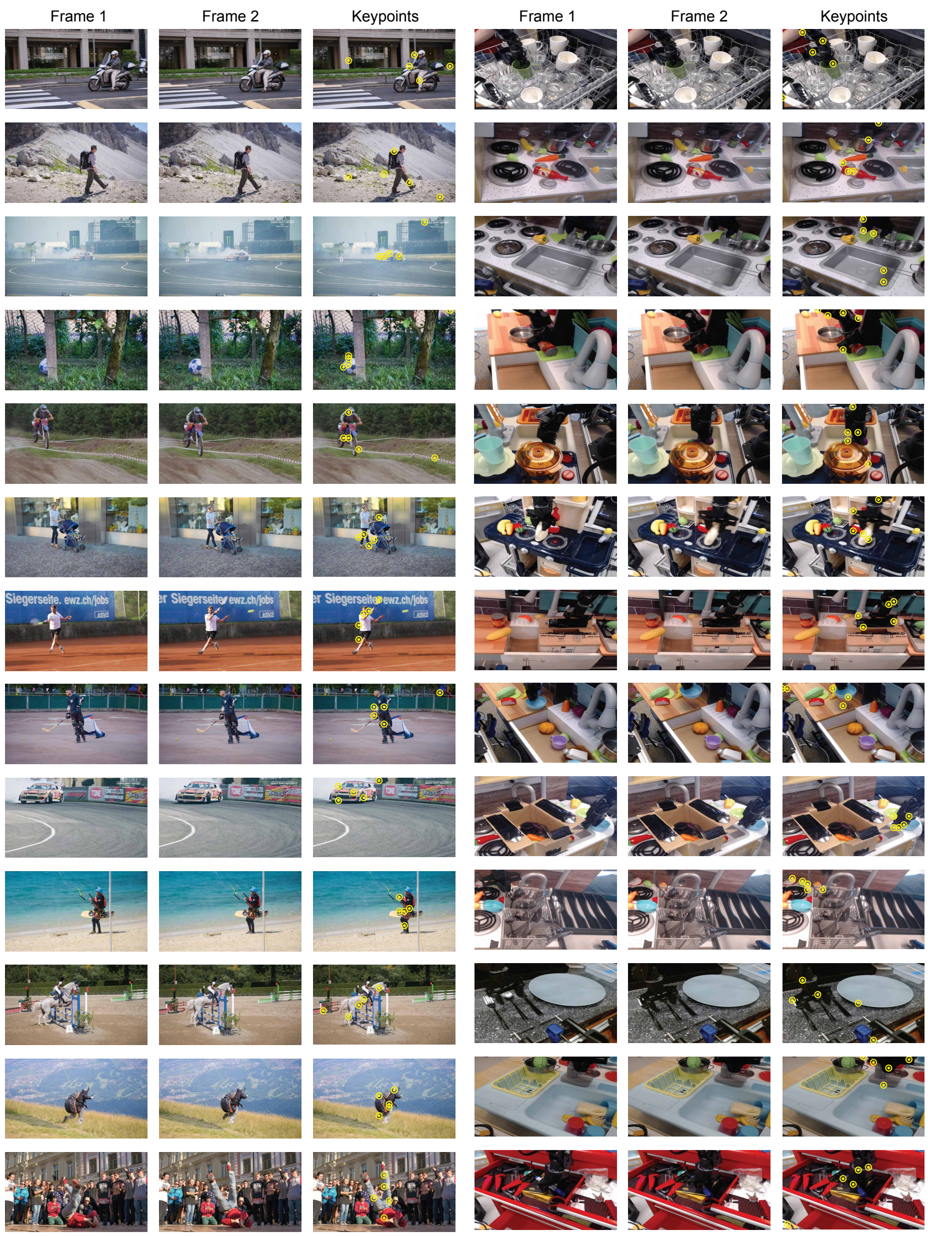


Fig. 2: Keypoints extracted by CWM. We extract 5 keypoints for each example on DAVIS 2016 (left) and Brige dataset (right). 3

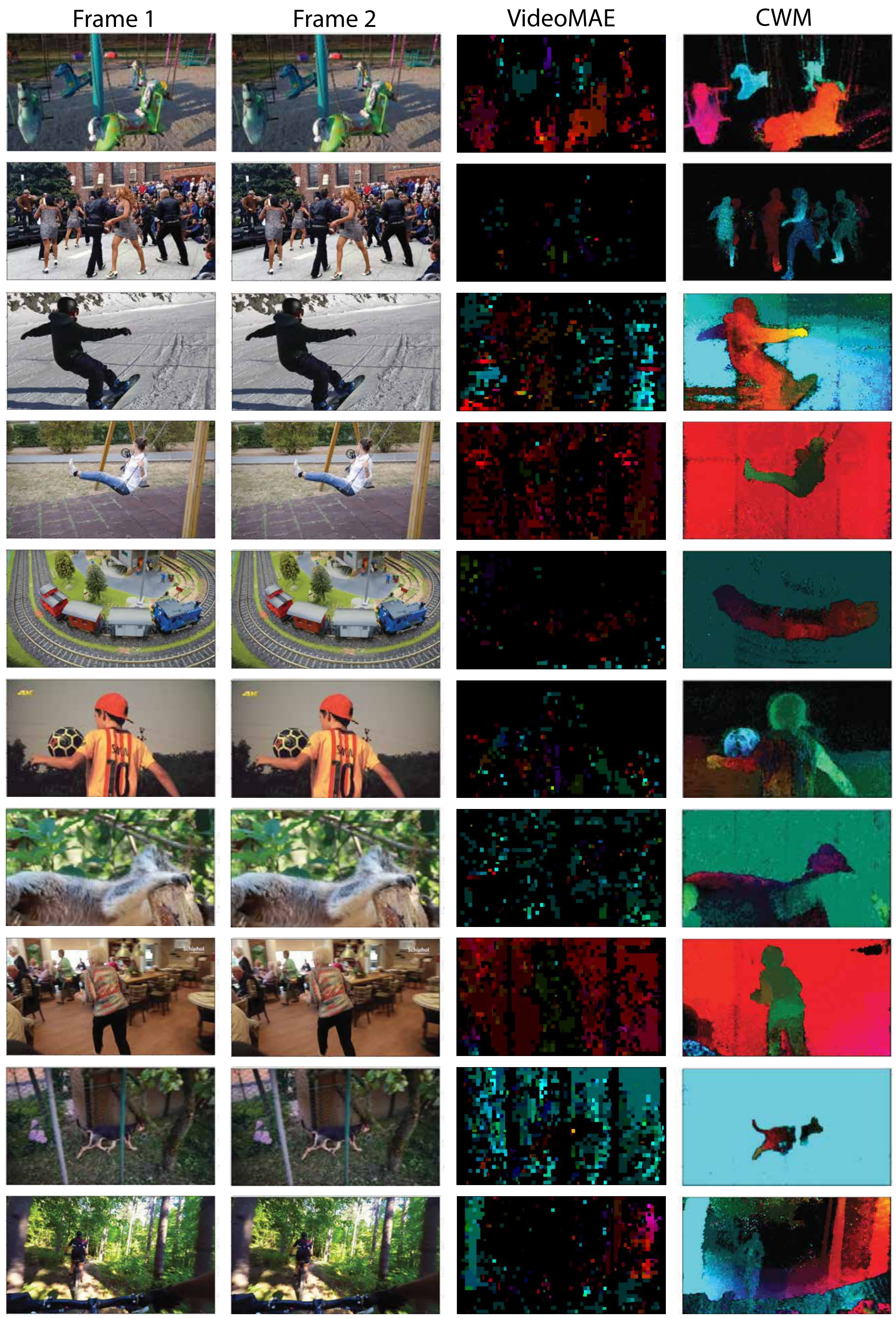


Fig. 3: Additional Optical Flows extracted on the DAVIS dataset. We apply our flow extraction procedure to both CWM and VideoMAE predictors, and compare the extracted flows. 4

