🟧TraceGen: World Modeling in 3D Trace-Space Enables Learning from Cross-Embodiment Videos

Seungjae Lee[1*]   Yoonkyo Jung[1*]   Inkook Chun[2*]   Yao-Chih Lee[1]   Zikui Cai[1]

Hongjia Huang[2]   Aayush Talreja[1]   Tan Dat Dao[1]   Yongyuan Liang[1]

Jia-Bin Huang[1]   Furong Huang[1]

[1]University of Maryland, College Park   [2]New York University
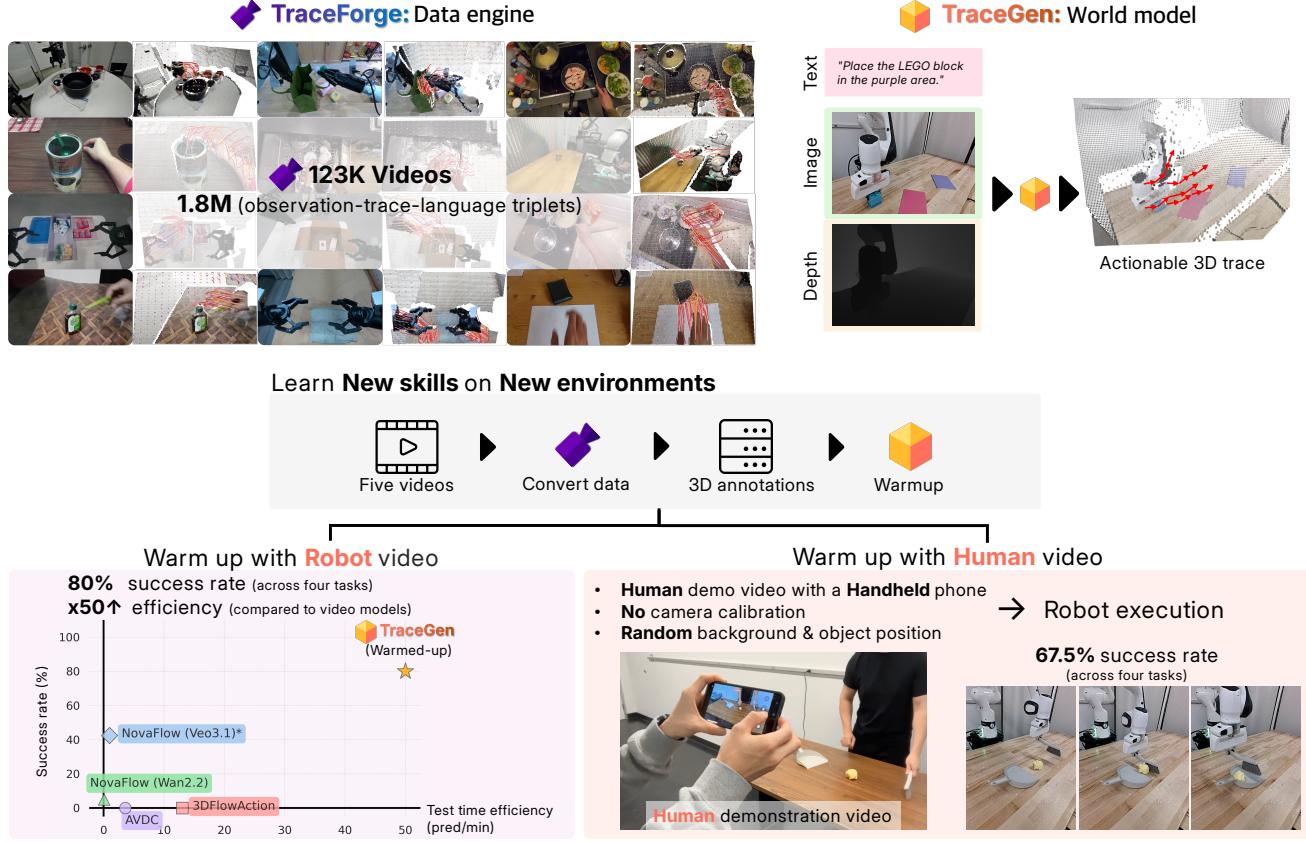
Project Page: https://tracegen.github.io

Figure 1. 🟪**TraceForge** provides the structured training signal, and 🟧**TraceGen** consumes this signal to learn a world model in 3D trace space. Pretrained on 1.8M observation–trace–language triplets from the TraceForge-123K corpus—combining in-the-wild human videos and heterogeneous robot datasets—TraceGen acquires a strong 3D motion prior, enabling rapid adaptation to new skills and new environments. *Bottom-left*: **Robot-domain warm-up**. With only five target-robot demonstrations, TraceGen reaches **80% success** across four tasks and is **50× faster** than video-based world models (Veo 3.1 inference via API averages). *Bottom-right*: **Human→Robot transfer**. With just five uncalibrated handheld human videos—featuring different backgrounds and object positions—TraceGen attains **67.5%** real-robot success.

