MolmoMotion: Forecasting Point Trajectories in 3D with Language Instruction

Jianing Zhang[1,2*†] **Chenhao Zheng**[1,2*†] **Yajun Yang**[2†] **Max Argus**[1] **Rustin Soraki**[1,2]
Winson Han[1] **Chun-Liang Li**[2] **Zhongzheng Ren**[1,2,3†] **Jieyu Zhang**[1,2†] **Ranjay Krishna**[1,2†]

[1]Allen Institute for AI [2]University of Washington [3]UNC-Chapel Hill

[*]Equal contribution [†]Core contributors

Code, models, and data: https://molmomotion.github.io

Abstract

Motion forecasting is central to visual intelligence: agents must anticipate how objects will move in order to plan actions, reason about physical interactions, and synthesize realistic futures. We argue that 3D points in world coordinates provide a general representation that is class-agnostic, view-stable, compact, and directly useful for downstream tasks. We formalize the task of *goal-conditioned 3D point motion forecasting*: given a short visual history, a set of 3D query points on an object of interest, and a language description of the intended goal, the model predicts the future 3D trajectory of each point. We introduce a full stack to study this task at scale: (1) **MolmoMotion-1M** is a large corpus of action-described, object-grounded 3D point trajectory dataset annotated from 1.16M unconstrained videos; (2) **PointMotionBench** is a human-verified benchmark spanning 111 object categories and 61 motion types; and (3) **MolmoMotion** is a general motion forecasting model that supports both autoregressive coordinate prediction and flow-matching-based trajectory generation. MolmoMotion is able to accurately predicts diverse motion patterns with different language instructions, and significantly outperforms all existing motion prediction baselines on PointMotionBench. Finally, we show that the learned 3D motion prior transfers well to downstream applications: it improves training efficiency and generalization for robot manipulation, and its predicted trajectories provide effective motion guidance for generative models to synthesize videos with more realistic object motion.

1 Introduction

Psychologists like James J. Gibson have long argued that motion is core to perception, hypothesizing that motion informs an observer how they and other objects move through space, explains object occlusion and permanence, and identifies affordances [1]. In the 70s, Ullman formalized motion perception as a computational problem [2], with Lucas and Kanade providing an algorithm for estimating motion to enable tracking as optical flow [3]. Although such methods have improved [4–6], they remain primarily focused on estimating motion that has *already* occurred. Many real-world applications instead require *forecasting* how motion will unfold. In robotics, an agent must anticipate how its actions will move objects through the scene [7, 8]. In video generation, realistic synthesis requires precise forecasting of the future motion of objects [9, 10].

Building a general motion forecasting model imposes several requirements on the representation used for motion. First, the representation should be *class-agnostic*: it should not depend on templates tailored to humans, hands, rigid objects, or any other fixed categories. Second, it should be *view-stable*: the same underlying motion should be represented consistently across different cameras, from static surveillance footage to egocentric robot videos and moving outdoor platforms. Third, it

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

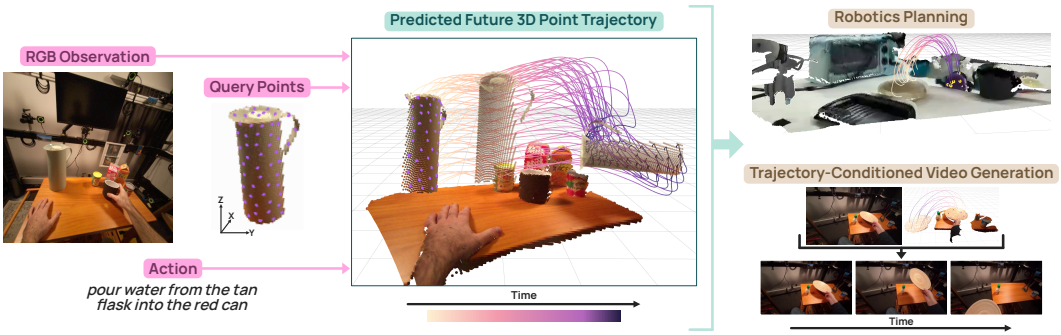


Figure 1: **Overview.** We introduce the task of goal-conditioned 3D point motion prediction. Given initial 3D query points on an object, history RGB observations, and a language description of the future action, our model predicts the future 3D positions of all queried points in a metric world coordinate frame. We show that pretraining this motion prediction task produces a transferable motion representation for downstream applications, including robotics planning and video generation.

should expose *physical structure* in a form that downstream systems can use directly. Existing approaches of motion prediction only partially satisfy these requirements. While *pixels* provide a rich rendering of possible futures [11, 9, 10], they are expensive to generate and often difficult to utilize directly in downstream applications. Many applications instead require explicit geometric and physical quantities, such as object pose [12–14] or particle-level dynamics [15, 16]. *Parametric 3D models* (for humans [17], hands [18], or rigid objects [19]) are useful for such applications but remain restricted to specific categories and embodiments. *2D point trajectories* are more category-agnostic, but image-plane coordinates entangle object motion with camera ego-motion and viewpoint change, making them difficult to disentangle from videos and transfer across domains [8, 20].

We argue that *object-attached 3D points in world coordinates* provide a suitable representation for general motion forecasting. A sparse set of surface points moving through 3D space can describe the motion of rigid, articulated, or deformable objects without assuming a category-specific template. Because 3D points share a world frame, the same physical motion can remain stable across cameras, viewpoints, and capture settings. Additionally, many task-relevant quantities can be expressed as the motion of 3D points, from the pose change of a robot gripper [14] to particle trajectories in physical simulation [15, 16]. Finally, by focusing only on object-attached points, the prediction target remains compact while still capturing the motion relevant for physical interaction.

We formalize this idea as the task of *goal-conditioned 3D point motion prediction* (Fig. 1). Given a short history of visual observations, a set of initial 3D query points on an object of interest, and a language description of the intended goal, the model predicts the future 3D trajectory of each query point over time. The language instruction disambiguates among plausible futures and reduces the search space of possible future states.

We present the full stack needed to study this task at scale: **a scalable data curation pipeline**, **a novel prediction model**, and **a human-verified benchmark**. The first challenge is 3D motion supervision data. Existing video datasets with 3D capture are small and domain-limited [21–23], while internet videos provide scale and diversity but lack 3D annotations. We therefore develop an automatic annotation pipeline for robustly extracting object-grounded 3D point trajectories from unconstrained videos. Applying this pipeline to roughly 1.16M video clips yields MolmoMotion-1M, the largest corpus of action-described, object-grounded 3D point trajectories. We additionally introduce PointMotionBench, a benchmark for 3D motion forecasting spanning 111 object categories and 61 motion types. The benchmark uses ground-truth 3D capture where available and human-verified 3D tracks otherwise, providing a reliable testbed for evaluating motion prediction.

We introduce MolmoMotion, a general motion prediction model pretrained on the MolmoMotion-1M dataset. We train two complementary classes of motion prediction models. The first predicts future trajectories autoregressively as coordinate sequences; the second predicts future motion with flow matching as a continuous trajectory distribution. Experiments show MolmoMotion accurately forecast diverse motion types and generalize across a wide range of scenes and instructions, and significantly outperforms all existing motion prediction methods in PointMotionBench.

Finally, we validate 3D point forecasting as a useful task: MolmoMotion transfers effectively to downstream applications. Because object trajectories in 3D world space are largely embodiment-agnostic [8, 24], the motion prior learned from internet-scale human video can transfer naturally to robotics planning. We show that this prior improves sample efficiency, closed-loop success, and generalization to unseen objects and scenes in the MolmoSpaces pick-and-place task [25], and adapts efficiently to real-world robot videos from DROID [26]. MolmoMotion can also guide video generation. When its predicted trajectories are used to condition a lightweight 3D-track-conditioned image-to-video model [27], the generated videos exhibit more realistic motion than those produced directly by much larger image-to-video models [11], while also improving multiple quantitative video-quality metrics.

2 Related work

Motion prediction models. Motion prediction models differ mainly in the representation they forecast. Pixel-space methods cast prediction as conditional video generation, from latent-action world models [28, 29] to recent large-scale video generators [10, 9, 11]; these models are general but spend substantial capacity rendering appearance, lighting, and camera motion. Alongside multimodal representation learning [30, 31], latent prediction methods forecast learned feature states [32, 33], avoiding pixels but producing encoder-tied representations that are difficult to use as physical quantities. Recent point-trajectory forecasting methods predict category-agnostic motion directly [8, 20], but operate in 2D image space, where object motion is entangled with camera motion and viewpoint change. 3D forecasting methods predict physically grounded states such as human motion [34], rigid-object pose [35, 19], or scene-level 3D motion [36], but are often tied to specific object categories or domains. We predict object-attached 3D point trajectories in world coordinates, yielding a category-agnostic and physically grounded representation.

3D point tracking. Our data annotation pipeline builds on recent advances in 3D point tracking from monocular video. 2D point trackers estimate temporally persistent pixel locations for queried points [5, 37], with recent models improving long-range tracking [38], occlusion reasoning [6], and dense correspondence over entire videos [39]. Recent monocular reconstruction methods lift image observations into 3D by estimating camera motion and per-frame depth, enabling pixel tracks to be lifted to 3D [40, 41]. There are also direct 3D point trackers that track points in 3D space [42–44]. Together, these methods have made it increasingly feasible to recover 3D motion from ordinary RGB videos, though the focus is on tracking observed frames rather than forecasting future motion.

Human-to-robot transfer. Prior work transfers non-robot video to robot control through different abstractions. Some methods re-target human hands, endpoints, or skill trajectories as planning scaffolds [45, 14, 46, 12, 47, 48]. Others train generalist vision-language-action policies on cross-embodiment robot data [49–53], sometimes with human video, but still predict embodiment-specific actions. Closest to ours are methods that learn motion priors or controls from unlabeled video through latent actions or inverse dynamics [54–58]. In contrast, we use 3D world-frame object point trajectories as the pretraining target itself, yielding a motion prior that is both embodiment-agnostic and directly usable for downstream robot learning.

Trajectory representations and recent world-action models. TrajViT [59] groups observed video content into panoptic sub-object trajectory tokens, while TrajTok [60] learns trajectory tokenization with a differentiable segmenter and encoder for video understanding. These methods use trajectories to organize visual representations; MolmoMotion instead predicts future object-attached 3D point trajectories in a shared world frame. 3DPWM [61] completes partial point clouds before learning action-conditioned dynamics, emphasizing completed scene geometry for long-horizon planning. PointZero [62] learns transferable 3D dynamics by completing point tracks from an RGB-D observation and sparse partial trajectories, with synthetic pretraining followed by adaptation to dynamics prediction and imitation learning. EgoWAM [63] studies human–robot co-training with alternative world-prediction targets, finding that DINO features and 3D motion flow transfer more effectively than pixels. ModAR [64] sequentially predicts future point tracks, visual features, and depth before predicting robot actions. These approaches complement our focus on learning a category-agnostic 3D motion prior from large-scale video.

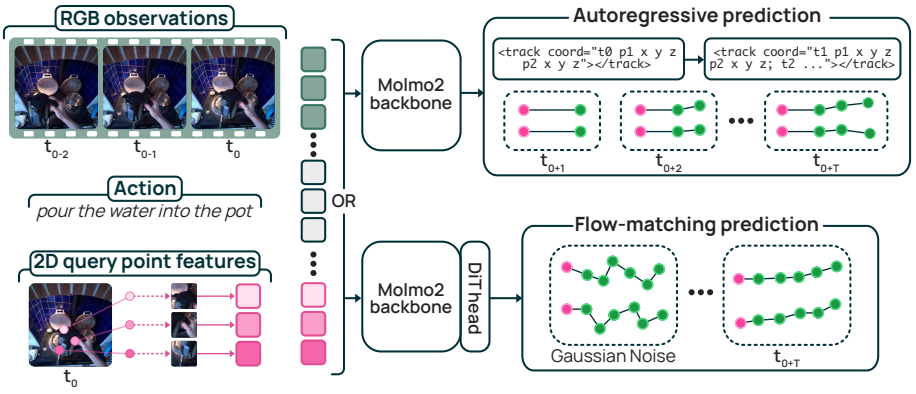


Figure 2: **MolmoMotion architecture.** The shared input to Molmo2 [65] backbone consists of image tokens of RGB observations, text tokens of action description, and 2D query point feature tokens sampled from Molmo2 vision encoder. The autoregressive variant encodes the initial 3D query coordinates and decodes future trajectories as quantized coordinate text, while the flow-matching variant represents them directly in continuous 3D coordinate space.

3 MolmoMotion

We formulate general motion prediction as the task of predicting future 3D trajectories of points attached to an object of interest, conditioned on an action description. In this section, we formulate the problem (Sec. 3.1) and introduce our model architecture (Sec. 3.2).