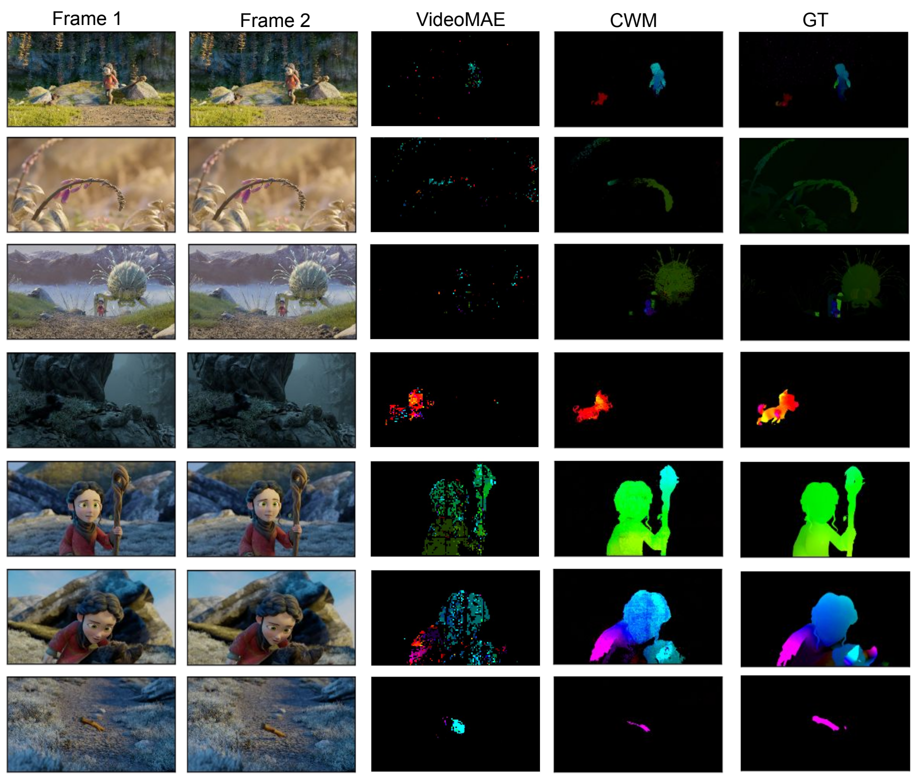


Fig. 4: Additional Optical Flows extracted on the Spring dataset. 4

Fig. 5a: Pseudo-masks extracted in a zero-shot manner on COCO training images. FreeSOLO extracts dense masks and removes redundancy via mask non-maximum-suppression (NMS). CutLER and CWM extracts at most 3 masks per image. The pseudo-masks are used as self-supervision signals for training downstream detectors. 5

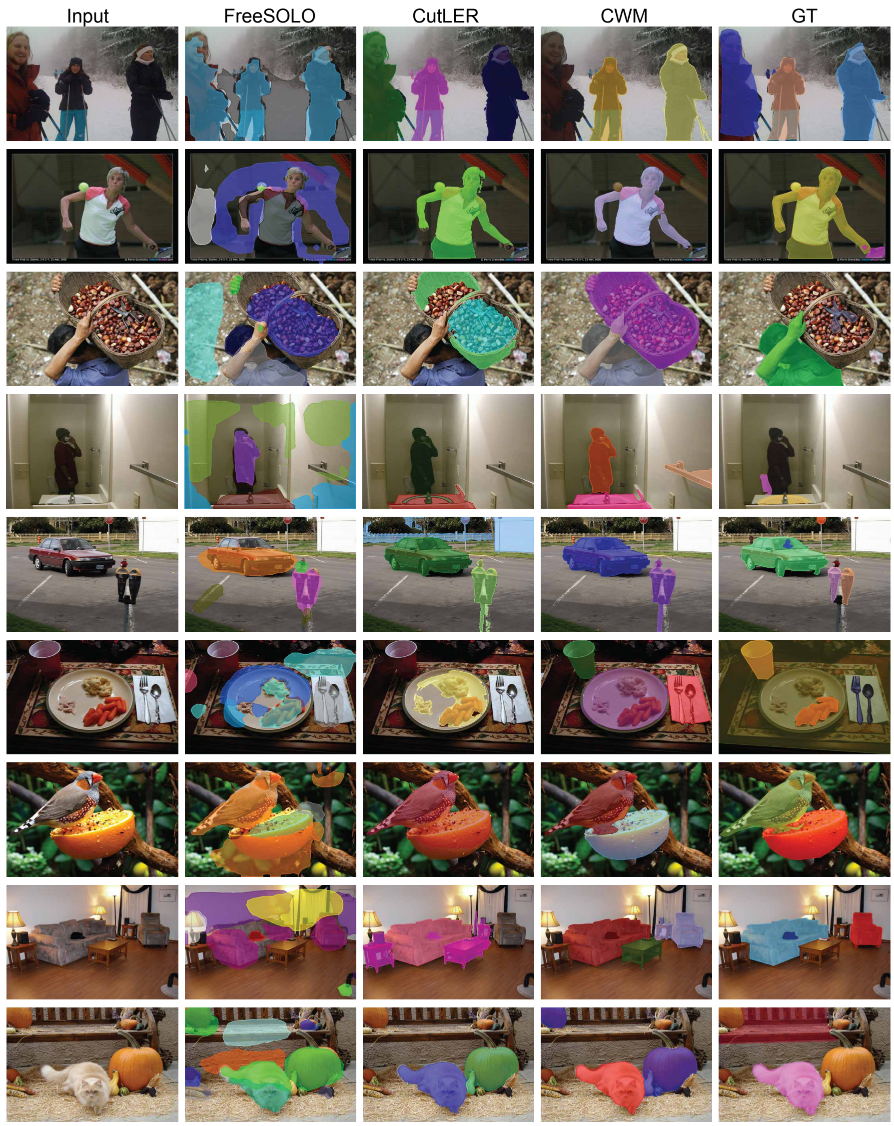


Fig. 5b: More qualitative results on pseudo-masks extracted in a zero-shot manner on COCO training images. 5