*Equal contribution

1

Abstract

Learning new robot tasks on new platforms and in new scenes from only a handful of demonstrations remains challenging. While videos of other embodiments—humans and different robots—are abundant, differences in embodiment, camera, and environment hinder their direct use. We address the small-data problem by introducing a unifying, symbolic representation—a compact 3D "trace-space" of scene-level trajectories—that enables learning from cross-embodiment, cross-environment, and cross-task videos. We present 🧊TraceGen, a world model that predicts future motion in trace-space rather than pixel space, abstracting away appearance while retaining the geometric structure needed for manipulation. To train TraceGen at scale, we develop 🔷TraceForge, a data pipeline that transforms heterogeneous human and robot videos into consistent 3D traces, yielding a corpus of 123K videos and 1.8M observation–trace–language triplets. Pretraining on this corpus produces a transferable 3D motion prior that adapts efficiently: with just five target robot videos, TraceGen attains 80% success across four tasks while offering 50–$600\times$ faster inference than state-of-the-art video-based world models. In the more challenging case where only five uncalibrated human demonstration videos captured on a handheld phone are available, it still reaches 67.5% success on a real robot, highlighting TraceGen's ability to adapt across embodiments without relying on object detectors or heavy pixel-space generation.

1. Introduction

Robots are expected to master diverse manipulation tasks across platforms and scenes, yet collecting sufficient, task-specific robot demonstrations is slow and costly. In contrast, large corpora of human videos are readily available, but embodiment, camera, and scene disparities make direct reuse difficult. We ask: *Can we exploit cross-embodiment videos to overcome small-data regimes for new robots and tasks?*

Limitations of pixel and language spaces. Recent progress in large vision-language-action models and multitask policies is notable, but performance often degrades outside training domains [3, 6, 23]. A natural alternative is to leverage pretrained world models built on video generation or vision-language models (VLMs) [27, 28, 36, 53–55]. However, video generators operate in *pixel space*, allocating capacity to backgrounds and textures that are irrelevant to control, while VLMs produce token sequences that lack the spatial precision required for fine-grained object motion. In both families, inference is computationally expensive, complicating real-time planning and fine-tuning.
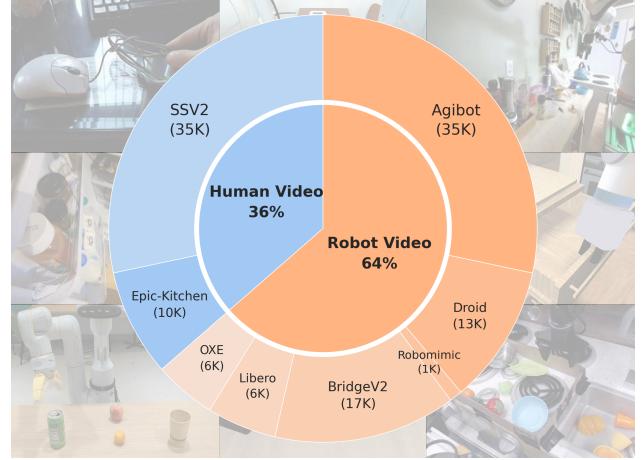

Figure 2. **TraceForge-123K dataset distribution.** Our corpus contains 1.8M observation–trace–language triplets, spanning tabletop, egocentric, and in-the-wild footage with moving cameras to support generalization across embodiments and scenes.

Key insight: a shared 3D structure. Although embodiments differ in kinematics and scale, the motion of manipulated objects and end-effectors admits a shared, scene-centric 3D structure. We term this compact, symbolic representation the *trace-space*: a sequence of 3D trajectories that captures the *where and how* of motion while discarding appearance and backgrounds. Learning in trace-space promises invariance to camera and environment, and a practical path to reusing cross-embodiment, in-the-wild video.

Approach: TraceGen in trace-space. We propose 🧊TraceGen, a world model that predicts future motion directly in 3D trace-space rather than pixels. By modeling scene-level trajectories, TraceGen focuses on the geometric signal pertinent to manipulation and avoids heavy generative rendering. Pretraining on in-the-wild human videos and heterogeneous robot datasets gives TraceGen a transferable motion prior that adapts to new robots and scenes with only a few warm-up videos—enabling fast human→robot and robot→robot transfer without object detectors or heuristic filtering.

Data engine: TraceForge. To enable scalable training, we introduce 🔷TraceForge, which consolidates heterogeneous sources—from controlled in-lab robot videos to in-the-wild human videos—into a unified 3D trace representation. TraceForge compensates camera motion, reconstructs frame-level trajectories from multiple viewpoints, and applies speed retargeting to normalize embodiment-dependent motion. The resulting dataset comprises 123K videos and 1.8M observation–trace–language triplets, providing diverse supervision for a robust 3D motion prior.

Results in low-data regimes. We evaluate two low-data adaptation settings that differ in the source and embodiment: *(i) Robot→Robot adaptation (small in-domain warm-up)*—with a five warm-up set of target-robot videos, TraceGen attains 80% success across four tasks; and *(ii) Human→Robot transfer (no target-robot data)*—fine-tuning TraceGen only on five uncalibrated human demonstration videos recorded with a handheld phone in a different scene yields 67.5% real-robot success. In both settings, trace-space inference is 50–600× faster than state-of-the-art video-generation-based world models.