3.1 Problem formulation

We define the model input as follows. At a reference time t_0, the model is given N user-specified 2D query points on an object of interest in the image, $\{\mathbf{q}_{t_0}^n \in \mathbb{R}^2\}_{n=1}^N$. It is also given the corresponding initial 3D positions $\{\mathbf{p}_{t_0}^n \in \mathbb{R}^3\}_{n=1}^N$, expressed in the camera coordinate frame at t_0. In practice, these 3D positions can be obtained by lifting the 2D query points using estimated or measured depth together with known camera intrinsics. The model also receives a short history of RGB observations, $I_{t_s:t_0} = \{I_{t_s}, \ldots, I_{t_0}\}$, and a language description a of the intended action. The goal is to predict the future 3D positions of all query points, over a horizon of T steps: $\{\{\hat{\mathbf{p}}_t^n \in \mathbb{R}^3\}_{t=t_0+1}^{t_0+T}\}_{n=1}^N$. All future coordinates are expressed in a world coordinate frame anchored at the camera at time t_0. This choice makes the prediction independent of future camera motion.

3.2 MolmoMotion architecture

We implement two classes of trajectory predictors: one with an autoregressive objective and the other with a flow-matching objective. They share the same input encoding of the RGB observation, action description, and 2D query-point visual features. They differ in how the initial 3D query coordinates are encoded and how the future 3D trajectories are decoded (Fig. 2). We study both objectives because they offer complementary modeling biases: autoregressive decoding conditions each prediction on the previously generated trajectory, which encourages smooth temporal evolution, while flow-matching models a distribution over future trajectories and is better suited for capturing motion uncertainty [66].

Input encoding. Predicting object motion from a language instruction first requires grounding the relevant object in the image and understanding the instruction. We therefore use Molmo2 [65] as the vision-language backbone for input processing, leveraging its strong object-grounding capability. Given the RGB observation history $I_{t_s:t_0}$, the Molmo2 vision encoder produces image tokens $\mathcal{T}_{\text{img}}$. The action description a is tokenized into language tokens $\mathcal{T}_{\text{text}}$. To condition on the query points, we additionally insert one visual point token for each 2D query coordinate. Let F_{t_0} be the anchor-frame feature map produced by the vision encoder. For each query point $\mathbf{q}_{t_0}^n$, we bilinearly sample the anchor-frame feature map F_{t_0} at its 2D location to obtain a point feature $\mathbf{e}_{\text{pt}}^n$. These features form the point-token sequence $\mathcal{T}_{\text{pt}} = \{\mathbf{e}_{\text{pt}}^1, \ldots, \mathbf{e}_{\text{pt}}^N\}$. We concatenate the image, text, and point tokens as $\mathcal{C} = [\mathcal{T}_{\text{img}}, \mathcal{T}_{\text{text}}, \mathcal{T}_{\text{pt}}]$ and process them with the Molmo2 language model component.

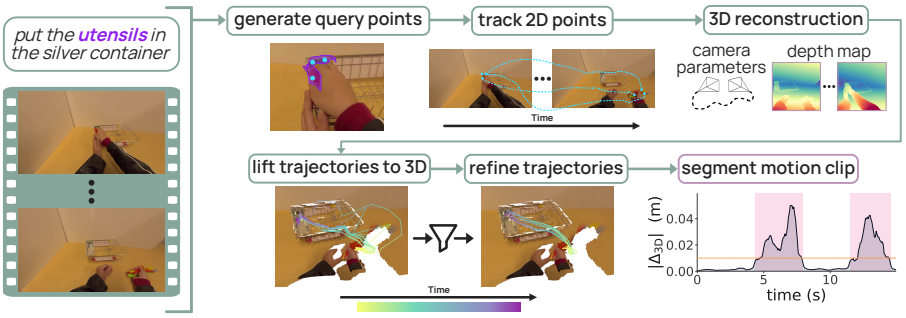


Figure 3: **Overview of data annotation pipeline.** Given a video of an action event and its description, we first ground the moving object and sample query points on it. We then track dense 2D points on the object, lift these tracks into a shared metric 3D frame, and use object-level spatial and temporal consistency priors to filter unreliable trajectories. Finally, we clip the video around intervals where the grounded object undergoes meaningful motion.

For both prediction variants, we represent 3D coordinates relative to the first query point at t_0. Let $\mathbf{p}_{\text{anc}} = \mathbf{p}^1_{t_0}$ denote this anchor point. For each point n and time t, our models represent 3d point coordinates as $\boldsymbol{\delta}^n_t = \mathbf{p}^n_t - \mathbf{p}_{\text{anc}}$. All 3D point coordinates are in metric scale, in units of *meter*.

Autoregressive objective. The autoregressive variant follows Molmo2's coordinate representation, encoding both the initial 3D query coordinates and the future trajectory as structured text. Each anchor-relative coordinate is discretized into *millimeter* bins ($\bar{\boldsymbol{\delta}}^n_t = \text{round}(1000\,\boldsymbol{\delta}^n_t)$) and serialized as timestamped point-coordinate tuples. Fig. 2 shows an example of the coordinate encoding format. The input prompt contains the visual-language conditioning $\mathcal{C}$ together with the serialized initial query coordinates $\{\bar{\boldsymbol{\delta}}^n_{t_0}\}_{n=1}^N$. The output is the serialized future trajectory $y_{1:L}$, generated in temporal order. An example of text pieces encoded and decoded can be found at Fig. 2. We train the autoregressive model with the standard next-token objective. At inference time, the model decodes the trajectory string autoregressively and the coordinates are parsed back into $\hat{\mathbf{P}}_{t_0+1:t_0+T}$. Since coordinates are emitted in temporal order, each future timestamp is conditioned on all earlier generated coordinates, giving the model a direct mechanism for modeling temporal dependence and smooth rollouts.

Flow-matching objective. The flow-matching variant predicts future trajectories in continuous 3D coordinate space. Following a MolmoBot-style design [67], we use a DiT decoder [68] conditioned on Molmo2 features from all layers, with lightweight MLPs for coordinate encoding and decoding. We concatenate the clean initial 3D query coordinates with a noised version of the future coordinates and project them into point tokens. We use RoPE along both the point and time axes to distinguish point identity and timestamp, analogous to the Multimodal RoPE used in Qwen2.5-VL [69, 70].

The decoder is trained with the standard flow-matching objective [71]. Specifically, we sample Gaussian noise with the same shape as the future trajectory, linearly interpolate between the noise and the ground-truth trajectory, and train the decoder to predict the corresponding velocity field from the noised trajectory to the clean trajectory. At inference time, the model starts from Gaussian noise and integrates the learned velocity field with 10 Euler steps to obtain the predicted future trajectory.

4 MolmoMotion-1M and PointMotionBench

Existing 3D point-track datasets are small, domain-limited, and often lack object grounding or language annotations. We therefore build an automatic annotation pipeline that extracts object-grounded 3D point trajectories from unconstrained videos. Applying it to 1.16M public videos yields **MolmoMotion-1M**, the largest action-described, object-grounded 3D point trajectory dataset. We also introduce **PointMotionBench**, a held-out benchmark with verified 3D point trajectories for object-centric 3D motion forecasting.

4.1 MolmoMotion-1M Annotation

We start with public video datasets that provide action descriptions or task captions [72–75]. Our goal is to localize the described object, select query points, and track them in 3D world coordinates.

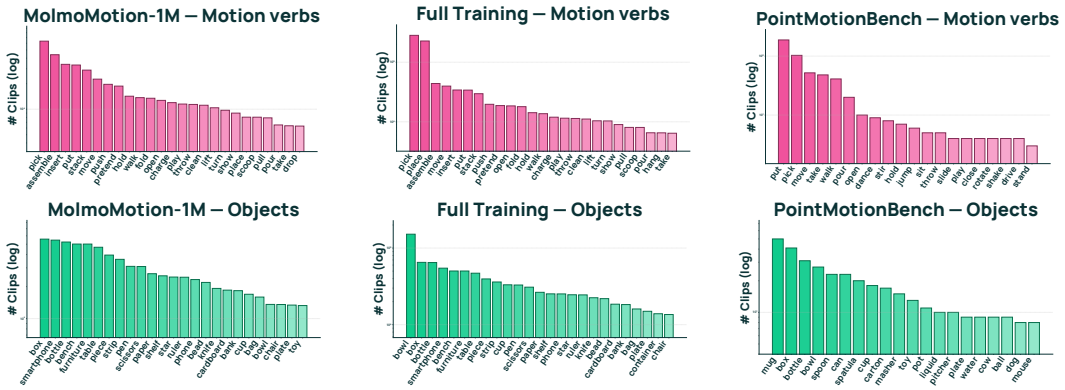


Figure 4: **Data diversity.** Most frequent motion verbs (top) and manipulated objects (bottom) in the pretraining corpus (MolmoMotion-1M, left), the full training corpus including the MolmoSpaces and DROID downstream data (middle), and PointMotionBench (right). Generic object tokens (*object*, *hand*, *person*) are excluded; bars are log-scaled.

The pipeline consists of five stages, shown in Fig. 3: semantic object grounding, temporal point correspondence, metric 3D lifting, trajectory-level filtering, and video-level clipping.

Semantic object grounding. Given an action description a, we use an LLM [76] to extract the manipulated or moving entity as a short object phrase. Instead of prompting SAM3 [77] to directly produce an object mask from this phrase, we first use MolmoPoint [78] to localize the entity as a 2D point in the reference frame t_0. We find MolmoPoint more reliable for vague phrases like "an object on top of the table," because the prompt can specify the object by its action, e.g., the object being moved, rather than by appearance alone. We then use the point prompt with SAM3 to obtain the object mask M_{t_0}, and sample N query points $\{\mathbf{q}_{t_0}^n\}_{n=1}^N$ inside M_{t_0} using K-means cluster centers.

2D point tracking and metric 3D lifting. We next establish point correspondence across frames by tracking the query points through the video. We run AllTracker [39] to obtain temporally persistent 2D tracks $\{\mathbf{q}_t^n\}_{t=1}^L$ and visibility masks $\{m_t^n\}_{t=1}^L$. To lift these tracks into 3D, we run ViPE [40] on the monocular video to estimate per-frame metric depth and camera geometry. We empirically find that this paradigm produces more accurate 3D trajectories than current end-to-end 3D point trackers such as SpatialTrackerV2 [42]. ViPE also provides metric scale output, allowing the resulting trajectories to be expressed in physical 3D units rather than a arbitrary relative scale. Using the estimated depth, intrinsics, and camera poses, we back-project each visible 2D track point into a world frame anchored at the first-frame camera, producing metric 3D tracks $\{\tilde{\mathbf{p}}_t^n \in \mathbb{R}^3\}_{t=1}^L$.

Trajectory-level filtering and smoothing. We empirically find that some lifted trajectories can be corrupted by 2D tracking drift, depth noise, and camera estimation errors. We therefore both filter outlier tracks and smooth remaining tracks using the object-level prior that sampled points should move coherently as parts of the same physical entity. We remove tracks with consistently low trust using a MAD-based outlier criterion [79]. For retained tracks, we follow the smoothing algorithm in Stereo4D [75] to smooth their depth value along each camera ray. More details can be found in Appendix B.4. This removes high-frequency depth jitter in annotated 3D tracks.

Video-level clipping. Event-level video clips often contain long static intervals before or after the described motion, which provide little supervision for learning dynamics. We therefore re-clip videos around intervals where the grounded object actually moves. Given the filtered 3D trajectories, we compute a per-frame object motion score s_t as the median 3D displacement of valid object points: $s_t = \text{median}_n \left\| \mathbf{p}_t^n - \mathbf{p}_{t-1}^n \right\|_2$. We threshold s_t to obtain contiguous segments with non-trivial object motion. This step also automatically removes static videos.

Pretraining corpus statistics. Our videos span human manipulation, hand-object interaction, and in-the-wild scenarios. The largest portion comes from human-object manipulation datasets: EgoDex [72], HD-EPIC [73], and Xperience-10M [74] together contribute the bulk of egocentric and third-person manipulation clips; we additionally include YT-VIS [80] and Stereo4D [75] clips, which broaden coverage to outdoor scenes and deformable objects like animals.

After filtering and segmentation, the pipeline produces approximately 1M clips with motion (per-corpus counts in Tab. 3). The corpus spans 736 unique action verbs and 5,692 unique manipulated

Paradigm	Model	Inputs		HOT3D [21]			WorldTrack [43]			DAVIS [81]		
		Frames	Text	ADE↓	FDE↓	PWT↑	ADE↓	FDE↓	PWT↑	ADE↓	FDE↓	PWT↑
Non-parametric	Static	1	✗	0.180	0.316	0.293	0.167	0.317	0.390	2.281	4.360	0.085
	Extrapolate	3	✗	0.159	0.309	0.351	0.184	0.432	0.436	2.683	5.741	0.104
Pixel-space	Wan2.2-5B [11]	1	✓	0.200	0.308	0.253	0.852	1.046	0.090	3.074	5.192	0.051
	Cosmos Predict [9]	5	✓	0.225	0.294	0.199	0.831	0.988	0.072	4.191	6.368	0.033
3D model	ObjectForesight [19]	3	✗	0.129	0.192	0.353	–	–	–	–	–	–
	EgoScaler [35]	1	✓	0.170	**0.179**	0.218	–	–	–	–	–	–
	Robot4DGen [82]	3	✗	0.212	0.271	0.112	0.548	0.704	0.121	2.120	3.382	0.081
2D track	Track2Act [8]	1	✗	0.294	0.413	0.202	1.230	1.567	0.053	4.853	8.110	0.018
3D track (Ours)	**MolmoMotion-FM**	1	✓	0.183	0.311	0.286	0.165	0.305	0.401	1.380	2.205	0.165
	MolmoMotion-FM	3	✓	0.135	0.255	0.382	0.158	0.295	0.438	1.480	2.520	0.130
	MolmoMotion-AR	1	✓	0.157	0.290	0.303	0.148	0.269	0.424	**1.146**	**1.843**	**0.199**
	MolmoMotion-AR	3	✓	**0.109**	0.217	**0.444**	**0.143**	**0.261**	**0.445**	1.227	2.108	0.153

Table 1: **3D point trajectory prediction on PointMotionBench.** We report 3D ADE, FDE, and average PWT in meters. Input columns indicate the number of past observations consumed and whether the model accepts a natural-language action description. *MolmoMotion-AR* denotes our autoregressive variant, while *MolmoMotion-FM* denotes the flow-matching variant.

objects. Clips are short by construction: median clip length is 0.8–1.1 s on the manipulation corpora and 1.7 s on Stereo4D, where third-person walking subjects yield longer motion windows. Median per-clip 3D displacement ranges from 7–9 cm on the manipulation corpora to 51 cm on Stereo4D, reflecting the difference between tabletop manipulation and walking subjects. After filtering, each clip retains a median of 88 query points (range 60–100). Fig. 4 shows the per-cohort distributions of motion verbs and manipulated objects across the pretraining corpus (MolmoMotion-1M), and the full training corpus that additionally includes the MolmoSpaces and DROID data used for downstream robot finetuning.