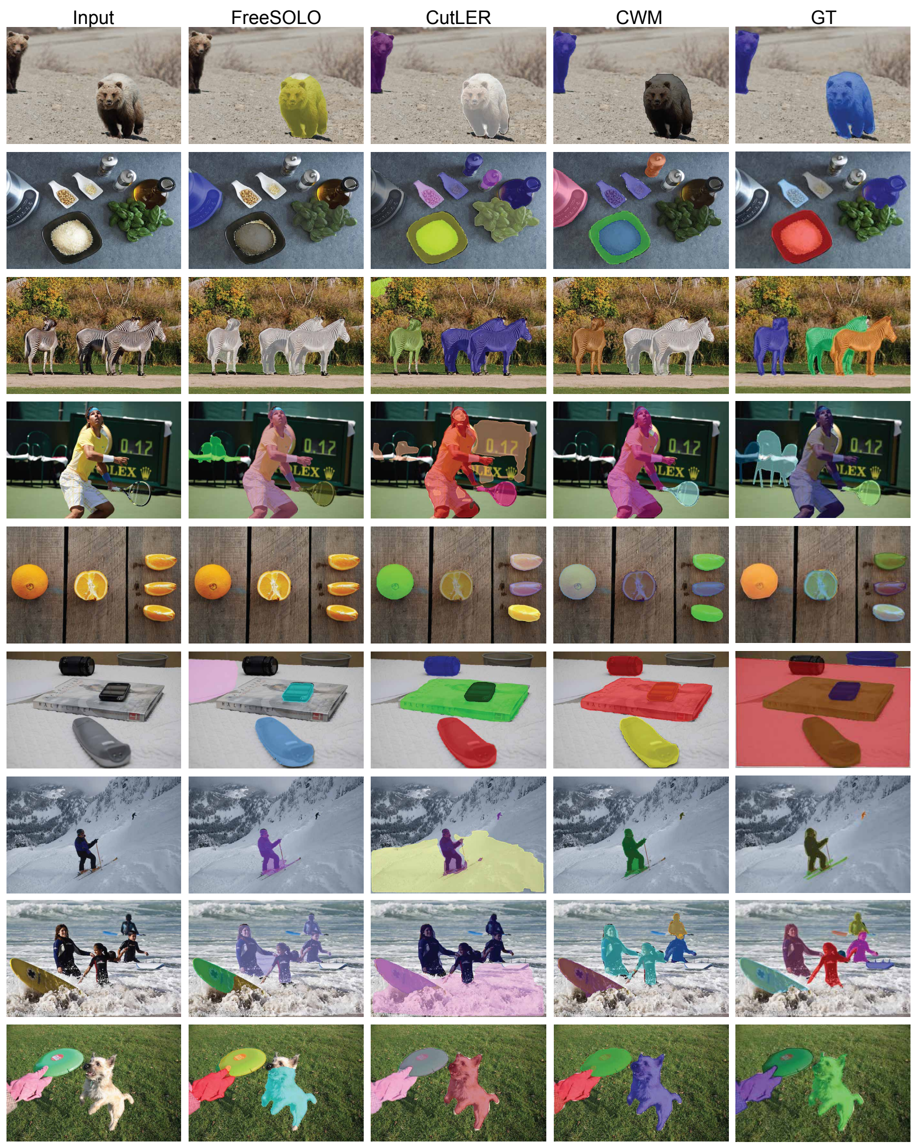
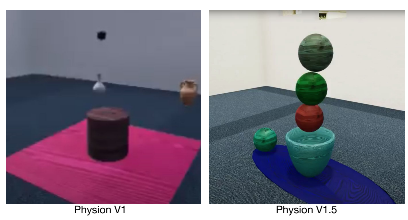


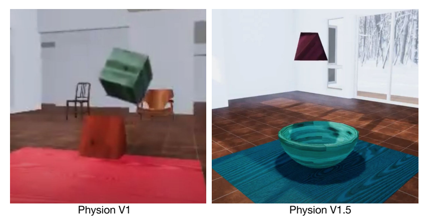
Fig. 6a: Unsupervised segmentation results from the distilled instance segmenter. We show the results on COCO validation images. 5

Fig. 6b: More unsupervised segmentation results from the distilled instance segmenter. 5

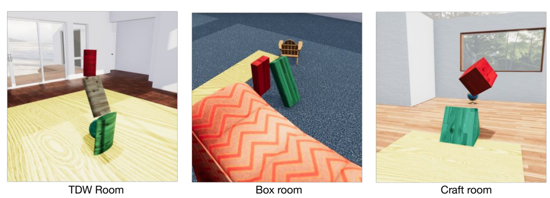
(a) Diverse lighting conditions



(b) Improved visual realism: high resolution rendering



(c) Improved visual realism: rendering better shadows



(d) New indoor environments

Fig. 7: Physion v1.5 features key improvements over v1 such as a) enhanced diversity of lighting conditions by employing HDRI skyboxes, b) higher resolution rendering, c) improved rendering of shadows and c) an additional indoor environment ("Craft room"). 5

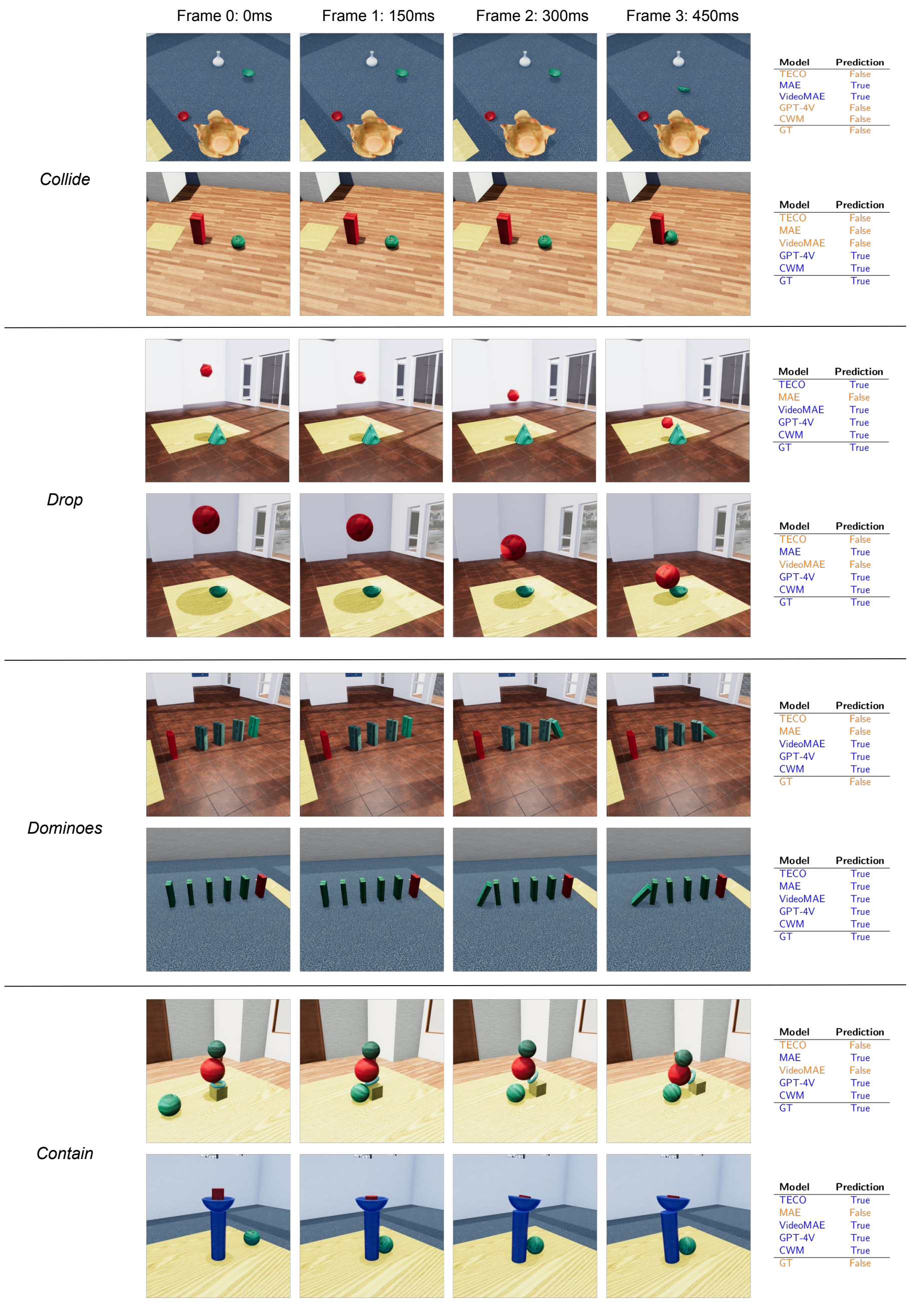


Fig. 8: Model predictions on the OCP task. The input frames sampled at a frame gap of 150ms are shown on the left and the model predictions are shown on the right. We compare against the best performing models in each model category outlined in Table 4. 5