Contributions. Our contributions are:

- 🟠**TraceGen**: a world model that operates in 3D trace-space, enabling learning *from* cross-embodiment, cross-environment, and cross-task videos by abstracting appearance and camera variation.
- 🔷**TraceForge**: a unified pipeline that converts cross-embodiment videos into consistent 3D traces via camera-motion compensation, and speed retargeting.
- **Scalable 3D trace learning and unified policy:** training on 1.8M observation–trace–language triplets across 123K videos (>15× prior work) to learn a *single, embodiment-agnostic policy in trace space* that predicts scene-level 3D motion without detectors or heuristic filtering.
- **Efficient few-shot adaptation**: 80% success across four tasks with five in-domain robot videos and 67.5% success from five human demos (human → robot transfer from handheld, uncalibrated camera), while achieving > 50× faster inference than video-based world models.

2. Related Work

2.1. Embodied World Models

World-model formulations for robotic manipulation span three major output-space families:

First, video generation models predict raw pixels in future frames [27, 34, 58]. While expressive, they spend capacity reconstructing backgrounds and textures irrelevant to control, increasing computational cost, and risking geometry/affordance hallucinations (Fig. 3(a)).

Second, language token-space models, such as VLM-based planners, generate discrete tokens; however, token-level outputs lack the spatial and temporal resolution required to represent fine object motion, limiting downstream control [1, 26, 28, 48] (Fig. 3(b)). Some works attempt to represent motions as skill tokens [8, 22], but such representations inherit the limitations of their predefined extractors.

Third, trace prediction models directly output future motion signals. Although more efficient and better aligned with control, previous work primarily trains on static, in-lab demonstrations and is largely restricted to 2D traces [4, 14, 38, 45, 47]. Few 3D variants [62] still focus solely

(a) NovaFlow (Wan2.2-I2V)   (b) Gemini Robotics-ER 1.5
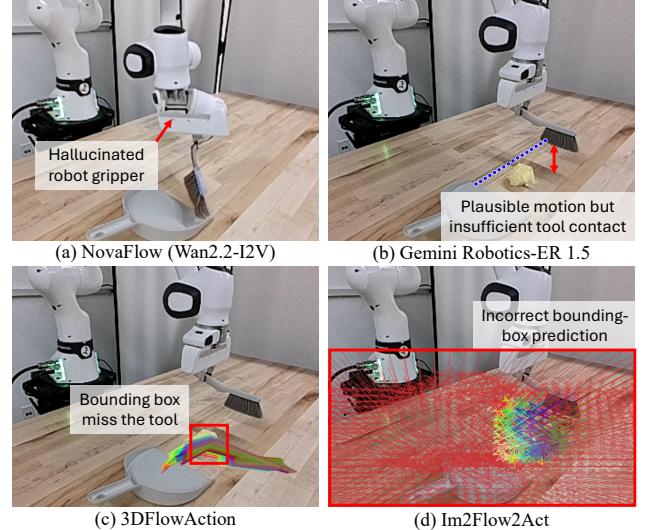
(c) 3DFlowAction   (d) Im2Flow2Act

Figure 3. **Failure cases of existing embodied world models.** (a) Video-based models can hallucinate geometry or affordance. (b) VLM token outputs fail to capture fine motion. Bounding boxes miss the tool (c) or become overly broad (d).

on manipulated objects, requiring auxiliary object detection and heuristic filtering. These modules introduce error cascades and cannot capture robot motion, yielding an incomplete physical representation (Fig. 3(c)).

In contrast, 🔷 **TraceForge** provides a lightweight pipeline that extends beyond in-lab data to in-the-wild videos, enabling the construction of large-scale training sets. Building on this, 🟠**TraceGen** is trained on 15× larger image–trace–language triple data than prior work [62] and predicts scene-level 3D trajectories—robot and objects together—without heuristic filtering or bounding boxes. This yields a unified motion representation suitable for cross-embodiment learning.