4.2 PointMotionBench

We also introduce **PointMotionBench**, a held-out benchmark for evaluating object-centric 3D motion forecasting. Unlike the pretraining corpus, which starts from videos with action descriptions, PointMotionBench repurposes datasets with ground-truth 3D capture to ensure annotation accuracy. For HOT3D [21] and WorldTrack [43], we extract 3D point trajectories directly from the provided 3D object mesh or points, ground the foreground points to objects, and annotate action descriptions, with all annotations verified by humans (details in Appendix C). Since HOT3D and WorldTrack mainly cover indoor manipulation and egocentric hand-object interaction, we further include DAVIS [81] to cover outdoor dynamic scenes; for this split, we run our annotation pipeline and manually verify the correctness of each resulting trajectory. In total, the benchmark contains **742** clips spanning **111** object categories and **61** action/motion types. Fig. 4 (right column) shows the per-cohort distributions of motion verbs and manipulated objects across PointMotionBench.

5 Experiments

This section evaluates MolmoMotion on 3D point trajectory prediction (Sec. 5.1) and on two downstream transfer tasks: robotic planning (Sec. 5.2) and trajectory-guided video generation (Sec. 5.3). Model ablations are presented in Appendix E.

5.1 3D point motion forecasting

We first evaluate MolmoMotion and existing motion prediction methods on the 3D point trajectory prediction task in PointMotionBench.

MolmoMotion implementation. MolmoMotion uses the pretrained 4B Molmo2 [65] as its VLM backbone. Training proceeds in two stages. In the first stage, we randomly sample a start timestamp t_0 from each video clip and sample $N = 8$ query points from one object. The model is supervised to predict $T = 8$ future timestamps at 15 fps, giving 64 future 3D point targets per example. We



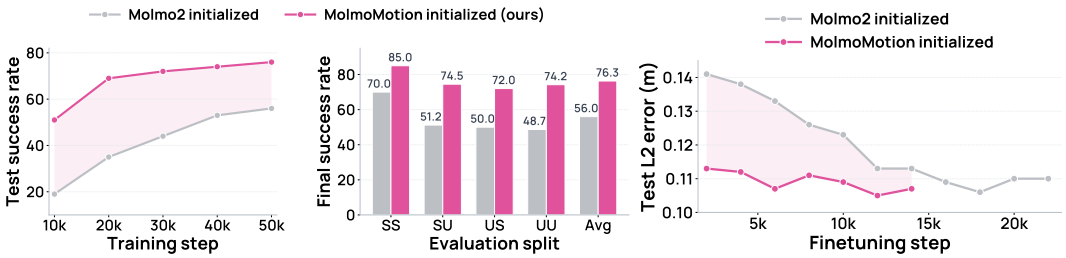
Figure 5: **Qualitative MolmoMotion prediction.** MolmoMotion predicts accurate motion trajectories on diverse motion patterns with different action instructions.

train with history length $H = 3$ in the first stage, providing a short visual history before t_0. This stage runs for 40K steps. In the second stage, we continue training for 10K steps while increasing the prediction horizon to $T = 32$. In this stage we train two varients of models, with $H = 3$ and $H = 1$ respectively, to support both short visual history and single frame history settings.

Baselines. Baselines fall into four families. **Non-parametric baselines** include *Static*, which keeps each point fixed at its initial 3D position, and *Extrapolate*, which estimates a constant velocity from the history frames and linearly extrapolates into the future. **Pixel-space methods** first generate future RGB video and then recover query-point trajectories using the 3D tracking pipeline from Sec. 4.1; we compare against Wan2.2-I2V-5B [11] and Cosmos-Predict-2.5 [9]. **Parametric 3D prediction methods** predict an intermediate 3D representation; we include methods whose output can be converted to 3D point motion. ObjectForesight [19] and EgoScaler [35] forecast future 6-DoF object poses under a rigid-object assumption, and we use their predicted pose sequence to transform all query points. Robot4DGen predicts future scene flow [82], from which we extract query point trajectories. **2D point-track methods** predict future image-plane trajectories; we evaluate Track2Act by lifting its predicted 2D tracks to 3D using ground-truth depth [8]. Note that PointWorld [36] and MotionForcast [20] are also closely related works, but their models have not yet been released.

Evaluation setting and metrics. We evaluate all methods in PointMotionBench. Each method predicts future motion for up to 2 seconds at 15 fps; if the ground truth clip has a shorter valid future horizon, we evaluate only on the available frames. Since baselines vary in the number of input frames H and whether they accept text condition, we follow each baseline's native setting where applicable. Following prior work [20], for baselines with stochastic outputs we generate 5 samples per clip and report the best (best-of-5). We follow the point-track forecasting metrics used in [20], adapted from 2D pixel distance to 3D metric distance. In our setting, one unit corresponds to one meter in 3D world coordinates. ADE measures the mean displacement error across all visible query points and predicted timesteps, while FDE measures the final-timestep displacement error. PWT is the average fraction of predicted points within $\{0.01, 0.02, 0.05, 0.10, 0.20\}$ meters of the ground truth.

Results. Tab. 1 shows the quantitative results. Note that ObjectForesight [19] and EgoScaler [35] are evaluated only on the HOT3D subset because they require object mesh inputs, which are available only for HOT3D. MolmoMotion outperforms prior methods by a large margin in almost all subsets of PointMotionBench, with the autoregressive variant achieving the strongest overall performance. The autoregressive model performs better than the flow-matching model under deterministic trajectory metrics, likely because conditioning on the previously generated coordinate sequence encourages temporally smooth predictions. Using three RGB observation frames generally improves over the single-frame setting, as additional history provides velocity cues for future motion. A notable finding is that simple non-parametric baselines are competitive with, or stronger than, several learned baselines, including pixel-space video prediction methods, suggesting that visually plausible RGB futures do not necessarily recover accurate metric point motion. We also show qualitative examples in Fig. 5, where MolmoMotion accurately predicts 3D motion trajectories across diverse motion patterns and language instructions.



(a) Pick-and-place task on the MolmoSpaces benchmark. (b) Trajectory finetuning on DROID.

Figure 6: **MolmoMotion transfers effectively to robotics planning.** (a) Task success rate on the MolmoSpaces Franka Pick-and-place benchmark over robot-finetuning steps (left) and the final step per split breakdown (right) across SS (seen scene, seen object), SU (seen scene, unseen object), US (unseen scene, seen object), and All. (b) Test L2 error of predicted 3D trajectories on held-out DROID clips. In both settings, initializing from MolmoMotion substantially outperforms the baseline.



Figure 7: **MolmoMotion trajectory prediction on real-world scenarios.** Predicted future 3D object trajectories on a held-out scenario after finetuning MolmoMotion on DROID videos. MolmoMotion can plan accurate object trajectories for various robotic manipulation tasks.

5.2 MolmoMotion transfers effectively to robotics planning

The representation learned by MolmoMotion is well-suited for robotics planning. The intuition is that object motion in 3D is more transferable than embodiment-specific actions. A human hand and a robot gripper execute differently, but successful manipulation often produces similar object trajectories in 3D space.

We use the pick-and-place task as a controlled transfer setting, focusing on the post-grasp stage where the policy must lift, transport and place the object at the correct target. We train two MolmoBot policies [67] with the same flow-matching action head and 20K released episodes, differing only in backbone initialization: Molmo2 pretrained weights vs. MolmoMotion-AR. During evaluation, the original MolmoBot policy performs grasping, after which control is handed to our trained policy. On FrankaPickandPlaceDroidMiniBench in MolmoSpaces [25], we report closed-loop success across seen/unseen scene and object splits. As shown in Fig. 6a, MolmoMotion initialization substantially improves training efficiency and final performance: success reaches 51% at 10K steps vs. 19% for Molmo2, and final average success increases from 56.0% to 76.3%. The smaller drop in unseen-object and unseen-scene splits further suggests MolmoMotion improves downstream generalization.

We further evaluate whether the MolmoMotion can be adapted to real-world robot scenarios. We finetune MolmoMotion on DROID single-camera videos [26] using the same 3D object point trajectory prediction task. Fig. 7 shows qualitative examples of object trajectory planned by MolmoMotion across different scenes, objects, and tasks in DROID held out videos. We also compare against training the same architecture with Molmo2 initialization. As shown in Fig. 6b, MolmoMotion starts with substantially lower trajectory error and reaches the best performance much quicker. This suggests that the motion prior learned by MolmoMotion can be efficiently adapted to real-world robot data. We leave closed-loop real-robot evaluation to future work.

5.3 MolmoMotion provides controllable motion for video generation

A natural question is whether the 3D point trajectories predicted by MolmoMotion can serve as an explicit motion-control signal for downstream video generation. Given an input image and action description, we first obtain query points on the manipulated object and estimate their initial 3D positions using the pipeline in Sec. 4.1. MolmoMotion then predicts their future 3D trajectories, which we use to guide DaS [27], a 3D point-trajectory-guided image-to-video model built on

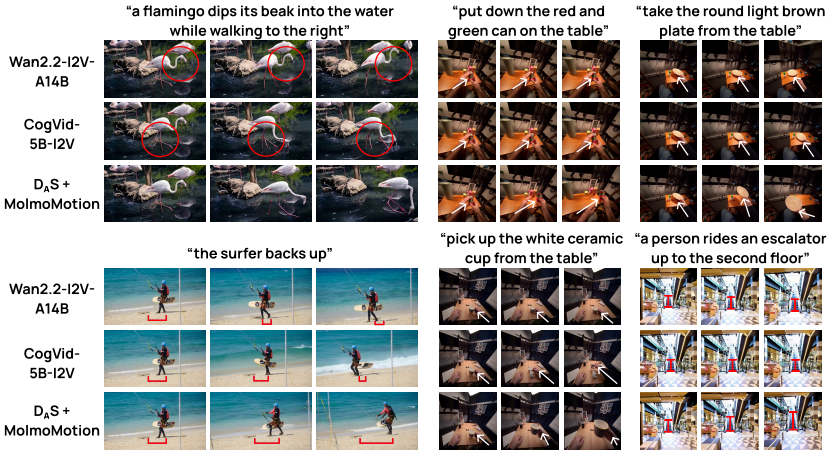


Figure 8: **Qualitative Video Generation Comparison.** MolmoMotion-guided videos exhibit more physically plausible object motion and follow the prompted action more faithfully.

CogVideoX-5B [83]. We compare against caption-conditioned image-to-video generators without explicit motion guidance, including DaS's base model CogVideoX-5B and the larger Wan2.2-I2V-A14B [11].

As shown in Tab. 2, DaS+MolmoMotion improves over CogVideoX-5B on all metrics and outperforms Wan2.2-I2V-A14B on four out of five metrics. Qualitatively (Fig. 8), it produces more physically plausible object motion, especially for fine-grained manipulation. It also follows the prompted action more faithfully. These results suggest that MolmoMotion predicts 3D trajectories that can serve as an effective control interface for downstream video generation.

Method	Tem-Con	Subj-Cons	M-Smooth	Dyn-Deg	Bg-Cons
CogVideoX-5B [83]	0.964	0.939	0.988	0.861	0.941
Wan-14B [11]	0.965	0.940	0.983	**0.908**	0.947
DAS [27] + MolmoMotion	**0.968**	**0.950**	**0.990**	0.876	**0.948**

Table 2: **Video generation quality on PointMotionBench videos.** We report metrics in VBench [84] that evaluates motion: temporal consistency, subject consistency, motion smoothness, dynamic degree, and background consistency.

6 Conclusions and Limitations

We introduce MolmoMotion, a language-guided motion predictor that forecasts future trajectories of object-attached points in a 3D world frame. We built MolmoMotion-1M for pretraining and PointMotionBench for 3D motion evaluation. Experiments show that MolmoMotion significantly outperforms prior motion prediction methods and transfers to robot manipulation and video generation.