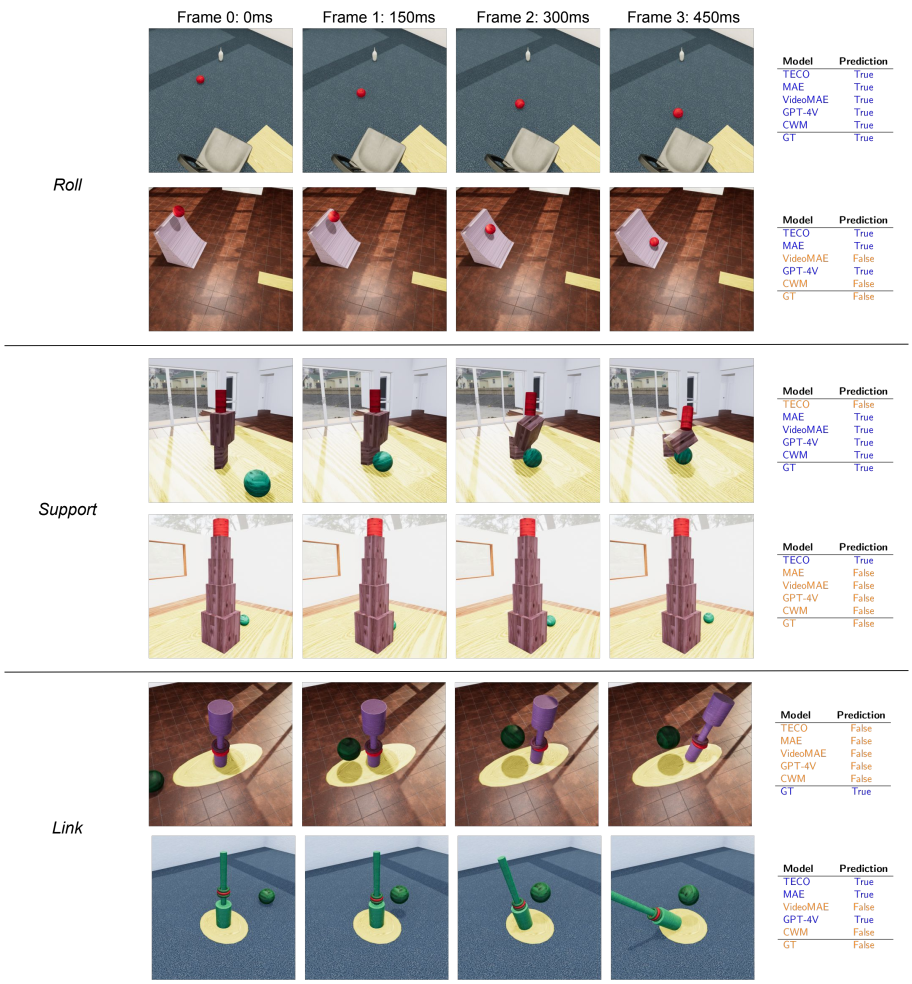


Fig. 9: More model predictions on the OCP task. 5

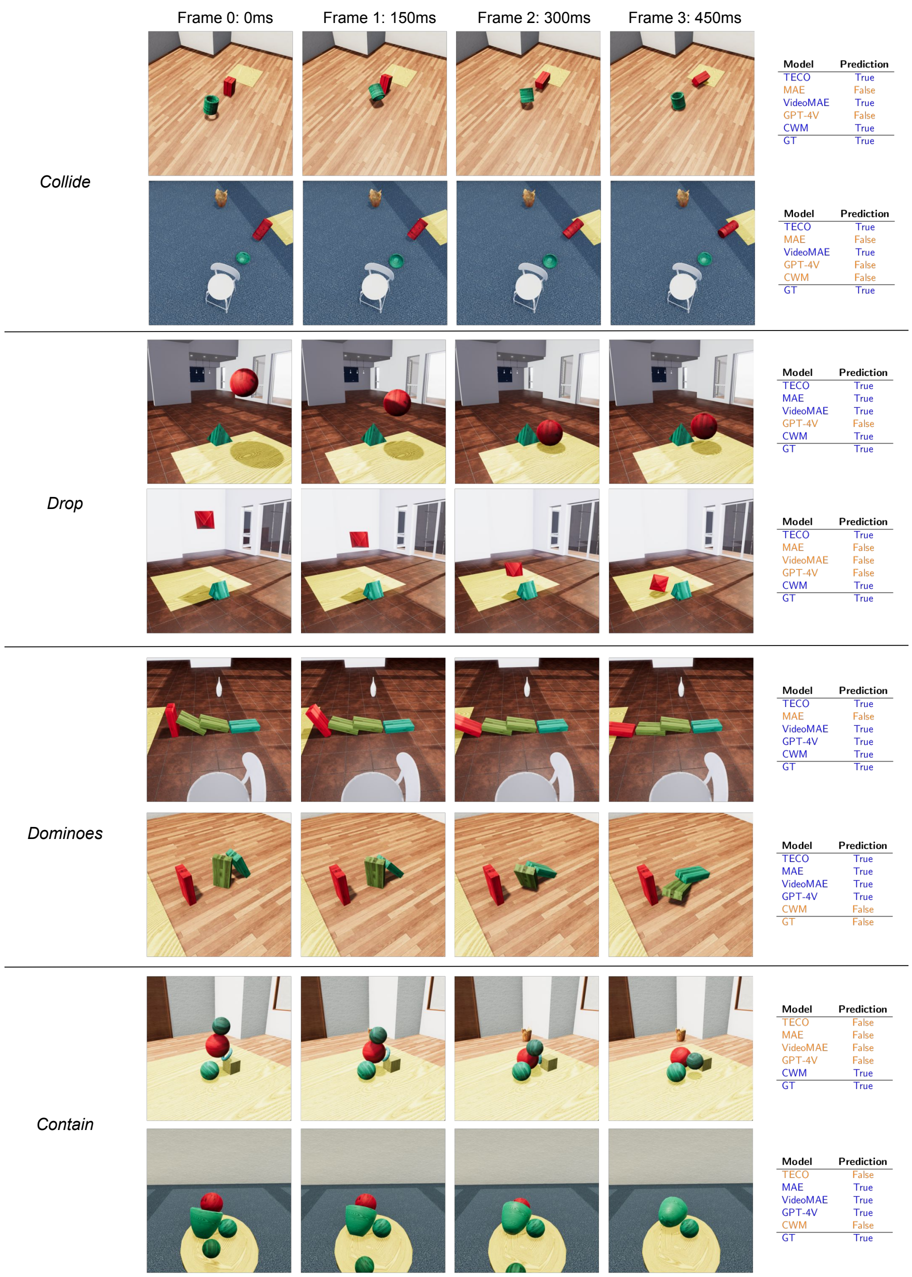


Fig. 10: Model predictions on the OCD task. The input frames sampled at a frame gap of 150ms are shown on the left and the model predictions are shown on the right. We compare against the best performing models in each model category outlined in Table 5. 5