Implicit world models for representation learning. Or-thogonal to the output-space families above, a body of work learns *implicit* world models that shape latent dynamics for control without explicitly decoding future pixels or object/scene traces. [9, 41, 59, 60, 63]. These methods have shown strong representation transfer, but typically operate in 2D feature space and do not provide metrically consistent, scene-level 3D motion; precise object/end-effector trajectories then require additional modules. TraceGen is complementary: it *explicitly* models future motion in a compact 3D trace space, yielding a physically grounded, retargetable representation; in principle, implicit objectives can be layered onto TraceGen's encoder to further strengthen pretraining.
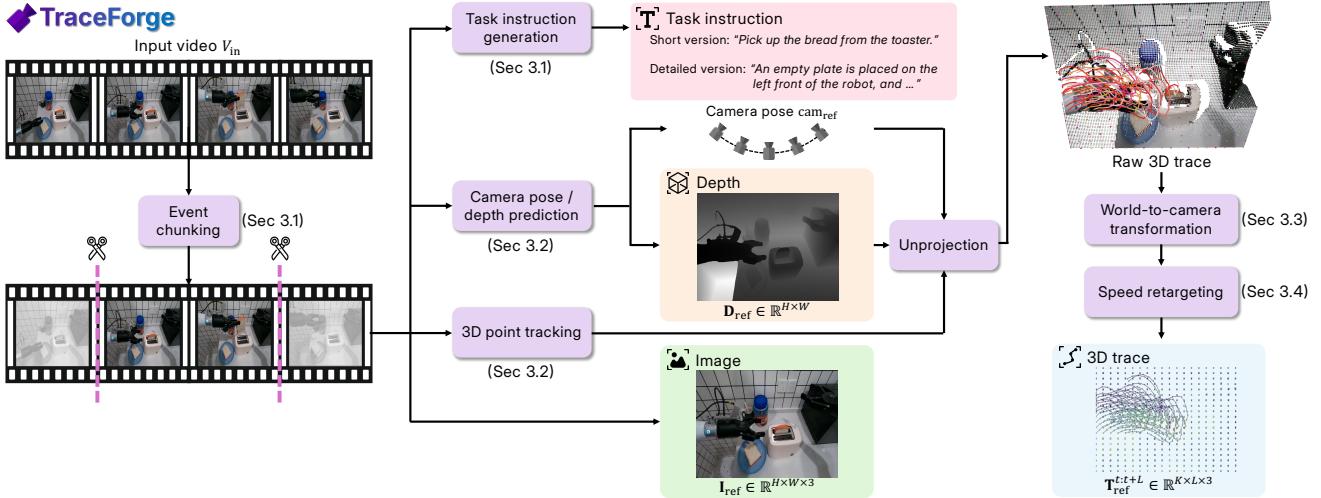
Figure 4. **Building the ◆TraceForge dataset.** From an input video V_{in}: (i) chunk task-relevant spans for curation and generate task instructions (Sec. 3.1); (ii) estimate camera pose and depth, select a reference image and track 3D points to form a raw trace (Sec. 3.2); (iii) apply world–to-camera alignment (Sec. 3.3); (iv) speed retargeting to produce the final 3D trace (Sec. 3.4).

2.2. Trace for Robot Manipulation

While scaling visual imitation has achieved promising manipulation skills, large models often need numerous expert robot trajectories and still struggle to generalize to new objects and scenes [5, 13, 18, 30, 32, 50, 64]. To improve transfer while reducing reliance on robot-only data, previous work leverages structured motion representations — trace. The predicted trace can be used by a variety of downstream modules, such as planning or tracking-based execution [12, 15, 27, 39, 42, 51], supervision or observation for policy learning [17, 52, 61], or high-level planning. In this work, we adopt a basic tracking controller as a minimal demonstration of executing our scene-level 3D traces; developing more sophisticated policies is left for future work.

3. ◆ TraceForge: Dataset Construction

Overview of the TraceForge-TraceGen Pipeline. TraceForge and TraceGen together form a unified world-modeling framework. TraceForge serves as a scalable data engine (Sec. 3), converting heterogeneous human and robot videos into consistent 3D trace annotations paired with multimodal observations and language. TraceGen (Sec. 4) is trained on these large-scale trace-annotated triplets to learn a scene-level motion prior that predicts future trajectories directly in 3D trace-space. The next sections detail each component.

We introduce ◆ **TraceForge**, a unified pipeline that turns heterogeneous human and robot videos into large-scale, trace-annotated world-modeling data. Unlike prior work limited to static cameras or object-centric filtering, TraceForge operates directly on in-the-wild footage with moving viewpoints: it estimates camera pose, compensates

camera motion, and reprojects traces into a fixed reference camera cam_{ref}. Each episode is paired with automatically generated task instructions, yielding multimodal triplets of {observation, trace, language}. Using TraceForge, we curate 123K episodes ($\sim$1.8M observation–trace–language triplets) from eight sources spanning human demonstrations, single-arm robot manipulation, and bimanual robot manipulation [7, 10, 11, 16, 21, 31, 33, 35, 43].

3.1. Event Chunking and Instruction Generation

To align traces and language with underlying actions, we isolate task-related segments from each video and use them to construct observation–trace–language triplets. When start–end event indices are available, we extract the corresponding segment, splitting episodes with multiple labels into separate chunks. Otherwise, we identify task-relevant frames by removing those with negligible motion, as determined from point-tracking results.

For each event chunk, we generate diverse task instructions to better reflect how humans naturally specify goals to a robot and to reduce sensitivity to any single phrasing. Using a VLM, we produce three complementary instructions: (i) a short imperative, (ii) a multi-step decomposition, and (iii) a natural, human-like request. When the dataset already provides a human-written instruction, we keep it and augment it with these three variants. Otherwise, we sample representative frames from the start, middle, and end of the chunk and prompt the VLM to propose task instructions.