Limitations. MolmoMotion requires multiple forward passes to predict dense point tracks on an object, since pretraining uses only 8 query points per example due to Molmo2 context length limit in stage 2 training. This sparse point setting is not sufficient to densely represent object geometry, limiting the model's understanding of fine-grained structure and complex deformable motion. In addition, more downstream evaluations are needed to fully establish the effectiveness of this motion pretraining task, such as closed-loop real-world robot experiments.

Acknowledgments

This work would not be possible without the support of our colleagues at Ai2. We thank David Albright, Stephen Kelman, Kristin Cha, Byron Bischoff, David Everhart, Jon Borchardt, Kyle Wiggers, Will Smith, Peter Clark, Dieter Fox, and Noah Smith for their important work for the MolmoMotion public release. We thank Ropedia for providing access to the Xperience dataset used in this work, and granting permission the release of MolmoMotion under the Apache License 2.0. Chenhao Zheng is partially funded through an Apple grant. We thank Oncel Tuzel, Pavan Kumar, and Rick Chang for the helpful discussion and support on this project.

References

[1] J.J. Gibson. *The Ecological Approach to Visual Perception*. Resources for ecological psychology. Lawrence Erlbaum Associates, 1986. ISBN 9780898599596. URL https://books.google.com/books?id=DrhCCWmJpWUC.

[2] Shimon Ullman. *The Interpretation of Visual Motion*. The MIT Press, 03 1979. ISBN 9780262257121. doi: 10.7551/mitpress/3877.001.0001. URL https://doi.org/10.7551/mitpress/3877.001.0001.

[3] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[4] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020.

[5] Carl Doersch, Ankush Gupta, Larisa Markeeva, Adria Recasens, Lucas Smaira, Yusuf Aytar, Joao Carreira, Andrew Zisserman, and Yi Yang. Tap-vid: A benchmark for tracking any point in a video. *Advances in Neural Information Processing Systems*, 35:13610–13626, 2022.

[6] Nikita Karaev, Iurii Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker3: Simpler and better point tracking by pseudo-labelling real videos. *arXiv preprint arXiv:2410.11831*, 2024.

[7] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2786–2793. IEEE, 2017.

[8] Homanga Bharadhwaj, Roozbeh Mottaghi, Abhinav Gupta, and Shubham Tulsiani. Track2act: Predicting point tracks from internet videos enables generalizable robot manipulation. In *European Conference on Computer Vision (ECCV)*, 2024.

[9] 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.

[10] Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal Behbahani, Stephanie Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. *arXiv preprint arXiv:2402.15391*, 2024.

[11] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[12] Han Zhou, Jinjin Cao, Liyuan Ma, Xueji Fang, and Guo jun Qi. Traj2action: A co-denoising framework for trajectory-guided human-to-robot skill transfer, 2026. URL https://arxiv.org/abs/2510.00491.

[13] Bowen Wen, Jonathan Tremblay, Valts Blukis, Stephen Tyree, Thomas Müller, Alex Evans, Dieter Fox, Jan Kautz, and Stan Birchfield. Bundlesdf: Neural 6-dof tracking and 3d reconstruction of unknown objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 606–617, 2023.

[14] Hanzhi Chen, Boyang Sun, Anran Zhang, Marc Pollefeys, and Stefan Leutenegger. Vidbot: Learning generalizable 3d actions from in-the-wild 2d human videos for zero-shot robotic manipulation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 27661–27672, 2025.

[15] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B. Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. *arXiv preprint arXiv:1810.01566*, 2018.

[16] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pages 8459–8468. PMLR, 2020.

[17] Ling-Hao Chen, Jiawei Zhang, Yewen Li, Yiren Pang, Xiaobo Xia, and Tongliang Liu. Humanmac: Masked motion completion for human motion prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9544–9555, 2023.

[18] Hongzhe Bi, Lingxuan Wu, Tianwei Lin, Hengkai Tan, Zhizhong Su, Hang Su, and Jun Zhu. H-rdt: Human manipulation enhanced bimanual robotic manipulation, 2025. URL https://arxiv.org/abs/2507.23523.

[19] Rustin Soraki, Homanga Bharadhwaj, Ali Farhadi, and Roozbeh Mottaghi. Objectforesight: Predicting future 3d object trajectories from human videos. *arXiv preprint arXiv:2601.05237*, 2026.

[20] Neerja Thakkar, Shiry Ginosar, Jacob Walker, Jitendra Malik, Joao Carreira, and Carl Doersch. Forecasting motion in the wild. *arXiv preprint arXiv:2604.01015*, 2026.

[21] Prithviraj Banerjee, Sindi Shkodrani, Pierre Moulon, Shreyas Hampali, Shangchen Han, Fan Zhang, Linguang Zhang, Jade Fountain, Edward Miller, Selen Basol, Richard Newcombe, Robert Wang, Jakob Julian Engel, and Tomas Hodan. HOT3D: Hand and object tracking in 3D from egocentric multi-view videos. *CVPR*, 2025.

[22] Skanda Koppula, Ignacio Rocco, Yi Yang, Joe Heyward, Joao Carreira, Andrew Zisserman, Gabriel Brostow, and Carl Doersch. Tapvid-3d: A benchmark for tracking any point in 3d. *Advances in Neural Information Processing Systems*, 37:82149–82165, 2024.

[23] Yunze Liu, Yun Liu, Che Jiang, Kangbo Lyu, Weikang Wan, Hao Shen, Boqiang Liang, Zhoujie Fu, He Wang, and Li Yi. Hoi4d: A 4d egocentric dataset for category-level human-object interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21013–21022, June 2022.

[24] Karthik Dharmarajan, Wenlong Huang, Jiajun Wu, Li Fei-Fei, and Ruohan Zhang. Dream2flow: Bridging video generation and open-world manipulation with 3d object flow. *arXiv preprint arXiv:2512.24766*, 2025.

[25] Yejin Kim, Wilbert Pumacay, Omar Rayyan, Max Argus, Winson Han, Eli VanderBilt, Jordi Salvador, Abhay Deshpande, Rose Hendrix, Snehal Jauhri, Shuo Liu, Nur Muhammad Mahi Shafiullah, Maya Guru, Ainaz Eftekhar, Karen Farley, Donovan Clay, Jiafei Duan, Arjun Guru, Piper Wolters, Alvaro Herrasti, Ying-Chun Lee, Georgia Chalvatzaki, Yuchen Cui, Ali Farhadi, Dieter Fox, and Ranjay Krishna. Molmospaces: A large-scale open ecosystem for robot navigation and manipulation. *arXiv preprint arXiv:2602.11337*, 2026.

[26] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[27] Zekai Gu, Rui Yan, Jiahao Lu, Peng Li, Zhiyang Dou, Chenyang Si, Zhen Dong, Qifeng Liu, Cheng Lin, Ziwei Liu, Wenping Wang, and Yuan Liu. Diffusion as shader: 3d-aware video diffusion for versatile video generation control. *arXiv preprint arXiv:2501.03847*, 2025.

[28] Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=vhFu1Acb0xb.

[29] Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. Gaia-1: A generative world model for autonomous driving. *arXiv preprint arXiv:2309.17080*, 2023.

[30] Chenhao Zheng, Ayush Shrivastava, and Andrew Owens. EXIF as Language: Learning Cross-Modal Associations Between Images and Camera Metadata. In *CVPR*, 2023. URL https://arxiv.org/abs/2301.04647.

[31] Chenhao Zheng, Jieyu Zhang, Aniruddha Kembhavi, and Ranjay Krishna. Iterated Learning Improves Compositionality in Large Vision-Language Models. In *CVPR*, 2024. URL https://arxiv.org/abs/2404.02145.

[32] Gaoyue Zhou, Hengkai Pan, Yann Lecun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 79115–79135. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/zhou25t.html.

[33] Mahmoud Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. Technical report, FAIR at Meta, 2025.

[34] Wei Mao, Miaomiao Liu, Mathieu Salzmann, and Hongdong Li. Learning trajectory dependencies for human motion prediction, 2020. URL https://arxiv.org/abs/1908.05436.

[35] Tomoya Yoshida, Shuhei Kurita, Taichi Nishimura, and Shinsuke Mori. Generating 6dof object manipulation trajectories from action description in egocentric vision. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 17370–17382, 2025.

[36] Wenlong Huang, Yu-Wei Chao, Arsalan Mousavian, Ming-Yu Liu, Dieter Fox, Kaichun Mo, and Li Fei-Fei. Pointworld: Scaling 3d world models for in-the-wild robotic manipulation. *arXiv preprint arXiv:2601.03782*, 2026.

[37] Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, Joao Carreira, and Andrew Zisserman. Tapir: Tracking any point with per-frame initialization and temporal refinement. *arXiv preprint arXiv:2306.08637*, 2023.

[38] Carl Doersch, Pauline Luc, Yi Yang, Dilara Gokay, Skanda Koppula, Ankush Gupta, Joseph Heyward, Ignacio Rocco, Ross Goroshin, João Carreira, and Andrew Zisserman. Bootstap: Bootstrapped training for tracking-any-point. *arXiv preprint arXiv:2402.00847*, 2024.

[39] Adam W. Harley, Yang You, Xinglong Sun, Yang Zheng, Nikhil Raghuraman, Yunqi Gu, Sheldon Liang, Wen-Hsuan Chu, Achal Dave, Pavel Tokmakov, Suya You, Rares Ambrus, Katerina Fragkiadaki, and Leonidas J. Guibas. Alltracker: Efficient dense point tracking at high resolution. *arXiv preprint arXiv:2506.07310*, 2025.

[40] Jiahui Huang, Qunjie Zhou, Hesam Rabeti, Aleksandr Korovko, Huan Ling, Xuanchi Ren, Tianchang Shen, Jun Gao, Dmitry Slepichev, Chen-Hsuan Lin, Jiawei Ren, Kevin Xie, Joydeep Biswas, Laura Leal-Taixé, and Sanja Fidler. Vipe: Video pose engine for 3d geometric perception. *arXiv preprint arXiv:2508.10934*, 2025.

[41] Zhengqi Li, Richard Tucker, Forrester Cole, Qianqian Wang, Linyi Jin, Vickie Ye, Angjoo Kanazawa, Aleksander Holynski, and Noah Snavely. Megasam: Accurate, fast and robust structure and motion from casual dynamic videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10486–10496, 2025.

[42] Yuxi Xiao, Jianyuan Wang, Nan Xue, Nikita Karaev, Yuri Makarov, Bingyi Kang, Xing Zhu, Hujun Bao, Yujun Shen, and Xiaowei Zhou. Spatialtrackerv2: 3d point tracking made easy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025. URL https://arxiv.org/abs/2507.12462.

[43] Haiwen Feng, Junyi Zhang, Qianqian Wang, Yufei Ye, Pengcheng Yu, Michael J. Black, Trevor Darrell, and Angjoo Kanazawa. St4rtrack: Simultaneous 4d reconstruction and tracking in the world. *arXiv preprint arXiv:2504.13152*, 2025.

[44] Chuhan Zhang, Guillaume Le Moing, Skanda Koppula, Ignacio Rocco, Liliane Momeni, Junyu Xie, Shuyang Sun, Rahul Sukthankar, Joëlle K. Barral, Raia Hadsell, Zoubin Ghahramani, Andrew Zisserman, Junlin Zhang, and Mehdi S. M. Sajjadi. Efficiently reconstructing dynamic scenes one d4rt at a time, 2025. URL https://arxiv.org/abs/2512.08924.

[45] Shikhar Bahl, Russell Mendonca, Lili Chen, Unnat Jain, and Deepak Pathak. Affordances from human videos as a versatile representation for robotics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[46] Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, et al. Molmoact2: Action reasoning models for real-world deployment. *arXiv preprint arXiv:2605.02881*, 2026.

[47] Homanga Bharadhwaj, Debidatta Dwibedi, Abhinav Gupta, Shubham Tulsiani, Carl Doersch, Ted Xiao, Dhruv Shah, Fei Xia, Dorsa Sadigh, and Sean Kirmani. Gen2act: Human video generation in novel scenarios enables generalizable robot manipulation. *arXiv preprint arXiv:2409.16283*, 2024.

[48] Guangrun Li, Yaoxu Lyu, Zhuoyang Liu, Chengkai Hou, Jieyu Zhang, and Shanghang Zhang. H2r: A human-to-robot data augmentation for robot pre-training from videos. *arXiv preprint arXiv:2505.11920*, 2025.

[49] Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X. Lee, Maria Bauza, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, Antoine Laurens, Claudio Fantacci, Valentin Dalibard, Martina Zambelli, Murilo Martins, Rugile Pevceviciute, Michiel Blokzijl, Misha Denil, Nathan Batchelor, Thomas Lampe, Emilio Parisotto, Konrad Żołna, Scott Reed, Sergio Gómez Colmenarejo, Jon Scholz, Abbas Abdolmaleki, Oliver Groth, Jean-Baptiste Regli, Oleg Sushkov, Tom Rothörl, José Enrique Chen, Yusuf Aytar, Dave Barker, Joy Ortiz, Martin Riedmiller, Jost Tobias Springenberg, Raia Hadsell, Francesco Nori, and Nicolas Heess. Robocat: A self-improving generalist agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023.

[50] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[51] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

[52] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[53] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[54] Russell Mendonca, Shikhar Bahl, and Deepak Pathak. Structured world models from human videos. *arXiv preprint arXiv:2308.10901*, 2023.