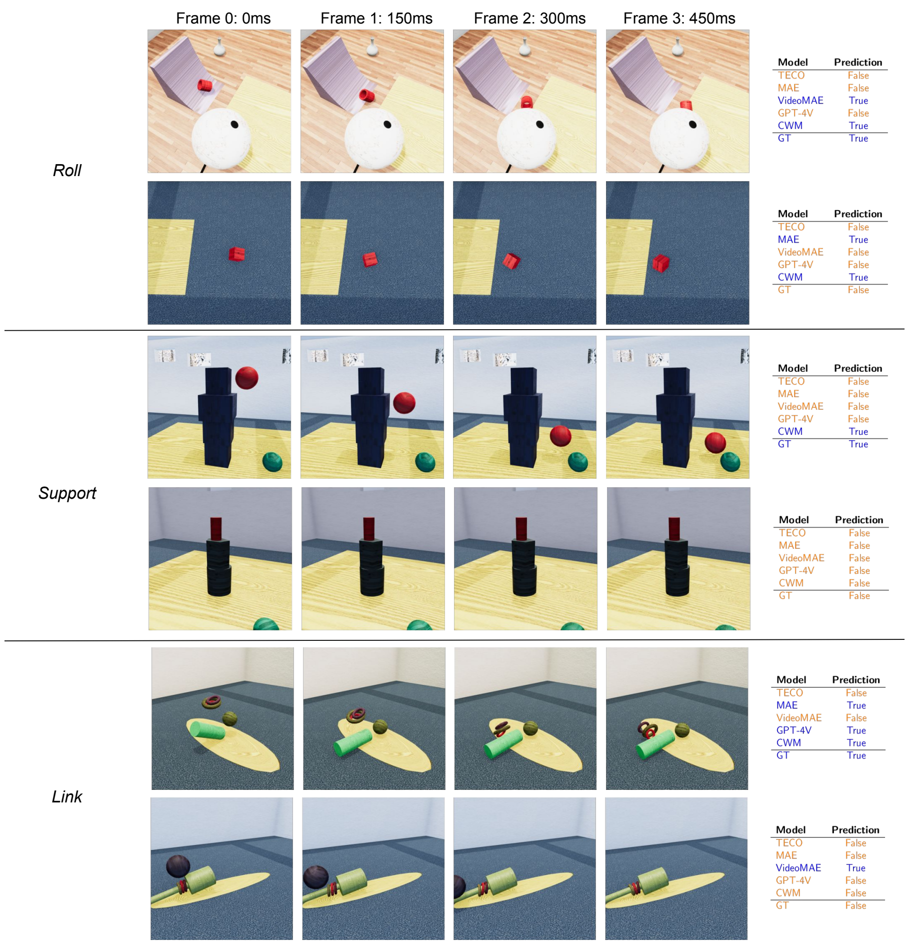


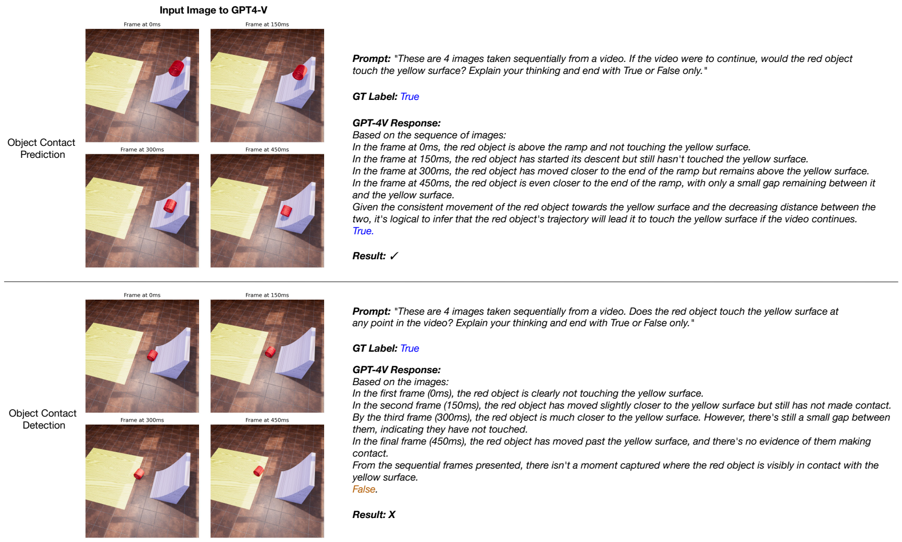
Fig. 11: More model predictions on the OCD task. 5

Table 4: Physion v1.5 scenario-wise OCP results. We compare CWM to four classes of baseline methods across different architectures on the OCP task. In the main text, we presented the scores averaged across all seven scenarios. Here we provide a detailed breakdown of performance for each specific scenario. For a strictly fair comparison we train VideoMAE* with the same patch size and number of frames as CWM. 5

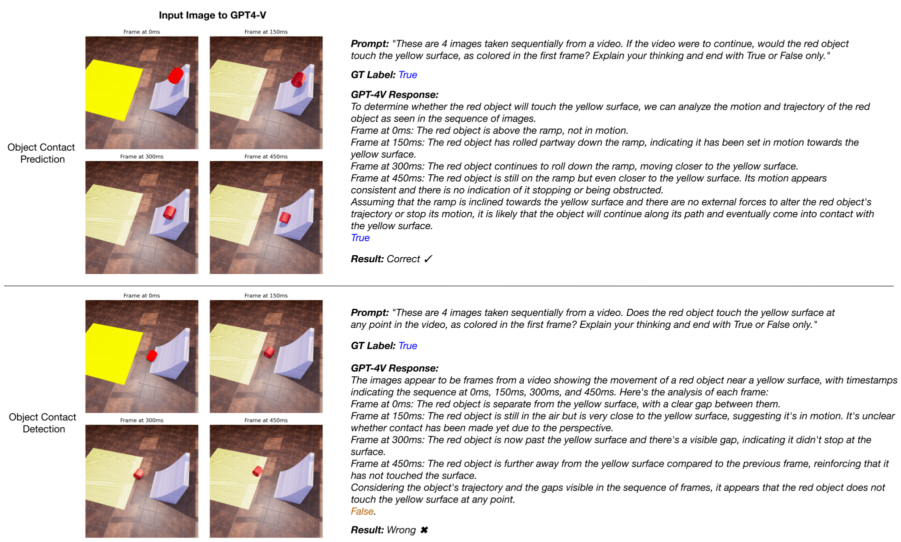
method	arch	OCP by scenario ↑							Avg. OCP ↑
		collide	drop	support	link	roll	contain	dominoes	
video prediction models									
MCVD [24]	UNet	73.3	65.0	70.7	59.2	51.0	66.7	57.9	63.4
R3M [16]	Res50	79.0	61.8	70.1	66.9	**66.2**	62.1	67.3	67.6
FitVid [1]	VAE	65.0	68.2	71.5	59.9	54.1	60.8	70.7	64.3
TECO [29]	vq-gan	75.9	70.6	74.4	65.2	59.1	72.4	67.5	69.3
self-supervised image representation models									
DINO [5]	ViT-B	79.0	72.6	81.0	**72.0**	61.8	69.3	69.2	72.1
DINOv2 [18]	ViT-B	78.1	70.7	80.3	70.7	64.3	70.6	70.4	72.2
DINOv2 [18]	ViT-L	81.0	68.8	82.3	68.8	61.1	69.9	73.6	72.2
DINOv2 [18]	ViT-g	80.0	74.5	81.0	66.2	61.8	74.5	71.1	72.7
MAE [11]	ViT-B	80.0	72.6	78.9	70.1	64.3	68.0	74.2	72.6
MAE [11]	ViT-L	80.0	73.2	81.0	69.4	58.0	69.3	70.4	71.6
MAE [11]	ViT-H	**83.8**	72.0	**84.4**	70.1	61.8	69.3	71.7	73.3
MAE [11]	ViT-B	81.9	70.7	83.0	68.8	59.9	67.3	73.0	72.1
MAE [11]	ViT-L	83.8	70.7	81.0	68.2	59.9	70.6	74.2	72.6
self-supervised video representation models									
VMAE [23]	ViT-B	74.3	74.5	83.0	65.6	61.8	71.2	74.2	72.1
VMAE* [23]	ViT-B	80.0	71.3	82.3	70.1	58.6	72.5	76.7	73.2
VMAE [23]	ViT-L	79.0	73.9	82.3	66.9	65.0	72.5	75.5	73.6
VMAE [23]	ViT-H	81.0	73.2	81.6	70.7	63.1	70.6	74.2	73.5
VMAEv2 [25]	ViT-g	77.1	75.2	83.0	65.0	62.4	70.6	72.3	72.2
V-JEPA [23]	ViT-L	80.1	68.8	84.3	69.4	62.4	73.8	74.2	73.4
vision-language models									
GPT4-V [17]	-	52.7	46.5	58.5	54.8	56.2	56.1	46.5	52.9
CWM	ViT-B	82.9	75.2	83.7	70.7	63.7	**77.8**	77.4	75.9
CWM	ViT-L	**83.8**	74.5	**84.4**	71.3	65.0	75.8	**78.0**	**76.1**