3.2. 3D Point Tracking with Camera Pose and Depth Prediction

We extract 3D traces from videos with camera motion by recovering per-frame traces from each camera viewpoint.
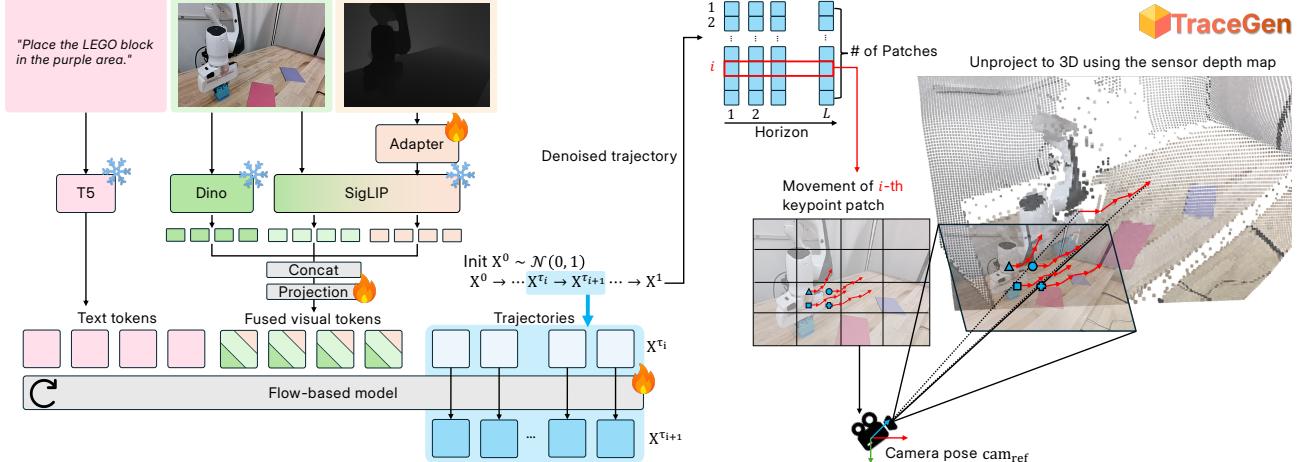
4

Figure 5. **Overview of 🔶TraceGen.** Given language, RGB, and depth inputs, text is encoded by a frozen T5 encoder, RGB images are processed by DINOv3 and SigLIP, and depth maps are passed through a SigLIP encoder with a learnable stem adapter. The resulting visual features (RGB + depth) are concatenated and linearly projected to form unified visual tokens. Together with text tokens, these serve as conditioning inputs to a CogVideoX-based flow model, which predicts a velocity field that transforms Gaussian noise into trace patches via ODE integration. $\mathbf{X}^1$ represents the velocity-like 3D keypoint increments across frames predicted by the flow decoder, where $0, \cdots \tau_i, \tau_{i+1}, \cdots, 1$ denote the continuous interpolation times from pure noise to the clean trace increments. The predicted patches are then unpatched into 3D keypoint trajectories, expressed in the camera coordinate frame. These trajectories can be executed using various low-level controllers; in our experiments, we apply inverse kinematics to map predicted 3D traces to robot joint commands.

At the beginning of each event chunk, we select a reference frame, place a uniform 20×20 grid of keypoints K on its image, and track these points for a trace length of L steps. Instead of representing traces in full camera coordinates, we model each 3D trace point as (x, y, z), where (x, y) denotes the image-plane coordinates and z is the corresponding depth. This allows 3D traces to share the same screen alignment as 2D traces, enabling co-training and consistent supervision across both 2D and 3D modalities.

For 3D estimations of a video, including camera pose, depth, and 3D point traces, we adopt TAPIP3D [57] as the 3D tracking model with CoTracker3 [19, 25] as the point tracker. To improve efficiency, we replace its MegaSAM [29] component with a fine-tuned VGGT [44] depth and camera pose predictor from SpatialTrackerV2 [46], which achieves comparable accuracy while providing significantly faster inference without 3D optimization. Given an event chunk, our model generates per-frame camera poses and depth maps, and then reconstructs 3D point traces for the tracked keypoints. We designate the reference camera frame as $\mathrm{cam_{ref}}$ and its depth map as $\mathbf{D}_{\mathrm{ref}}$, and express all 3D traces in the coordinate system of $\mathrm{cam_{ref}}$, providing a consistent reference frame that effectively compensates for camera motion during data curation. We additionally run CoTracker3 as a pure 2D point tracker on videos that require only image-plane motion, yielding extra 2D-only traces that increase the overall dataset size. Approximately 20% of all traces in our corpus are 2D-only.

3.3. World-to-camera Transformation

We transform all 3D traces to the reference camera frame $\mathrm{cam_{ref}}$ to maintain point-of-view-consistency across time. Given K 3D traces in the world coordinates, we first use the estimated camera extrinsics at $\mathrm{cam_{ref}}$ to transform them to camera coordinates, yielding $[X^c, Y^c, Z^c]^\top$. Subsequently, we obtain the pixel coordinates of the traces, (x, y), transformed by the estimated camera intrinsics. Finally, we compose the pixel coordinates and depth values as screen-aligned 3D traces $\mathbf{T}_{\mathrm{ref}}^{t:t+L} = [x_i, y_i, z_i]_{i=t}^{t+L}$, where L denotes the number of timesteps and $z = Z^c$.

3.4. Speed Retargeting

Human and robot demonstrations of the same task often differ in duration and execution speed. If we use these traces, the model sees the same behavior with different lengths and time scales, making it harder to learn a consistent motion representation. To make traces comparable across episodes and embodiments, we apply speed retargeting.