[55] Raktim Gautam Goswami, Amir Bar, David Fan, Tsung-Yen Yang, Gaoyue Zhou, Prashanth Krishnamurthy, Michael Rabbat, Farshad Khorrami, and Yann LeCun. World models for learning dexterous hand-object interactions from human videos. *arXiv preprint arXiv:2512.13644*, 2025.

[56] Quentin Garrido, Tushar Nagarajan, Basile Terver, Nicolas Ballas, Yann LeCun, and Michael Rabbat. Learning latent action world models in the wild. *arXiv preprint arXiv:2601.05230*, 2026.

[57] Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from videos. In *International Conference on Learning Representations (ICLR)*, 2025.

[58] Ruihan Yang, Qinxi Yu, Yecheng Wu, Rui Yan, Borui Li, An-Chieh Cheng, Xueyan Zou, Yunhao Fang, Xuxin Cheng, Ri-Zhao Qiu, et al. Egovla: Learning vision-language-action models from egocentric human videos. *arXiv preprint arXiv:2507.12440*, 2025.

[59] Chenhao Zheng, Jieyu Zhang, Mohammadreza Salehi, Ziqi Gao, Vishnu Iyengar, Norimasa Kobori, Quan Kong, and Ranjay Krishna. One Trajectory, One Token: Grounded Video Tokenization via Panoptic Sub-object Trajectory. *arXiv preprint arXiv:2505.23617*, 2025. URL https://arxiv.org/abs/2505.23617.

[60] Chenhao Zheng, Jieyu Zhang, Jianing Zhang, Weikai Huang, Ashutosh Kumar, Quan Kong, Oncel Tuzel, Chun-Liang Li, and Ranjay Krishna. TrajTok: Learning Trajectory Tokens enables better Video Understanding. *arXiv preprint arXiv:2602.22779*, 2026. URL https://arxiv.org/abs/2602.22779.

[61] Skand Peri, Hung Nguyen, Chanho Kim, Li Fuxin, and Stefan Lee. 3D Point World Models: Point Completion Enables More Accurate Dynamics Learning. *arXiv preprint arXiv:2607.00148*, 2026. URL https://arxiv.org/abs/2607.00148.

[62] Bardienus P. Duisterhof, Kaifeng Zhang, Adam Hung, Bowen Wen, Stan Birchfield, Yunzhu Li, Deva Ramanan, and Jeffrey Ichnowski. PointZero: 3D Point Track Completion for Learning Transferable 3D Dynamics. *arXiv preprint arXiv:2609.19142*, 2026. URL https://arxiv.org/abs/2609.19142.

[63] Baoyu Li, Xinchen Yin, Mengying Lin, Yixin Zhang, and Danfei Xu. EgoWAM: World Action Models Beyond Pixels with In-the-Wild Egocentric Human Data. *arXiv preprint arXiv:2607.08436*, 2026. URL https://arxiv.org/abs/2607.08436.

[64] Adam Hung, Bardienus P. Duisterhof, Deva Ramanan, and Jeffrey Ichnowski. Modality-Autoregressive World-Action Models. *arXiv preprint arXiv:2609.17524*, 2026. URL https://arxiv.org/abs/2609.17524.

[65] Christopher Clark, Jieyu Zhang, Zixian Ma, Jae Sung Park, Mohammadreza Salehi, Rohun Tripathi, Sangho Lee, Zhongzheng Ren, Chris Dongjoo Kim, Yinuo Yang, Vincent Shao, Yue Yang, Weikai Huang, Ziqi Gao, Taira Anderson, Jianrui Zhang, Jitesh Jain, George Stoica, Winson Han, Ali Farhadi, and Ranjay Krishna. Molmo2: Open weights and data for vision-language models with video understanding and grounding. *arXiv preprint arXiv:2601.10611*, 2026.

[66] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2024. URL https://arxiv.org/abs/2303.04137.

[67] Abhay Deshpande, Maya Guru, Rose Hendrix, Snehal Jauhri, Ainaz Eftekhar, Rohun Tripathi, Max Argus, Jordi Salvador, Haoquan Fang, Matthew Wallingford, Wilbert Pumacay, Yejin Kim, Quinn Pfeifer, Ying-Chun Lee, Piper Wolters, Omar Rayyan, Mingtong Zhang, Jiafei Duan, Karen Farley, Winson Han, Eli VanderBilt, Dieter Fox, Ali Farhadi, Georgia Chalvatzaki, Dhruv Shah, and Ranjay Krishna. Molmobot: Large-scale simulation enables zero-shot manipulation. *arXiv preprint arXiv:2603.16861*, 2026.

[68] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[69] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

[70] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[71] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[72] Ryan Hoque, Peide Huang, David J. Yoon, Mouli Sivapurapu, and Jian Zhang. Egodex: Learning dexterous manipulation from large-scale egocentric video. *arXiv preprint arXiv:2505.11709*, 2025.

[73] Toby Perrett, Ahmad Darkhalil, Saptarshi Sinha, Omar Emara, Sam Pollard, Kranti Kumar Parida, Kaiting Liu, Prajwal Gatti, Siddhant Bansal, Kevin Flanagan, et al. Hd-epic: A highly-detailed egocentric video dataset. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 23901–23913, 2025.

[74] Ropedia AI. Xperience-10m: A large-scale egocentric multimodal dataset with structured 3d/4d annotations. https://huggingface.co/datasets/ropedia-ai/xperience-10m, 2026. Hugging Face dataset.

[75] Linyi Jin, Richard Tucker, Zhengqi Li, David Fouhey, Noah Snavely, and Aleksander Holynski. Stereo4d: Learning how things move in 3d from internet stereo videos. *arXiv preprint arXiv:2412.09621*, 2024.

[76] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[77] Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. Sam 3: Segment anything with concepts. *arXiv preprint arXiv:2511.16719*, 2025.

[78] Christopher Clark, Yue Yang, Jae Sung Park, Zixian Ma, Jieyu Zhang, Rohun Tripathi, Mohammadreza Salehi, Sangho Lee, Taira Anderson, Winson Han, et al. Molmopoint: Better pointing for vlms with grounding tokens. *arXiv preprint arXiv:2603.28069*, 2026.

[79] Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764–766, 2013.

[80] Linjie Yang, Yuchen Fan, and Ning Xu. Video instance segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5188–5197, 2019.

[81] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017.

[82] Zeyi Liu, Shuang Li, Eric Cousineau, Siyuan Feng, Benjamin Burchfiel, and Shuran Song. Geometry-aware 4d video generation for robot manipulation. *arXiv preprint arXiv:2507.01099*, 2025.

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

[84] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[85] Vincent A Traag, Ludo Waltman, and Nees Jan van Eck. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1):5233, 2019.

[86] 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.

Appendix

A Qualitative examples 19
B MolmoMotion-1M Data Generation Details 19
B.1 Source video corpora 19
B.2 Recaptioning and object name extraction 19
B.3 Semantic grounding and 3D lifting 20
B.4 Trajectory filtering and smoothing 20
C PointMotionBench 21
C.1 HOT3D 21
C.2 WorldTrack 22
C.3 DAVIS 22
C.4 Evaluation Protocol 23
D Model Implementation Details 23
D.1 Architecture 23
D.2 Prompt format 24
D.3 Training hyperparameters 24
D.4 Flow-matching objective and inference 24
E Model Ablations 25
F Robotics Transfer Settings 26
F.1 MolmoSpaces pick-and-place 26
F.2 DROID trajectory finetuning 27
G Video generation experiment details 27
G.1 Methods compared 28
G.2 Evaluation protocol 28

A Qualitative examples

We provide additional qualitative results for two downstream applications of MolmoMotion: real-robot trajectory prediction on DROID, and trajectory-conditioned video generation.

Real-robot trajectory prediction on DROID. Fig. 9 shows MolmoMotion's predicted future 3D trajectories on held-out DROID clips after finetuning on real-robot video. MolmoMotion plans accurate point trajectories across diverse manipulation scenes, objects, and tasks.

Trajectory-conditioned video generation. Fig. 10 and Fig. 11 compare videos generated by DaS [27] conditioned on MolmoMotion-predicted 3D trajectories with the unconditioned CogVideoX-5B and Wan-14B baselines on held-out PointMotionBench prompts. MolmoMotion-guided videos exhibit more physically plausible object motion, preserve manipulated-object identity better, and follow the prompted action more faithfully than the unconditioned baselines.

Together these examples illustrate that MolmoMotion's 3D trajectories transfer to real-world robot data and serve as an effective control signal for downstream video generation, complementing the quantitative results in the main paper.

B MolmoMotion-1M Data Generation Details

This appendix expands the annotation pipeline summarized in Sec. 4.1. Given a public video and its caption, the pipeline recovers an object phrase that names the moving object, grounds the phrase to a set of query points on the object, and lifts those points into a metric 3D world frame. We describe the source corpora (§B.1), recaptioning and object-name extraction (§B.2), object grounding and 3D lifting (§B.3), and trajectory filtering and smoothing (§B.4).

B.1 Source video corpora

MolmoMotion-1M aggregates seven source corpora (Tab. 3) that together cover egocentric and third-person human manipulation, simulated and robot manipulation, and in-the-wild scenes. The five manipulation corpora ship with action descriptions or task templates, which we use directly. YT-VIS supplies object masks but no captions, so we re-caption each clip from the video. Stereo4D contributes short third-person clips with metric stereo depth, expanding coverage to outdoor scenes and deformable subjects.

Corpus	Motion clips	Domain	Action-description source
EgoDex	∼160K	egocentric, human	VLM re-caption
HD-EPIC	∼21K	egocentric, human	narrations
Xperience	∼500K	3rd-person, human	metadata
MolmoSpaces	∼185K	3rd-person, sim	task templates
DROID	∼27K	3rd-person, robot	language instructions
YT-VIS	∼2K	3rd-person, wild	VLM caption
Stereo4D	∼70K	3rd-person, wild	paper captions

Table 3: Source corpora used to construct MolmoMotion-1M.

B.2 Recaptioning and object name extraction

For corpora that do not ship object-level captions (EgoDex, where the original task labels are too coarse to ground a specific entity, and YT-VIS), we generate a one-sentence visual description of each clip with Molmo2-8B [65]. The full 15 FPS re-encoded video is passed to Molmo2-8B. The prompt is:

```
Watch this video carefully. Describe the manipulation action you
observe in exactly this format: [action verb] [specific object with
color/material/shape] [preposition and location if present].
Examples: "pick up red ceramic coffee mug", "place blue plastic
bottle on table". Be specific about the object -- include its color,
```

```
material, and shape as you see them.
Output only the short description, no extra words.
```

Then we extract the manipulated object as a noun phrase with Qwen3-0.6B [76].

B.3 Semantic grounding and 3D lifting

Given an object phrase, we localize the entity, segment it, sample query points, track those points across the video, and lift the tracks to a metric 3D world frame.

Localization with motion-aware prompting. We first localize the object as a 2D point with MolmoPoint-8B [78], and then convert that point into a segmentation mask with SAM 3 [77]. The non-trivial choice here is the prompt given to MolmoPoint. Conditioning the prompt on the agent and the action, we give the model the following prompt: *"point to the {obj} gripped and picked up by the hand"* for human-manipulation corpora, and *"track the {obj}"* for in-the-wild video. This is substantially more reliable than a bare *"point to the {obj}"* prompt. The motion cue disambiguates vague phrases like *"an object on the table"* by biasing MolmoPoint toward the moving entity rather than a static distractor that matches the phrase equally well.

Point prompt for SAM 3. We feed the 2D point returned by MolmoPoint to SAM 3. We then sample $N = 100$ query points per mask using K-means cluster centers on the mask pixel coordinates, so points are spread across the object surface rather than concentrated near the centroid.

2D tracking and lifting. We propagate query points through the video with AllTracker [39], which yields temporally persistent 2D tracks and per-frame visibility scores. We run ViPE [40] on the same video to estimate per-frame metric depth, intrinsics, and camera-to-world poses in a single pass. Each visible 2D track location is back-projected with the estimated depth and intrinsics, then transformed by the corresponding camera pose into the world frame anchored at the query-time camera.

B.4 Trajectory filtering and smoothing

Lifted 3D trajectories are corrupted by noise. We make this prior precise as follows.

Anchor tracks and inconsistency score. We select a small set of anchor tracks $\mathcal{A}$ per object (sixteen in our pipeline) that are visible throughout the clip and have low per-frame velocity, taking these as the most reliable estimate of the object's body motion. For every other track n we measure how its distance to each anchor varies over time:

$$e_n(t) \;=\; \mathrm{median}_{k\in\mathcal{A}} \,\big|\, \|\tilde{\mathbf{p}}_t^n - \tilde{\mathbf{p}}_t^k\|_2 \;-\; \bar{d}_{nk} \,\big|, \qquad \bar{d}_{nk} \;=\; \mathrm{median}_t \,\|\tilde{\mathbf{p}}_t^n - \tilde{\mathbf{p}}_t^k\|_2. \tag{1}$$

A rigid or near-rigid object keeps its inter-point distances roughly constant, so $e_n(t)$ is large precisely at frames where track n has drifted in 2D, had its depth pulled to the background, or been displaced by a pose error.