Table 5: Physion v1.5 scenario-wise OCD results. We compare CWM to four classes of baseline methods across different architectures on the OCD task. In the main text, we presented the scores averaged across all seven scenarios. Here we provide a detailed breakdown of performance for each specific scenario. For a strictly fair comparison we train VideoMAE* with the same patch size and number of frames as CWM. 5

method	arch	OCD by scenario ↑							Avg. OCD ↑
		collide	drop	support	link	roll	contain	dominoes	
video prediction models									
MCVD [24]	UNet	82.9	74.5	95.9	75.8	68.8	77.8	89.9	80.8
R3M [16]	Res50	83.8	72.0	90.5	72.0	72.0	70.6	86.2	78.1
FitVid [1]	VAE	58.9	56.7	63.1	63.2	60.2	55.5	58.8	59.5
TECO [29]	vq-gan	87.0	77.5	87.5	70.7	72.6	76.3	95.0	80.9
self-supervised image representation models									
DINO [5]	ViT-B	87.6	79.6	95.2	81.5	76.4	80.4	**96.9**	85.4
DINOv2 [18]	ViT-B	89.5	**84.7**	**96.6**	**86.6**	76.4	79.1	**96.9**	87.1
DINOv2 [18]	ViT-L	91.4	79.0	**96.6**	84.1	73.9	77.8	95.6	85.5
DINOv2 [18]	ViT-g	91.4	80.3	95.2	83.4	70.1	77.1	94.3	84.6
MAE [11]	ViT-B	86.7	76.4	92.5	77.7	71.3	72.5	93.7	81.6
MAE [11]	ViT-L	86.7	75.8	93.9	80.3	70.7	73.2	95.6	82.3
MAE [11]	ViT-H	84.8	75.2	92.5	76.4	67.5	73.2	96.2	80.8
MAE [11]	ViT-B	86.7	75.8	91.2	76.4	70.7	74.5	**96.9**	81.7
MAE [11]	ViT-L	89.5	76.4	93.2	77.1	69.4	72.5	95.0	81.9
self-supervised video representation models									
VMAE [23]	ViT-B	91.4	78.3	94.6	80.3	76.4	83.0	95.6	85.7
VMAE* [23]	ViT-B	93.3	79.6	94.6	82.2	75.2	82.4	96.2	86.2
VMAE [23]	ViT-L	95.2	78.3	95.2	82.8	75.2	80.4	95.6	86.1
VMAE [23]	ViT-H	95.2	79.6	95.2	84.7	79.0	81.7	**96.9**	87.5
VMAEv2 [25]	ViT-g	91.4	78.3	91.8	84.7	73.9	81.7	93.1	85.0
V-JEPA [23]	ViT-L	93.3	84.7	95.9	83.4	73.2	83.0	95.6	87.0
vision-language models									
GPT4-V [17]	-	52.9	53.0	54.2	60.7	56.2	56.1	49.7	54.7
CWM	ViT-B	96.2	82.2	95.9	85.4	81.5	**86.3**	96.2	**89.1**
CWM	ViT-L	**96.2**	83.4	**96.6**	84.1	81.5	83.0	96.2	88.7



(a) Querying method: RGB frames only

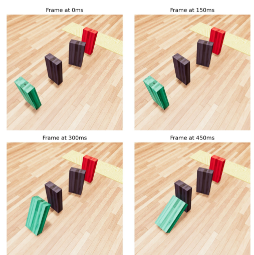


(b) Querying method: RGB frames with objects of interest highlighted

Fig. 12: GPT-4V results on the OCP and OCD tasks in Physion v1.5. The input image prompt is shown on the left and the text prompt, GPT-4V response and results are shown on the right. We show the results of two different querying strategies: a) with RGB frames only and b) RGB frames with ground truth segment map overlayed on the objects of interest. 7

Input Image to GPT4-V

Object Contact Prediction



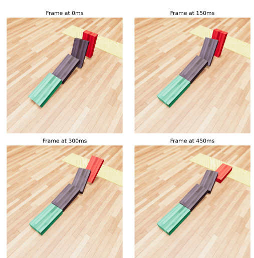
Prompt: *"These are 4 images taken sequentially from a video. If the video were to continue, would the red object touch the yellow surface? Explain your thinking and end with True or False only."*

GT Label: *True*

GPT-4V Response:
Analyzing the images, we can see the following:
Frame at 0ms: The green and red objects are standing upright, and the green object is closer to the yellow surface.
Frame at 150ms: The green object has started to fall towards the yellow surface.
Frame at 300ms: The green object is almost touching the yellow surface, and the red object has tilted slightly but hasn't begun falling yet.
Frame at 450ms: The green object is in contact with the yellow surface, and the red object is tilting further but is still relatively upright.
Based on the observed progression in these frames, the green object falls relatively quickly and touches the yellow surface in 450ms. The red object, however, is tilting at a much slower rate than the green object was. Given the distance between the red object and the yellow surface in the 450ms frame, and comparing it to the rate at which the green object fell, it seems unlikely that the red object would touch the yellow surface in the next immediate frames, especially if the time intervals remain consistent.
False.

Result: *Wrong* ✖

Object Contact Detection



Prompt: *"These are 4 images taken sequentially from a video. Does the red object touch the yellow surface at any point in the video? Explain your thinking and end with True or False only."*

GT Label: *True*

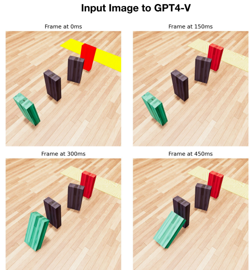
GPT-4V Response:
To determine if the red object touches the yellow surface at any point in the video, let's observe the sequence of events as shown in the four images:
Frame at 0ms: The red object is upright and not touching the yellow surface.
Frame at 150ms: The red object has fallen onto the brown object, still not touching the yellow surface.
Frame at 300ms: The red object is now leaning further on the brown object and is closer to the yellow surface but not yet touching it.
Frame at 450ms: The red object continues to lean and is almost touching or very close to the yellow surface. However, from this angle and without further frames, we cannot definitively say whether the red object has made contact with the yellow surface.
Since the images provided do not conclusively show the red object touching the yellow surface, and without assuming any motion beyond the last frame provided, the answer is:
False.

Result: *Wrong* ✖

(a) Querying method: RGB frames only

Input Image to GPT4-V

Object Contact Prediction



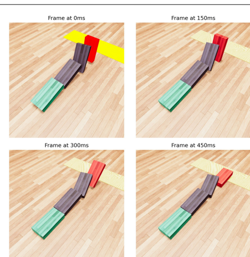
Prompt: *"These are 4 images taken sequentially from a video. If the video were to continue, would the red object touch the yellow surface, as colored in the first frame? Explain your thinking and end with True or False only."*

GT Label: *True*

GPT-4V Response:
To determine if the red object will touch the yellow surface as colored in the first frame, let's analyze the motion of the objects in the frames provided:
Frame at 0ms: The red object is upright and is the closest to the yellow surface.
Frame at 150ms: The red object has begun to fall towards the yellow surface.
Frame at 300ms: The red object is falling further and is now leaning at a greater angle towards the yellow surface.
Frame at 450ms: The red object is almost horizontal and appears to continue its motion towards the yellow surface.
Based on the sequence of the frames, the red object is falling in the direction of the yellow surface with a consistent motion. There is no indication that its path is being obstructed or that it is being redirected away from the yellow surface. Unless an external force acts upon it, the red object will continue in its trajectory and is likely to make contact with the yellow surface.
True

Result: *Correct* ✓

Object Contact Detection



Prompt: *"These are 4 images taken sequentially from a video. Does the red object touch the yellow surface at any point in the video, as colored in the first frame? Explain your thinking and end with True or False only."*

GT Label: *True*

GPT-4V Response:
The four images show a sequence of frames taken at different times: 0ms, 150ms, 300ms, and 450ms. The sequence depicts a red object in motion towards a yellow surface. In the first frame (0ms), the red object is clearly not touching the yellow surface. As the sequence progresses, the red object moves closer to the yellow surface.
In the frame at 150ms, the red object is closer but still not in contact with the yellow surface. By the frame at 300ms, the red object is shown overlapping with the yellow surface, indicating that contact has been made. Finally, the frame at 450ms confirms the red object is resting partly on the yellow surface.
Since we can observe the red object making contact and resting on the yellow surface in the sequence of images, we can conclude that the red object does indeed touch the yellow surface at some point in the video.
True.