Each trace is temporally normalized to a fixed length L while preserving its relative motion profile. Specifically, we compute the cumulative arc length along the 3D path, reparameterize by normalized arc-length parameter, and resample at L uniformly spaced targets. This yields consistently sampled, training-ready traces that align in length across embodiments without distorting local velocity patterns.

4. 🧊 TraceGen: Architecture and Training

We present 🧊 **TraceGen**, a flow-based world model that predicts future 3D motion trajectories from multimodal observations. Our model builds on the CogVideoX [49] architecture and employs a Prismatic-VLM [20] multi-encoder fusion strategy to integrate heterogeneous visual and linguistic information.

4.1. Multi-Encoder Feature Extraction

RGB encoders. We adopt a multi-stream encoding strategy that captures complementary visual representations. For each RGB input image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$, we extract features using two frozen pretrained encoders:

- **DINOv3** [40]: A self-supervised vision transformer (ViT-L/16) that produces spatially-aware geometric features $\mathbf{F}_{\text{dino}} \in \mathbb{R}^{N \times D_d}$.
- **SigLIP** [56]: A vision-language model (SigLIP-Base-Patch16-384) that generates semantically aligned features $\mathbf{F}_{\text{siglip}} \in \mathbb{R}^{N \times D_s}$ suitable for text-conditioned prediction.

Depth encoder. To incorporate 3D geometric information, we process depth maps $\mathbf{D} \in \mathbb{R}^{H \times W}$ through a third encoder equipped with a learnable *stem adapter*—a 1×1 convolutional layer that projects single-channel depth to the 3-channel input space expected by SigLIP, yielding $\mathbf{F}_{\text{depth}} \in \mathbb{R}^{N \times D_s}$

Text encoder. Task instructions are encoded using a frozen T5-base [37] encoder, producing contextualized text embeddings $\mathbf{F}_{\text{text}} \in \mathbb{R}^{M \times D}$, where we fix the text sequence length to $M = 128$ tokens and token dimension $D = 768$.

Prismatic VLM fusion. Following Prismatic VLM [20], we concatenate the three vision streams *along the feature dimension*:

$$\mathbf{F}_{\text{vis}} = \text{Concat}(\mathbf{F}_{\text{dino}}, \mathbf{F}_{\text{siglip}}, \mathbf{F}_{\text{depth}}) \in \mathbb{R}^{N \times (D_d + D_s + D_s)}, \quad (1)$$

then project to a unified dimension $D = 768$ via a learnable linear layer:

$$\mathbf{F}_{\text{vis}} = \text{Linear}(\mathbf{F}_{\text{vis}}) \in \mathbb{R}^{N \times D}. \quad (2)$$

The visual tokens $\mathbf{F}_{\text{vis}} \in \mathbb{R}^{N \times D}$ and text tokens $\mathbf{F}_{\text{text}} \in \mathbb{R}^{M \times D}$ are combined to form the conditioning input $\mathbf{F}_{\text{cond}} \in \mathbb{R}^{(N+M) \times D}$ for the flow-based trace decoder.

4.2. Flow-based Trace Decoder

Architecture. Our decoder adapts CogVideoX's [49] 3D transformer to operate in trace space. The input is a $K \times L$ grid where $K = 20 \times 20$ spatial keypoints are tracked across $L = 32$ future timesteps, with each point as $(x, y, z) \in$ $\mathbb{R}^3$ in the camera frame. We apply spatial patchification with patch size 2×2, where each 2×2 group of keypoints is processed as a single token, resulting in 10×10 spatial tokens per timestep. Following CogVideoX, we inject $\mathbf{F}_{\text{cond}}$ via Adaptive LayerNorms applied separately to contextual input and latent trace tokens, enabling efficient fusion.

Trace generation via stochastic interpolants. Our model aims to generate the 3D trace of the scene, denoted as $\mathbf{T}_{\text{ref}}^{t:t+L}$. Each $\mathbf{T}_{\text{ref}}^t$ corresponds to a 20×20 uniform grid with depth map value at time t. Thus, instead of predicting these absolute grid values directly, we observe that the full 3D trace $\mathbf{T}_{\text{ref}}^{t:t+L}$ can be equivalently reconstructed from the temporal differences

$$\Delta \mathbf{T}_{\text{ref}}^t = \mathbf{T}_{\text{ref}}^{t+1} - \mathbf{T}_{\text{ref}}^t. \quad (3)$$

Therefore, our neural network is trained to predict velocity-like increments in keypoints, implicitly capturing the scene's underlying 3D motion.

We adopt the Stochastic Interpolant framework [2], which unifies diffusion-based and flow-based generative models by defining an interpolation path between data and noise distributions. To streamline notation, we denote the keypoint increments $\Delta \mathbf{T}_{\text{ref}}^t$ as $\mathbf{X} \in \mathbb{R}^{K \times L \times 3}$, which serves as our target data distribution. The framework introduces a stochastic interpolant:

$$\mathbf{I}_\tau = \alpha_\tau \mathbf{X}^1 + \sigma_\tau \boldsymbol{\varepsilon}, \quad \tau \in [0, 1], \quad (4)$$

where $\mathbf{X}^1$ is the ground-truth trace, $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I})$ is Gaussian noise, and α_τ, σ_τ are time-dependent schedules. By varying α_τ and σ_τ, this framework encompasses a range of generative models, including diffusion models and flow-matching methods.