Trust weights and the choice of scale. We convert the inconsistency score into a per-frame trust weight $w_{t,n} = \exp\!\big(-e_n(t)/s_n\big)$ with a per-track scale s_n. We use $s_n = \mathrm{median}_t(e_n)$, which behaves well under uniform noise while still suppressing per-frame outliers within a track.

Spatial auto-split. A single object phrase sometimes resolves to two physically separate instances, and mixing tracks across instances violates the rigid-object assumption behind $e_n(t)$. We therefore cluster points by their temporal-mean 3D position with mean-shift and run anchor selection, trust scoring, and smoothing independently per sub-cluster.

Track-level outlier drop. Within each sub-cluster, we drop tracks whose mean trust $\bar{w}_n$ is a MAD-scaled z-score below the sub-cluster median.

Depth-ray smoothing. Surviving tracks are still noisy along the depth axis. Following Stereo4D [75], we re-parametrize the 3D point at frame t as $\mathbf{p}_t^n = \mathbf{c}_t + \lambda_t^n \mathbf{r}_t^n$, where $\mathbf{c}_t$ is the camera center and $\mathbf{r}_t^n$ is the unit ray through the 2D track location, and we optimize the depth scalars $\{\lambda_t^n\}$ with two competing objectives:

$$\min_{\{\lambda_t^n\}} \underbrace{\sum_{t,n} w_{t,n}^2 \left\| \mathbf{c}_t + \lambda_t^n \mathbf{r}_t^n - \tilde{\mathbf{p}}_t^n \right\|_2^2}_{\text{trust-weighted pin to lifted point}} \;+\; \beta \underbrace{\sum_n \sum_{\Delta\in\{1,3,5\}} \sum_t \left\| \mathbf{p}_{t+\Delta}^n - 2\mathbf{p}_t^n + \mathbf{p}_{t-\Delta}^n \right\|_2^2}_{\text{multi-stride acceleration penalty}}. \tag{2}$$

The pin term is gated by the trust weights, so untrusted frames are free to slide along the ray while trusted frames remain anchored; the acceleration term enforces smooth motion in 3D. Two design choices are worth explaining. First, the acceleration term is summed over multiple strides $\Delta \in \{1, 3, 5\}$ rather than only $\Delta = 1$. A single-stride second difference penalizes only sharp single-frame jitter, but our depth errors include slow ramps over five to ten frames (e.g. when ViPE depth gradually bleeds onto a textureless background); penalizing acceleration at $\Delta = 3$ and $\Delta = 5$ in addition to $\Delta = 1$ catches these slower drifts. Second, we optimize with first-order gradient descent rather than LBFGS. LBFGS converges faster on well-behaved tracks but diverges on near-degenerate tracks, so the more conservative optimizer is the safer default at corpus scale.

C PointMotionBench

PointMotionBench evaluates object-centric 3D motion forecasting across three diverse datasets; Tab. 4 summarizes the per-source clip counts, resolutions, and frame rates. The following sections detail the data preparation pipelines and evaluation protocol (including metric definitions).

Dataset	Clips	Resolution	FPS
HOT3D	497	1408×1408	30
WorldTrack	155	split-dependent†	30
DAVIS	90	854×480	24

Table 4: Source datasets used to construct PointMotionBench. †`adt_mini` 512×512; `ds_mini` 1280×720; `po_mini` 960×540; `pstudio_mini` 640×360.

Exclusions. We exclude 10 HOT3D clips that contain no ground-truth-visible query points at frame 0 (AllTracker crashes at initialization with an empty point set). For WorldTrack, we exclude the `tum` split entirely (no 3D tracks shipped) and the 27 `po_mini dancingroom1_3rd*` sequences that exhibit cluttered motion patterns. No DAVIS sequences are excluded. The same exclusion lists apply across all three baselines so that comparisons are always over identical clip sets, leaving 497 HOT3D, 155 WorldTrack, and 90 DAVIS clips.

Human verification. For HOT3D and WorldTrack, we view each clip alongside the annotated trajectories and confirm that (i) the foreground object is correctly identified and (ii) the one-sentence model-generated description accurately reflects the observed motion and action; clips that fail are re-annotated or excluded. For DAVIS, we directly wrote the descriptions during trajectory review rather than reviewing model-generated captions.

Contamination check. PointMotionBench contains no clip-level near-duplicates with the training data of the evaluated models. All per-clip captions are either written directly by human annotators or generated by Molmo2-8B and subsequently verified by annotators, ensuring no captions are sourced from existing dataset metadata that evaluated models may have been trained on.

C.1 HOT3D

HOT3D [21] Aria provides 1,415 source clips with per-frame 3D object pose annotations as fitted mesh models. We sample 2,000 surface points per object at its first active frame and propagate them forward with per-frame pose transforms, yielding world-space trajectories and their corresponding pixel-space projections.

Each source clip typically contains one or two manipulated objects alongside several static ones. We split every clip into single-moving-object sub-clips by detecting per-object motion windows: for each object, we threshold a per-frame body-speed signal (median surface-point displacement) at 0.005 m/frame ($\approx$15 cm/s at 30 fps), bridge single-frame gaps, and drop runs shorter than 0.5 s. This yields 2,534 sub-clips, each covering exactly one continuously moving foreground object. We then uniformly subsample one sub-clip per group of five (random selection within the group), yielding 507 clips; the 10 with no ground-truth-visible query points at frame 0 are excluded, leaving 497.

C.2 WorldTrack

WorldTrack [43] bundles four indoor motion-capture splits mixing egocentric and third-person viewpoints: `adt_mini`, `ds_mini`, `po_mini`, and `pstudio_mini` (resolutions listed in Tab. 4). Unlike HOT3D, source point tracks are not pre-assigned to objects. We augment the raw data through a four-step pipeline: (1) identify dynamic foreground points, (2) cluster them into per-object groups, (3) segment sequences into motion-coherent sub-clips, and (4) generate a one-sentence caption per clip.

Dataset	Filter type	Window(s)	Threshold
`adt_mini`	per-frame	3, 5 fr	0.25%
`po_mini`	global	—	1.0%
`pstudio_mini`	global+per-fr	3, 15 fr	1.0%
`ds_mini`	per-frame	10 fr	0.75%

Table 5: Per-dataset motion-filter settings for dynamic point extraction of WorldTrack data.

Dynamic point extraction. We lift camera-space tracks into a shared world coordinate frame using per-frame extrinsics; for `pstudio_mini` (fixed camera), camera-space coordinates serve directly as world-space. A point is classified as dynamic if its world-space displacement over a look-back window exceeds a scene-normalized threshold (a per-dataset fraction of the scene's 1st-to-99th percentile spatial extent; see Tab. 5). Masks from multiple window lengths are OR-ed together, and each active segment is extended backward to capture motion onset.

Object clustering. We cluster the dynamic points into per-object groups in two stages.

Stage 1 — Temporal clustering (Leiden [85]). We build a pairwise affinity graph over the dynamic points. For each pair (i, j), let $\mathcal{T}_{\text{co}}$ be the set of co-visible frames and $\mathbf{v}_i(t)$ the unit-normalized frame-to-frame velocity of point i at frame t, with zero-velocity frames contributing 0. The affinity is

$$W(i,j) \;=\; \frac{1}{|\mathcal{T}_{\text{co}}|} \sum_{t \,\in\, \mathcal{T}_{\text{co}}} \langle \mathbf{v}_i(t),\, \mathbf{v}_j(t) \rangle.$$

Pairs with no co-visible support or below a similarity threshold are zeroed out. Additionally, any pair whose 3D world-space distance exceeds a fixed fraction of scene extent at any sampled keyframe is forced apart—a single violating frame is conclusive evidence of distinct objects, since rigid-body points cannot drift far apart in 3D. We run Leiden community detection on the resulting graph and iteratively merge undersized clusters into their most temporally similar neighbor. Per-dataset minimum cluster sizes are 1 (`adt_mini`, `pstudio_mini`), 3 (`ds_mini`), and 5 (`po_mini`), set by visual inspection.

Stage 2 — Segmentation merging (SAM 2 [86]). Temporal clustering occasionally splits a single physical object across multiple clusters. For each cluster, we designate a representative point and sample three keyframes evenly across its active window, then query SAM 2. Two clusters are merged only when their representatives are spatially proximate and co-occur within the same SAM 2 mask in at least one keyframe without ever appearing in separate masks. The criterion is intentionally conservative: a single frame of visible separation blocks the merge, since merging distinct objects corrupts ground truth irreversibly.

Sub-clip extraction and captioning. We group objects with overlapping active spans, bridge gaps of fewer than three frames, split on longer gaps, and merge sub-clips shorter than 2 s into their nearest neighbor. Each sub-clip is captioned by Molmo2-8B [65], prompted to produce a one-sentence egocentric description (e.g. *"a hand moves a keyboard across the desk"*); captions are subsequently verified by human annotators.

C.3 DAVIS

DAVIS [81] provides RGB frames and per-object segmentation masks but no 3D ground truth. We generate 3D trajectories by running the MolmoMotion-1M annotation pipeline, seeding query points from the DAVIS masks and lifting them through ViPE depth and camera pose estimation. We verify the resulting tracks and write the per-clip action descriptions directly during trajectory review.

C.4 Evaluation Protocol

Conditioning and future split. All metrics are computed on future frames only. With $T_{\text{cond}} \in \{1, 3\}$, the model observes frames $0, \ldots, T_{\text{cond}} - 1$ and predicts frames $T_{\text{cond}}, \ldots, T - 1$. Scoring is restricted to points visible at frame 0, as these are the only query positions supplied to the model. The evaluation mask is

$$v_{\text{eval}}[t, n] \;=\; v[t, n] \;\wedge\; v[0, n],$$

applied before every metric. Let $\mathcal{S} = \{(t, n) : t \geq T_{\text{cond}},\ v_{\text{eval}}[t, n] = 1\}$ be the set of scored (frame, point) pairs, and let $N_{\text{eval}} = \sum_n v[0, n]$ be the per-clip count of evaluated points.

Temporal alignment. HOT3D and WorldTrack ground truth tracks are at 30 fps, and DAVIS ground truth tracks are at 24 fps; Wan2.2 and Cosmos predictions are at 24 fps. For HOT3D and Worldtrack, we resample these predictions to the ground truth timebase by linearly interpolating positions at source time $t_{\text{src}} = 0.8\, t_{\text{gt}}$ and nearest-neighbor-resampling visibility; only the $\min(T_{\text{gt}}, T_{\text{resampled}})$ overlapping frames are scored. Track2Act outputs exactly 8 frames at ground truth indices $\text{round}(i\,(T-1)/7)$ for $i = 0, \ldots, 7$; we match ground truth frames at the same indices directly, with no interpolation.

Metrics. All metrics are computed in 3D world space. Let $\hat{p}(t, n)$ and $q(t, n)$ denote the predicted and ground-truth positions of query point n at frame t (in metres; 1 unit = 1 m throughout).

ADE (Average Displacement Error):

$$\text{ADE} \;=\; \frac{1}{|\mathcal{S}|} \sum_{(t,n)\,\in\,\mathcal{S}} \|\hat{p}(t, n) - q(t, n)\|_2$$

FDE (Final Displacement Error), evaluated at the last frame $T - 1$:

$$\text{FDE} \;=\; \frac{1}{|\mathcal{S}_T|} \sum_{n\,\in\,\mathcal{S}_T} \|\hat{p}(T-1, n) - q(T-1, n)\|_2, \qquad \mathcal{S}_T = \{n : v_{\text{eval}}[T-1, n] = 1\}.$$

PWT (Percentage Within Threshold) at threshold δ:

$$\text{PWT}(\delta) \;=\; \frac{\big|\{(t, n) \in \mathcal{S} : \|\hat{p}(t, n) - q(t, n)\|_2 < \delta\}\big|}{|\mathcal{S}|}$$

We report $\overline{\text{PWT}}$, the mean of $\text{PWT}(\delta)$ over $\delta \in \{0.01, 0.02, 0.05, 0.10, 0.20\}$ m. ADE and FDE are in meters (lower is better); $\overline{\text{PWT}} \in [0, 1]$ (higher is better).

D Model Implementation Details

We expand the architecture and training recipe here.

D.1 Architecture

Vision-language backbone We initialize from the public `Molmo2-4B-Pretrain` checkpoint. The vision encoder is a SigLIP2 ViT operating on 378×378 RGB inputs at 14-pixel patch size, producing a 27×27 grid of 1152-D patch tokens per frame. The Molmo2 connector pools the patch grid by 3×3 and projects the pooled tokens through an MLP into the LLM hidden dimension of 2560. The language model is Qwen3-4B. All backbone parameters are trained end-to-end.

Decoder heads The autoregressive variant uses the unmodified Molmo2 LM head plus a small regex parser that maps the answer span back to coordinates at inference. The flow-matching variant uses a DiT trajectory expert with 36 blocks (one per LM layer). Each block applies a self-attention over the trajectory-token tensor of shape $(N, H+T, 3)$ followed by a cross-attention whose keys and values come from the corresponding LM-layer hidden states and whose queries come from the trajectory tokens. RoPE is applied along both the point-index and the frame-index axes so the same self-attention can distinguish "the same point at different times" from "different points at the same time" without architectural specialization.