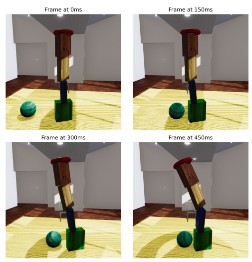
Result: *Correct* ✓

(b) Querying method: RGB frames with objects of interest highlighted

Fig. 13: More GPT-4V results. 7

Input Image to GPT4-V

Object Contact Prediction



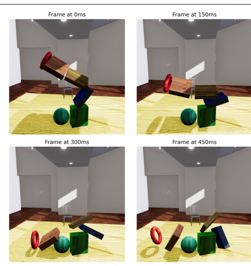
Prompt: *"These are 4 images taken sequentially from a video. If the video were to continue, would the red object touch the yellow surface? Explain your thinking and end with True or False only."*

GT Label: *True*

GPT-4V Response:
Based on the provided images, the red object, which appears to be the top of a hammer-like tool, is moving downward towards the yellow surface.
In the "Frame at 0ms", the hammer is upright, and there's a notable distance between the red object and the yellow surface. By "Frame at 150ms", the hammer starts tilting and continues its descent. In "Frame at 300ms", the hammer has tilted even further, and the distance between the red object and the yellow surface has decreased significantly. By "Frame at 450ms", the hammer is very close to the yellow surface and shows significant angular momentum, suggesting a continuation of the movement.
Given the consistent movement observed over the four frames and the progressive decrease in the distance between the red object and the yellow surface, it's reasonable to infer that the red object will continue its trajectory and make contact with the yellow surface if the video were to continue.
True.

Result: *Correct* ✓

Object Contact Detection



Prompt: *"These are 4 images taken sequentially from a video. Does the red object touch the yellow surface at any point in the video? Explain your thinking and end with True or False only."*

GT Label: *True*

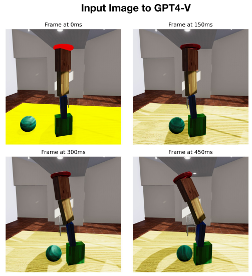
GPT-4V Response:
The four images depict a sequence of events where a structure made of colored blocks is collapsing.
In the first frame (0ms), the structure is beginning to topple over. The red object, which appears to be a red ring, is at the top of the structure, firmly in contact with the brown block beneath it.
In the second frame (150ms), the structure has continued to fall, and the red ring is now in free fall, separated from the brown block. There's visible space between the red ring and all other objects, including the yellow surface.
The third frame (300ms) shows the red ring further down, and it's now clear that it is falling straight downward without tilting towards the yellow surface.
By the fourth frame (450ms), the red ring has nearly reached the ground and is clearly going to land on the green surface.
The path of its descent and its orientation suggest it will not make contact with the yellow surface at any point.
Therefore, based on the provided frames, the red object does not touch the yellow surface at any point in the video.
False.

Result: *Wrong* ✖

(a) Querying method: RGB frames only

Input Image to GPT4-V

Object Contact Prediction



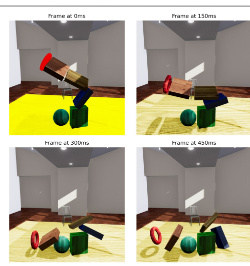
Prompt: *"These are 4 images taken sequentially from a video. If the video were to continue, would the red object touch the yellow surface, as colored in the first frame? Explain your thinking and end with True or False only."*

GT Label: *True*

GPT-4V Response:
The sequence of images provided shows a red-topped object (presumably a hammer) in different positions at various time frames (0 ms, 150 ms, 300 ms, and 450 ms). The object appears to be falling over, pivoting on the green base. It's clear that the motion is a rotation around the point where the hammer touches the green block.
With each successive frame, the angle of the hammer's handle with respect to the vertical is increasing, indicating it is falling towards the yellow surface. Given this consistent motion over the time frames provided, it is reasonable to predict that if this motion continues without any external forces acting on it (like a sudden stop or reverse in the direction of motion), the red object would indeed come into contact with the yellow surface.
So, based on the information provided and assuming no other forces come into play:
True.

Result: *Correct* ✓

Object Contact Detection



Prompt: *"These are 4 images taken sequentially from a video. Does the red object touch the yellow surface at any point in the video, as colored in the first frame? Explain your thinking and end with True or False only."*

GT Label: *True*

GPT-4V Response:
To determine if the red object touches the yellow surface at any point in the video, we can examine the sequence of frames:
Frame at 0ms: The red object is clearly above the yellow surface, not touching it.
Frame at 150ms: The red object appears to be in contact with the yellow object, which is in turn resting on the yellow surface. However, it is not clear if the red object itself is touching the yellow surface directly or if it is only touching the yellow object.
Frame at 300ms: The red object has fallen further and is now below the top level of the yellow object, indicating that if it was not directly touching the yellow surface before, it is now likely doing so.
Frame at 450ms: The red object has clearly fallen onto the yellow surface.
From this analysis, we can infer that the red object does indeed touch the yellow surface at some point between the frames at 150ms and 300ms, most likely when it falls past the level of the yellow object.
True

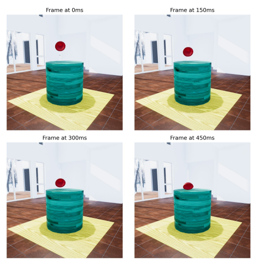
Result: *Correct* ✓

(b) Querying method: RGB frames with objects of interest highlighted

Fig. 14: More GPT-4V results. 7

Input Image to GPT4-V

Object Contact Prediction



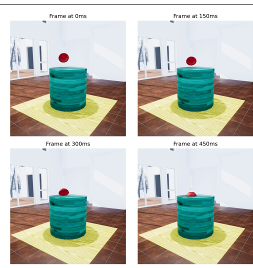
Prompt: *"These are 4 images taken sequentially from a video. If the video were to continue, would the red object touch the yellow surface? Explain your thinking and end with True or False only."*

GT Label: *False*

GPT-4V Response:
Frame at 0ms: The red object is above the blue cylinder.
Frame at 150ms: The red object has come down a bit, and it is closer to the blue cylinder than before.
Frame at 300ms: The red object is even closer to the blue cylinder.
Frame at 450ms: The red object is very close to the top of the blue cylinder, but it hasn't touched it yet.
From the frames provided, we can observe that the red object is consistently moving downwards towards the blue cylinder. It's difficult to determine the exact rate of descent just from the still images, but considering the movement pattern in the time intervals shown, it seems that the red object is likely to touch the blue cylinder soon after the 450ms frame. However, whether it touches the yellow surface after that would depend on whether it bounces back upon hitting the blue cylinder or continues its descent.
Given the data available, it's more probable that it will touch the blue cylinder and depending on its velocity and the nature of the red object and the blue cylinder (whether they are hard, soft, bouncy, etc.), it may or may not touch the yellow surface immediately.
True.

Result: *Wrong* ✖

Object Contact Detection



Prompt: *"These are 4 images taken sequentially from a video. Does the red object touch the yellow surface at any point in the video? Explain your thinking and end with True or False only."*

GT Label: *False*

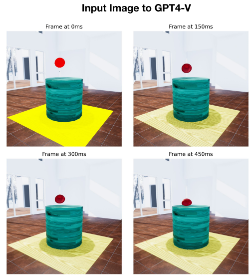
GPT-4V Response:
Based on the images provided:
In the "Frame at 0ms," the red object is above the blue container and does not appear to be touching the yellow surface.
In the "Frame at 150ms," the red object's position hasn't changed much and is still not touching the yellow surface.
In the "Frame at 300ms," the red object is much closer to the blue container's top but still does not seem to be touching the yellow surface.
In the "Frame at 450ms," the red object appears to be right at the edge of the blue container's top, and it doesn't seem to have touched the yellow surface.
From these observations, there's no indication that the red object ever touches the yellow surface in these frames.
False.