The framework learns a velocity field $\mathbf{v}(\mathbf{x}, \tau, \mathbf{F}_{\text{cond}})$ that characterizes the time evolution of the interpolant, where $\mathbf{x}$ denotes a sample from the interpolant distribution at time τ:

$$\mathbf{v}(\mathbf{x}, \tau, \mathbf{F}_{\text{cond}}) = \mathbb{E}[\dot{\mathbf{I}}_\tau \mid \mathbf{I}_\tau = \mathbf{x}, \mathbf{F}_{\text{cond}}], \quad (5)$$

where $\dot{\mathbf{I}}_\tau$ denotes the time derivative and the expectation is over $\mathbf{X}^0, \mathbf{X}^1$ conditioned on $\mathbf{I}_\tau = \mathbf{x}$ and $\mathbf{F}_{\text{cond}}$.

Linear interpolation ODE. Among the variants within the Stochastic Interpolant framework, we implement a *linear interpolation ODE* by choosing $\alpha_\tau = \tau$ and $\sigma_\tau = 1 - \tau$:

$$\mathbf{X}^\tau = (1 - \tau)\mathbf{X}^0 + \tau \mathbf{X}^1, \quad \tau \in [0, 1], \quad (6)$$

With this linear schedule, the velocity field simplifies to $\dot{\mathbf{X}}^\tau = \mathbf{X}^1 - \mathbf{X}^0$, which is constant in time. We train a neural network v_θ to predict this velocity by minimizing:

$$\mathcal{L}_{\text{SI}} = \mathbb{E}_{\tau, \mathbf{X}^0, \mathbf{X}^1} \left[\|v_\theta(\mathbf{X}^\tau, \tau, \mathbf{F}_{\text{cond}}) - (\mathbf{X}^1 - \mathbf{X}^0)\|^2 \right]. \quad (7)$$

(a) Folding the black pants (Clothes)   (b) Inserting the tennis ball into the box (Ball)   (c) Sweeping trash into the dustpan (Brush)   (d) Placing a block on the purple book (Block)
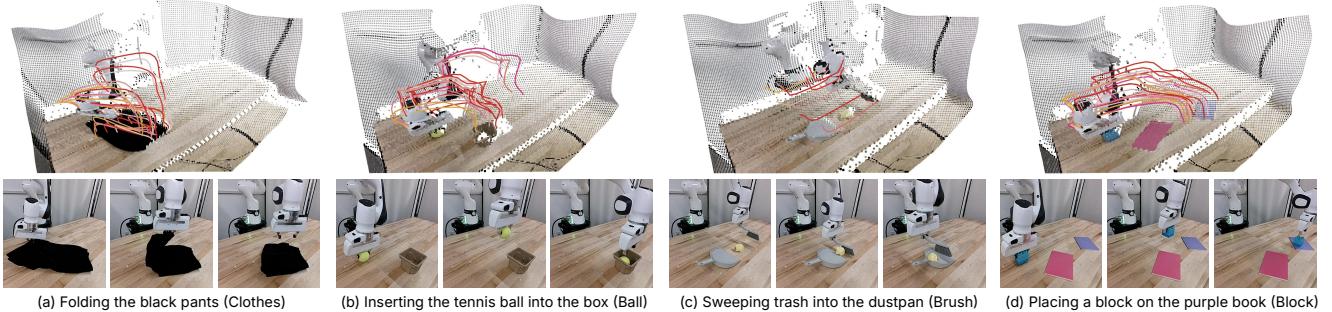
Figure 6. **Real-world experiments with predicted 3D traces.** We evaluate TraceGen and baselines on four real-world manipulation tasks on a Franka Research 3 robot, showing that the predicted 3D traces transfer effectively to real-robot execution.

At test time, we generate trajectories via 100-step ODE integration, relying solely on the conditional model with multimodal vision-language conditioning.

Encoder freezing strategy. To leverage pretrained representations efficiently, we keep all encoders (DINOv3, SigLIP, T5) frozen throughout training and trains only the fusion layer and decoder, following [20].

5. Experiments

Our experiments address three questions: (1) **Effectiveness of TraceGen**: Does planning in compact 3D trace space improve performance and inference efficiency compared to pixel-based alternatives? (Sec. 5.1) (2) **Human–Robot Transfer**: Can TraceGen enable efficient human-to-robot transfer from uncalibrated, in-the-wild videos with differing camera, backgrounds, and object layouts? (Sec. 5.2) (3) **Role of Pretraining and Warmup**: How much do large-scale cross-embodiment pretraining and lightweight warmup contribute to performance and generalization? (Sec. 5.3) (We report a quantitative sanity check of Trace-Forge trace accuracy in the Appendix B.)