D.2 Prompt format

The autoregressive variant serializes everything as a single multimodal prompt. Image tokens for the anchor frame and the H history frames are inserted by Molmo2's video preprocessor at the front; the textual portion follows. With $N = 8$ and $T = 32$, an example prompt is

```
Predict the future 3D point coordinates of 8 points over 32
timestamps,
given action:  "open the drawer",
2d history point features:  "<points anchor 1
<2d_feat_start><|point_feat|><2d_feat_end>
                                          2
<2d_feat_start><|point_feat|><2d_feat_end>
                                          ...  8
<2d_feat_start><|point_feat|><2d_feat_end>/>",
and history 3d point coordinates:
    "<tracks coords='0.0 1 0 0 0  2 12 -3 0 ... 8 5 4 -1'>3d object
history</tracks>".
```

The supervised answer span is

```
<tracks coords="1.0 1 4 -8 2  2 17 -10 5 ... 8 9 7 -3;
                2.0 1 8 -16 4 ...;
                ...
                32.0 1 ...">3d object trajectories</tracks>
```

Each per-frame block lists $(n,\ q_x,\ q_y,\ q_z)$ quadruples where n is the integer point identifier and (q_x, q_y, q_z) are the millimetre-quantised anchor-relative deltas $\bar{\boldsymbol{\delta}}_t^n$. Only points visible at frame t are emitted; occluded points are not imputed. At inference, the model is asked to generate this answer span autoregressively under greedy decoding, and the parser re-assembles $\hat{\mathbf{P}}_{t_0+1:t_0+T}$ from the recovered quadruples.

D.3 Training hyperparameters

We train with AdamW using the Molmo2 supervised-fine-tuning defaults ($\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay 0.1). The learning rate is warmed up linearly over the first 1K steps to its peak and then cosine-decayed to $0.1\times$ peak. Activations are bf16 with fp32 master weights; the model is distributed with FSDP2 full-shard across 16 H100 GPUs at per-device batch 16 for a global batch size of 256. Gradients are clipped at maximum-norm 1.0. Multi-dataset mixing across the six MolmoMotion-1M sources is square-root-weighted by per-dataset clip count. The autoregressive cross-entropy is computed on the answer span only, with the prompt portion (image, text, point-feature, and history-coordinate tokens) masked out of the loss; the flow-matching MSE is computed on the future positions only, leaving the clean history portion of the trajectory tensor unsupervised.

D.4 Flow-matching objective and inference

We give the full flow-matching specification briefly described in Sec. 3.2. The flow-matching head predicts the future anchor-relative trajectory tensor $\boldsymbol{\delta}_{t_0+1:t_0+T} \in \mathbb{R}^{N\times T\times 3}$ in continuous metric coordinates, conditioned on the multimodal context $\mathcal{C}$ (image, text, point-feature, and history-coordinate tokens) and on the clean initial 3D query coordinates $\{\boldsymbol{\delta}_{t_0}^n\}_{n=1}^N$.

Forward interpolation. For each training example we draw a flow timestep $\tau \sim \mathcal{U}(0,1)$ and a noise tensor $\epsilon \in \mathbb{R}^{N\times T\times 3}$ whose entries are i.i.d. standard Gaussian. The interpolated trajectory is

$$\boldsymbol{\delta}_\tau \;=\; (1-\tau)\,\epsilon \;+\; \tau\,\boldsymbol{\delta}_{t_0+1:t_0+T}, \tag{3}$$

which slides linearly from pure Gaussian noise at $\tau = 0$ to the clean ground-truth future trajectory at $\tau = 1$.

Velocity head. The DiT decoder v_ϕ predicts the velocity field at $(\boldsymbol{\delta}_\tau, \tau)$ in the direction of the clean future. It takes three inputs: (i) the noised trajectory tensor with clean history concatenated to the noisy future, RoPE-tagged along both the point-index and the frame-index axes; (ii) the scalar τ via

Variant	HOT3D			WorldTrack			DAVIS		
	ADE↓	FDE↓	PWT↑	ADE↓	FDE↓	PWT↑	ADE↓	FDE↓	PWT↑
MolmoMotion-AR ($H = 3$)	**0.109**	**0.217**	**0.444**	**0.143**	**0.261**	**0.445**	**1.227**	**2.108**	**0.153**
without 2D point feature	0.118	0.231	0.421	0.155	0.282	0.418	1.310	2.252	0.143
Absolute coords (no delta)	0.165	0.330	0.276	0.220	0.401	0.288	1.940	3.275	0.082
without language instruction	0.158	0.318	0.291	0.215	0.392	0.305	1.890	3.182	0.092
$N = 16$ query points	0.106	0.212	0.452	0.140	0.255	0.451	1.198	2.061	0.158

Table 6: **Model ablations on PointMotionBench.** All variants use the autoregressive head with the same Molmo2-4B initialization, the same 40K+10K training schedule, and the same evaluation protocol as Tab. 1. The reference row is MolmoMotion-AR at $H = 3$. Rows 2–4 remove one design choice; row 5 doubles the per-object query count. Anchor-relative parameterization and language conditioning are the largest contributors; the 2D point feature yields a small but consistent gain; doubling N improves accuracy slightly at substantial training cost (see text).

a sinusoidal embedding added to every trajectory token; and (iii) the per-layer Molmo2 LM hidden states $\mathcal{C}$ through the per-block cross-attention described in §D.1.

Training loss. The decoder is trained with the standard flow-matching mean-squared error [71],

$$\mathcal{L}_{\mathrm{FM}} \;=\; \mathbb{E}_{\tau,\,\epsilon}\Big[\,\big\|v_\phi\big(\boldsymbol{\delta}_\tau,\,\tau,\,\{\boldsymbol{\delta}^n_{t_0}\}_{n=1}^{N},\,\mathcal{C}\big) - \big(\boldsymbol{\delta}_{t_0+1:t_0+T} - \epsilon\big)\big\|_2^2\,\Big], \tag{4}$$

where the regression target $\boldsymbol{\delta}_{t_0+1:t_0+T} - \epsilon$ is the constant velocity that drives $\boldsymbol{\delta}_\tau$ along the straight-line path of Eq. (3). The loss is masked to the future positions only, leaving the clean history portion of the trajectory tensor unsupervised.

Inference. At inference we sample $\epsilon \sim \mathcal{N}(0, I)$ and integrate the learned velocity field with $K = 10$ Euler steps, advancing τ uniformly from 0 to 1 in increments of $\Delta\tau = 0.1$. Each step evaluates v_ϕ once and updates

$$\boldsymbol{\delta}_{\tau+\Delta\tau} \;=\; \boldsymbol{\delta}_\tau \;+\; \Delta\tau \cdot v_\phi\big(\boldsymbol{\delta}_\tau,\,\tau,\,\{\boldsymbol{\delta}^n_{t_0}\}_{n=1}^{N},\,\mathcal{C}\big)\,. \tag{5}$$

The final $\boldsymbol{\delta}_1$ is added to the anchor 3D position $\mathbf{p}_{\mathrm{anc}}$ to recover the predicted future positions in the world frame, $\hat{\mathbf{P}}_{t_0+1:t_0+T}$. Because the DiT block count matches the LM layer count and one v_ϕ evaluation reuses cached LM activations, each Euler step has roughly the cost of one LM forward pass.

Comparison to autoregressive decoding. We report flow-matching numbers in Tab. 1 alongside the autoregressive variant. The autoregressive head is stronger on the deterministic point-prediction metrics ADE / FDE / PWT because each predicted coordinate is conditioned in a strict left-to-right sense on every previously emitted coordinate, which encourages temporally smooth rollouts. The flow-matching head, by contrast, samples from the conditional distribution rather than committing to a single mode, which is desirable in settings where the action description leaves multiple plausible futures.

E Model Ablations

We ablate three load-bearing design choices in the autoregressive variant: the per-query-point 2D feature, the anchor-relative coordinate parameterization, and the language-instruction conditioning. All ablations use the autoregressive variant with everything else held identical to the main model.

Ablations. The four ablated variants are constructed as follows. (i) *Without 2D point feature.* The grid-sampled per-query-point feature is omitted to test if 2D point feature is useful. (ii) *Absolute coordinates.* We replace the anchor-relative deltas $\boldsymbol{\delta}^n_t$ with absolute world-frame positions $\mathbf{p}^n_t$ in both the prompt's history block and the answer span. The 1 mm quantization grid is preserved; only the coordinate origin changes. This isolates the value of the anchor-relative parameterization introduced in Sec. 3.2. (iii) *Without language instruction.* The action caption a is replaced with a single fixed token ("`motion`"); image, text, point-feature, and history-coordinate tokens are unchanged. This isolates the value of language conditioning. (iv) $N = 16$ *query points.* We double the number of

query points sampled per object from $N = 8$ to $N = 16$, leaving every other hyperparameter unchanged. This tests whether denser per-object coverage improves prediction accuracy and quantifies the cost of going beyond our default.

Results. Tab. 6 reports 3D ADE, FDE, and PWT on the three PointMotionBench splits. Removing the anchor-relative parameterization produces the largest single drop ($\approx 50\%$ on ADE/FDE across all splits), making absolute coordinates the strongest signal of design importance. Removing the language instruction produces a comparable drop, indicating that the action description does substantially more than disambiguate the object: it provides the direction prior the model relies on when the visual context alone leaves the future ambiguous, with the largest hit on DAVIS where intent is hardest to infer from a single anchor frame. The 2D point feature contributes a smaller but consistent gain (5–8% on ADE/FDE, $\approx 5\%$ on PWT) and is most helpful on DAVIS, where the manipulated object is small relative to the frame. Doubling the per-object query count from $N = 8$ to $N = 16$ improves accuracy by roughly 2–3% on ADE/FDE and $\approx 2\%$ on PWT across all three splits. The gain is small because the eight default points already cover the manipulated object's surface densely after K-means selection, and our query points are sampled to be spatially well-distributed. The cost, however, is substantial: with $T = 32$ in stage 2, the autoregressive answer span is twice as long under $N = 16$ and exceeds the 4096-token context window of the Qwen3-4B language model that backs Molmo2-4B. We therefore keep $N = 8$ as the default, which trades a small accuracy improvement for the longer prediction horizon $T = 32$. Lifting this constraint — through tokenization schemes that encode multiple coordinates per LM token, or through context-extension recipes for the backbone — is a natural next step for representing dense object coverage at long horizons; we leave it to future work.

Inference cost. The two heads share a single Molmo2-4B forward pass over the prompt and diverge only in how they decode the future trajectory. AR emits the answer span as text, so its cost grows linearly in $N{\cdot}T$. FM runs a fixed K Euler steps through the DiT trajectory expert, so its cost is independent of T. Tab. 7 reports per-clip latency at the headline ($N{=}8, T{=}32$) setting on a single A100. Flow-matching is roughly two orders of magnitude faster than autoregressive decoding, which is the regime that matters in closed-loop robotic control, trajectory-conditioned video generation, and large-scale evaluation.

Variant	Setting	Latency (s / clip)
MolmoMotion-AR	$N{=}8,\ T{=}8$	37.2
MolmoMotion-AR	$N{=}8,\ T{=}32$	148.4
MolmoMotion-FM ($K{=}10$)	$N{=}8,\ T{=}32$	**1.1**

Table 7: **Inference cost.** Per-clip latency at $H{=}3$ on a single A100. AR cost scales linearly in $N{\cdot}T$; FM cost is independent of T. Flow-matching is roughly $\mathbf{150}\times$ faster than autoregressive decoding at $T{=}32$, at the modest accuracy cost in Tab. 1.

F Robotics Transfer Settings

This appendix documents the implementation specifics of the two robotics experiments reported in Sec. 5.2: closed-loop pick-and-place on MolmoSpaces (Fig. 6a) and 3D-trajectory finetuning on DROID (Fig. 6b).

F.1 MolmoSpaces pick-and-place

Policy. The downstream policy follows the MolmoBot [67] prompt-encoder recipe. The Molmo2-4B vision-language backbone is followed by an ActionExpert head: a flow-matching transformer with 36 cross-attention blocks (one per LM layer), action dimension 8 (7 arm joints plus 1 gripper), and action horizon 16. At inference, action chunks are produced by integrating the ActionExpert's velocity field for 10 Euler steps from Gaussian noise. The backbone is initialised either from MolmoMotion (autoregressive variant after both training stages) or from the public `Molmo2-4B-Pretrain` checkpoint for the matched control.

Inputs. Each step provides three history frames per camera at sim-step deltas $\{-4, -2, 0\}$ from two cameras (egocentric exo and wrist-mounted), plus the current 8-D robot state normalised to

Metric	What it measures
Tem-Con↑	mean CLIP cosine similarity between adjacent frames
Subj-Cons↑	feature-similarity of the moving subject across frames
M-Smooth↑	smoothness of the estimated optical-flow residual
Dyn-Deg↑	fraction of clips whose flow magnitude exceeds a motion threshold
Bg-Cons↑	feature-similarity of the background region across frames

Table 8: Higher-is-better video-quality metrics used to compare the three image-to-video generators. Subj-Cons, M-Smooth, Dyn-Deg, and Bg-Cons are standard VBench dimensions; Tem-Con is computed separately from CLIP frame embeddings.

unit-quantile space using statistics fit on the training split. The textual prompt reuses the trajectory-prediction wrapper of §D.2 with an empty answer span; the LM head's cross-entropy weight is set to zero, so only the ActionExpert flow-matching MSE contributes to the gradient.