Result: *Correct* ✓

(a) Querying method: RGB frames only

Input Image to GPT4-V

Object Contact Prediction



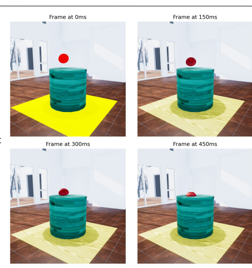
Prompt: *"These are 4 images taken sequentially from a video. If the video were to continue, would the red object touch the yellow surface, as colored in the first frame? Explain your thinking and end with True or False only."*

GT Label: *False*

GPT-4V Response:
The images show a red object above a cylinder with a yellow surface around it. The sequence of images is taken at 0ms, 150ms, 300ms, and 450ms intervals. From the sequence, we can deduce the following:
The red object appears to be falling, as it is positioned lower with each consecutive frame.
Given that the object is already quite close to the cylinder at the 450ms frame and continues to move downwards, we can infer that if this movement continues in the same manner, the object will likely touch the yellow surface.
There's no evidence in the provided frames that the object is slowing down or that there are any forces (like an updraft) that would cause it to rise or hover.
Based on this analysis and assuming that no other forces come into play, the red object will likely continue its downward trajectory and touch the yellow surface.
True.

Result: *Wrong* ✖

Object Contact Detection



Prompt: *"These are 4 images taken sequentially from a video. Does the red object touch the yellow surface at any point in the video, as colored in the first frame? Explain your thinking and end with True or False only."*

GT Label: *False*

GPT-4V Response:
The sequence of images shows a red object descending towards a blue cylinder on a yellow square, captured at 0ms, 150ms, 300ms, and 450ms. There's no visual evidence in these frames of the red object touching the yellow surface; it remains above the cylinder at all times. Without further frames showing the continuation of the motion, we cannot observe the object contacting the yellow surface within the time frame provided.

Thus, based on the available images:

False

Result: *Correct* ✓

(b) Querying method: RGB frames with objects of interest highlighted

Fig. 15: More GPT-4V results. 7

References

1. Babaeizadeh, M., Saffar, M.T., Nair, S., Levine, S., Finn, C., Erhan, D.: Fitvid: Overfitting in pixel-level video prediction. arXiv preprint arXiv:2106.13195 (2021) 21, 22
2. Bao, H., Dong, L., Piao, S., Wei, F.: Beit: Bert pre-training of image transformers. arXiv preprint arXiv:2106.08254 (2021) 1
3. Bardes, A., Garrido, Q., Ponce, J., Rabbat, M., LeCun, Y., Assran, M., Ballas, N.: Revisiting feature prediction for learning visual representations from video. arXiv preprint (2024) 8
4. Bear, D.M., Feigelis, K., Chen, H., Lee, W., Venkatesh, R., Kotar, K., Durango, A., Yamins, D.L.: Unifying (machine) vision via counterfactual world modeling. arXiv preprint arXiv:2306.01828 (2023) 3
5. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: Proceedings of the International Conference on Computer Vision (ICCV) (2021) 21, 22
6. Chen, M., Radford, A., Child, R., Wu, J., Jun, H., Luan, D., Sutskever, I.: Generative pretraining from pixels. In: International conference on machine learning. pp. 1691–1703. PMLR (2020) 3
7. Ebert, F., Yang, Y., Schmeckpeper, K., Bucher, B., Georgakis, G., Daniilidis, K., Finn, C., Levine, S.: Bridge data: Boosting generalization of robotic skills with cross-domain datasets. arXiv preprint arXiv:2109.13396 (2021) 3
8. elenium Contributors: Selenium: Browser automation framework, `https://www.selenium.dev/` 8
9. Gan, C., Schwartz, J., Alter, S., Mrowca, D., Schrimpf, M., Traer, J., De Freitas, J., Kubilius, J., Bhandwaldar, A., Haber, N., et al.: Threedworld: A platform for interactive multi-modal physical simulation. arXiv preprint arXiv:2007.04954 (2020) 6
10. Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., He, K.: Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677 (2017) 3
11. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 16000–16009 (2022) 1, 2, 21, 22
12. Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., et al.: The kinetics human action video dataset. arXiv preprint arXiv:1705.06950 (2017) 2
13. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016) 3
14. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017) 3
15. Mehl, L., Schmalfuss, J., Jahedi, A., Nalivayko, Y., Bruhn, A.: Spring: A high-resolution high-detail dataset and benchmark for scene flow, optical flow and stereo. In: Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2023) 4
16. Nair, S., Rajeswaran, A., Kumar, V., Finn, C., Gupta, A.: R3m: A universal visual representation for robot manipulation. arXiv preprint arXiv:2203.12601 (2022) 21, 22
17. OpenAI: Gpt-4 for vision (chatgpt with image input) (2023), `https://openai.com/`, accessed: October 27, 2023 21, 22

18. Oquab, M., Darcet, T., Moutakanni, T., Vo, H.V., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Howes, R., Huang, P.Y., Xu, H., Sharma, V., Li, S.W., Galuba, W., Rabbat, M., Assran, M., Ballas, N., Synnaeve, G., Misra, I., Jegou, H., Mairal, J., Labatut, P., Joulin, A., Bojanowski, P.: Dinov2: Learning robust visual features without supervision (2023) 21, 22
19. Perazzi, F., Pont-Tuset, J., McWilliams, B., Van Gool, L., Gross, M., Sorkine-Hornung, A.: A benchmark dataset and evaluation methodology for video object segmentation. In: Computer Vision and Pattern Recognition (2016) 3
20. Pont-Tuset, J., Perazzi, F., Caelles, S., Arbelaez, P., Sorkine-Hornung, A., Gool, L.V.: The 2017 davis challenge on video object segmentation. arXiv preprint arXiv:1704.00675 (2017) 4
21. Riochet, R., Castro, M.Y., Bernard, M., Lerer, A., Fergus, R., Izard, V., Dupoux, E.: Intphys 2019: A benchmark for visual intuitive physics understanding. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(9), 5016–5025 (2021) 8
22. Teed, Z., Deng, J.: Raft: Recurrent all-pairs field transforms for optical flow. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16. pp. 402–419. Springer (2020) 4
23. Tong, Z., Song, Y., Wang, J., Wang, L.: Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. Advances in neural information processing systems **35**, 10078–10093 (2022) 21, 22
24. Voleti, V., Jolicoeur-Martineau, A., Pal, C.: Mcvd-masked conditional video diffusion for prediction, generation, and interpolation. Advances in Neural Information Processing Systems **35**, 23371–23385 (2022) 21, 22
25. Wang, L., Huang, B., Zhao, Z., Tong, Z., He, Y., Wang, Y., Wang, Y., Qiao, Y.: Videomae v2: Scaling video masked autoencoders with dual masking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14549–14560 (2023) 21, 22
26. Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Van Gool, L.: Temporal segment networks for action recognition in videos. IEEE transactions on pattern analysis and machine intelligence **41**(11), 2740–2755 (2018) 3
27. Wang, X., Yu, Z., De Mello, S., Kautz, J., Anandkumar, A., Shen, C., Alvarez, J.M.: Freesolo: Learning to segment objects without annotations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14176–14186 (2022) 5
28. Wang, X., Girdhar, R., Yu, S.X., Misra, I.: Cut and learn for unsupervised object detection and instance segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3124–3134 (2023) 5
29. Yan, W., Hafner, D., James, S., Abbeel, P.: Temporally consistent transformers for video generation (2023) 21, 22