Action representation. Actions are absolute joint-position targets supervised over a 16-step chunk ($\approx$ 1.06 s of simulated time at 15 fps). At execution time we run 8 steps of the predicted chunk and re-query the policy at sim-step $+8$, matching the training-time action-time semantics. Per-joint deltas are clamped to ± 0.2 rad/step at execution as a safety guard against out-of-distribution velocity predictions.

Training. We finetune on the 20K pick-and-place episodes released with MolmoBot, restricted to `policy_phase` $\in \{5, 6, 7, 8\}$ (lift, preplace, place, retreat) so the policy is supervised only on the post-grasp portion of each trajectory. Optimization uses AdamW with the Molmo2 SFT defaults, maximum sequence length 2048, gradient clipping at norm 1.0, bf16 mixed precision, FSDP2 full-shard, per-device batch 2 with 8 gradient-accumulation micro-batches for a global batch size of 128 across 8 H100 GPUs. Training runs for 100K optimization steps; checkpoints are saved every 5K steps and evaluated periodically.

Hybrid rollout and evaluation. Closed-loop evaluation uses a hybrid rollout. The released `MolmoBot-DROID` policy drives every episode through the approach and grasp phases. The first sim step at which the simulator reports `held = true` on the pickup object marks a hand-off trigger; one execute-horizon window (8 sim steps) later, control is handed to the evaluated policy, which performs the lift, transport, and placement. Both policies output absolute joint-position targets. Each rollout runs for at most 600 sim steps ($\approx$ 40 s at 15 fps); an episode succeeds when the simulator's terminal `success` flag is true, i.e. when the pickup object lies inside the success-position threshold of its target receptacle.

F.2 DROID trajectory finetuning

Task and data. We use the same 3D point-trajectory prediction objective as in Sec. 3.2; no robot-action loss is applied. DROID wrist and shoulder videos are paired with 3D trajectories produced by running the MolmoMotion-1M annotation pipeline of Sec. 4.1 on the DROID corpus, yielding the same triplet of (object mask, action description, dense 3D trajectory) as the pretraining data. We hold out a fixed 10 percent subset of clips for the test set.

Initialization and finetuning. The pretraining-init run starts from the MolmoMotion-AR ($H = 1$) checkpoint; the matched control starts from `Molmo2-4B-Pretrain` and is trained on DROID alone. Both runs use the same hyperparameters: AdamW, 128 global batch on 8 H100s, max-norm 1.0 gradient clipping, bf16 mixed precision. Both runs are trained for 12K finetuning steps; we evaluate every 1K steps and trace the test-loss curves shown in Fig. 6b.

Evaluation. We report 3D test L2 on held-out DROID clips, computed identically to the ADE metric of Sec. 5.1 but on the DROID test split. The pretraining-init run starts at substantially lower L2 and reaches the matched-control's 12K-step error after only $\approx$ 2K finetuning steps.

G Video generation experiment details

We expand the video-generation experiment summarized in Sec. 5.3: starting from the first frame and action description of a benchmark clip, we ask three image-to-video generators to synthesize

the rest of the clip and compare their outputs on standard video-quality metrics. We describe the methods compared (§G.1) and the evaluation protocol (§G.2).

G.1 Methods compared

We evaluate three image-to-video generators. DAS [27] is a 3D-track-conditioned image-to-video model built on the CogVideoX-5B backbone [83], and is the only method that consumes MolmoMotion's predicted 3D trajectories. CogVideoX-5B-I2V is the same backbone without the tracking branch; comparing DAS+MolmoMotion against it isolates the contribution of track conditioning while controlling for the underlying generator. Wan2.2-I2V-A14B [11] is an unconditioned image-to-video baseline at roughly $2.8\times$ the parameter count of CogVideoX-5B-I2V, and tells us how a much larger generator without explicit motion conditioning compares to a smaller one with it.

G.2 Evaluation protocol

For each clip we feed the same first-frame image and caption to all three generators; DAS additionally receives the MolmoMotion prediction track. Each generated clip is then transformed to the ground-truth frame count, frame rate, and resolution, so all methods are scored on the same temporal and spatial grid.

We score each generated clip on five higher-is-better video-quality metrics (Tab. 8): four standard VBench dimensions [84]—subject consistency, motion smoothness, dynamic degree, and background consistency—and a CLIP-based temporal-consistency score, reported alongside as a complementary check on frame-to-frame coherence. We aggregate over PointMotionBench (§C).

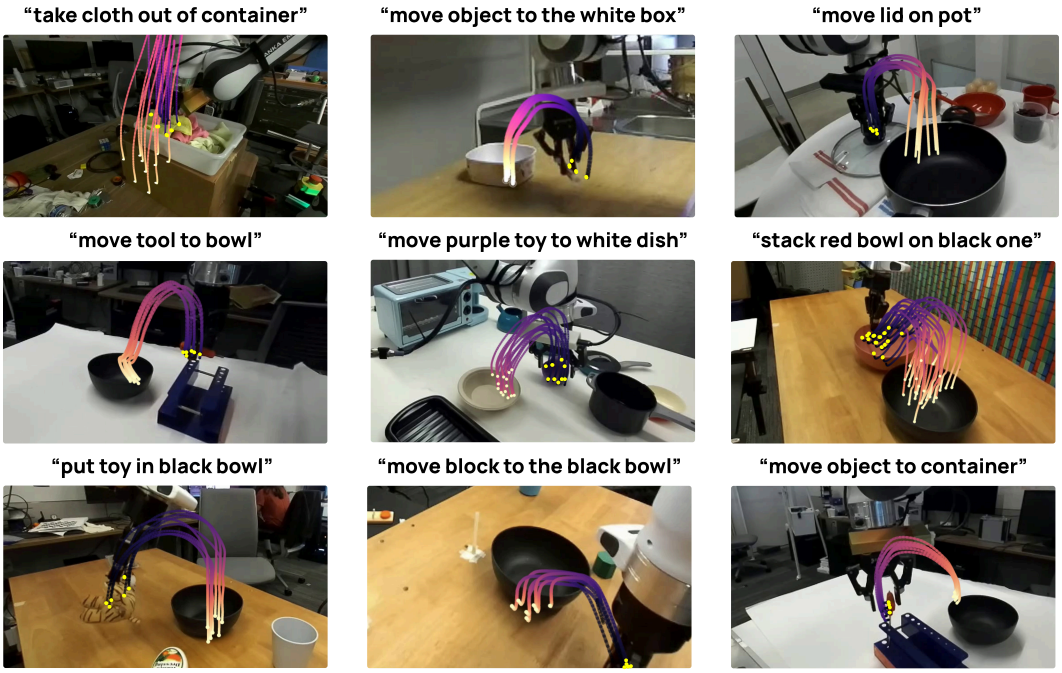


Figure 9: **MolmoMotion predictions on held-out DROID clips.**



Figure 10: **Video generation comparisons on held-out PointMotionBench prompts (1/2).**

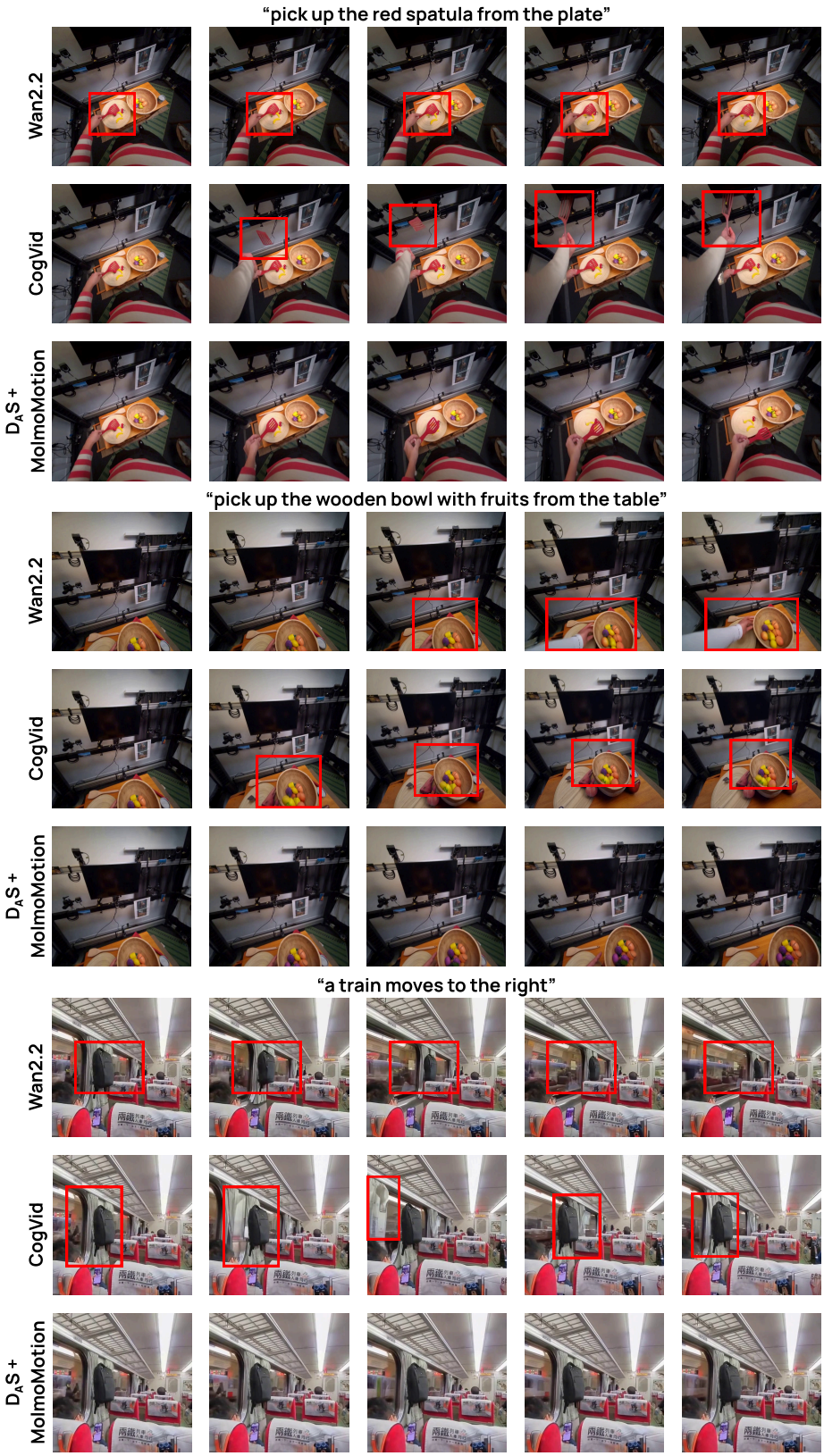


Figure 11: **Video generation comparisons on held-out PointMotionBench prompts (2/2).**

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state three contributions: the MolmoMotion-1M data pipeline and PointMotionBench benchmark, the MolmoMotion motion-prediction model, and downstream transfer of MolmoMotion to robot manipulation and trajectory-conditioned video generation. All three are described in detail in the corresponding sections and empirically evaluated.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations of our work are discussed in the "Limitations" paragraph of Sec. 6 (Conclusions and Limitations).

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper contains no theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The MolmoMotion architecture and training are described in the method and experiments sections; the MolmoMotion-1M data pipeline and PointMotionBench construction protocol are documented in detail in the appendix, including model versions, prompts, thresholds, and per-corpus settings sufficient for reproduction.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The MolmoMotion training and evaluation code is released at https://github.com/allenai/molmo-motion, the model checkpoints at https://huggingface.co/collections/allenai/molmomotion, and MolmoMotion-1M and PointMotionBench at https://huggingface.co/datasets/allenai/molmo-motion-1m and https://huggingface.co/datasets/allenai/PointMotionBench, with instructions to reproduce the main results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The MolmoMotion training recipe (stages, optimizer, batch size, and hardware) is given in the experiments section, and the MolmoMotion-1M data-pipeline hyperparameters together with the relevant training parameters are documented in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [N/A]

Justification: Multi-seed error bars are not standard for the 3D motion-forecasting or video-generation benchmarks used to evaluate MolmoMotion given the training compute required; we report single-run point estimates on the fixed PointMotionBench test split.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: MolmoMotion training compute is reported in the experiments section, and the MolmoMotion-1M data-pipeline compute is in the appendix. Moreover, the benchmark and experiments compute can be found in appendix as well.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: Our work focuses on foundational 3D motion forecasting and is built entirely from publicly available datasets and pretrained models, so we do not foresee specific incremental societal impacts beyond those already inherent to those components.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: MolmoMotion-1M and PointMotionBench are 3D-trajectory annotation sets derived from existing public corpora; they pose no incremental misuse risk beyond the source datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All source datasets and pretrained models used to construct MolmoMotion-1M and PointMotionBench are cited and used under their respective licenses, with versions and provenance documented in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: MolmoMotion-1M and PointMotionBench are documented in detail in the experiment section and the appendix, and are publicly released on HuggingFace with documentation, per-corpus statistics, and licenses.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: No crowdsourcing or human-subject experiments were conducted.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The work does not involve human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: This research does not use LLMs as part of its core methods. Any use of LLMs was limited to writing or editing support and does not impact the scientific content of the paper.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.