Warm-up rationale. 🟧TraceGen learns a unified policy in 3D trace space, predicting future scene-level trajectories that are embodiment-agnostic. To execute on a specific robot, these traces must be "*translated*" into the robot's action space via a lightweight warm-up.
Settings. We evaluate two lightweight regimes that differ only in the source of supervision for warm-up (For both warm-up regimes, we do not cherry-pick demonstrations. We also provide visualizations of *all* warm-up data in the Appendix E.1.):
1. **Robot→Robot (small same-embodiment warm-up).** We fine-tune 🟧TraceGen using a **five** in-domain set of robot demonstrations. The demonstrations differ from the target tasks in the object/target configuration and the robot's initial pose. For example, in *Brush*, demonstra-

tions begin with the brush already in contact with (or very close to) the table and thus omit the critical "lower the brush" motion required at test time. Similarly, for *Block*, demonstration videos use target regions with randomly varying colors and positions.
2. **Human→Robot (no target-robot data).** We fine-tune 🟧TraceGen with **five** uncalibrated human demonstrations (handheld phone, different scene) to adapt trace predictions to the target task; no target-robot demonstrations are used. Each video is only **3–4 seconds** long, and a single person performing the task while another records it is sufficient to obtain the data. Overall, collecting **20 demonstrations** across four tasks required **under 4 minutes**, making the warm-up extremely easy.

5.1. Performance and Efficiency Comparison in Real-World Experiment

Tasks and setup. We evaluate on four manipulation tasks executed on a Franka Research 3 robot: folding a garment (*Clothes*), inserting a tennis ball into a box (*Ball*), sweeping trash into a dustpan with a brush (*Brush*), placing a block in the purple region (*Block*). Given a single RGB-D frame and a language instruction, TraceGen predicts a 3D trace, which is converted to joint commands via inverse kinematics.

Baselines. We include both video-based and trace-based world models. Video-generation approaches such as AVDC [24] and NovaFlow [27] first synthesize future video and then estimate 3D motion post hoc; consistent with NovaFlow's evaluation, we use only the video-generation component of AVDC and apply a unified video-to-trace extraction pipeline across all video-based baselines. For 3DFlowAction [62], which relies on segmentation masks, we supply ground-truth masks due to frequent failures of the original mask estimator.

Performance. Fig. 7 shows that all methods below 10B parameters—**except** 🟧**TraceGen (0.67B)**—fail to produce

Figure 7. **Success rate vs. inference efficiency** (predictions per minute; higher and rightward is better). ⬦**TraceGen** achieves the best combination of success and efficiency, outperforming both video and trace-based baselines by a large margin. Gains stem from its strong 3D motion prior and a *lightweight warm-up* in trace space via ⬦**TraceForge**. In contrast, video-generation baselines (e.g., NovaFlow or video backbone in AVDC) offer no practical few-shot warm-up path in our setting, and several trace baselines rely on object detectors or heuristic object filtering, making warm-up technically difficult. (The Veo 3.1 latency is measured based on its average API call time.)

executable trajectories in the zero-shot setting (0% success across all tasks). Large video-generation models—NovaFlow (Wan2.2) and NovaFlow (Veo3.1)—achieve non-zero zero-shot success, but at the cost of extremely slow inference. Under the same 5-video warm-up procedure, ⬦**TraceGen** attains **80%** success across four tasks despite variations in object layouts and initial robot poses. Large video-generation models exceeding 10B parameters are impractical to warm up due to proprietary APIs or substantial computational requirements. These results suggest that TraceGen's few-shot adaptability stems from its compact trace-space representation and the TraceForge pipeline, which together provide stable motion priors and consistent supervision for lightweight warm-up.

Inference efficiency. Planning in 3D trace space offers substantial computational benefits. ⬦**TraceGen** runs $3.8\times$ faster than trace-generation baselines and over $50\times$ faster than large video-generation models. NovaFlow (Wan2.2) requires more than $600\times$ longer inference time, highlighting the difficulty of scaling pixel-space video prediction for real-time robotics. TraceGen thus provides a practical and efficient solution for closed-loop planning.
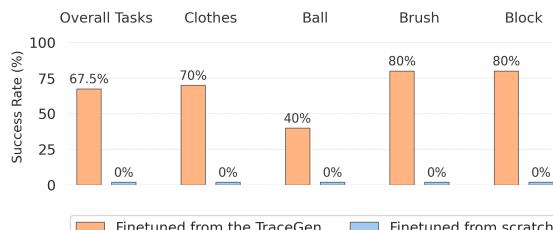
5.2. Human–Robot Skill Transfer

Figure 8. **Human-to-robot skill transfer using human demo videos.** TraceGen, finetuned on 5 in-the-wild handheld phone videos, successfully executes four manipulation tasks, with a success rate of 67.5%. In contrast, the From Scratch model fails (0%), indicating that cross-embodiment pretraining is essential.

Protocol and results. We evaluate whether TraceGen can transfer skills from in-the-wild human demo videos to a real robot. For each task, we collect five handheld phone videos recorded without camera calibration, with varying viewpoints, backgrounds, and object layouts. TraceForge reconstructs 3D traces from these demonstrations, and TraceGen is finetuned on the resulting traces before deployment on a Franka Research 3 robot. As shown in Fig. 8, finetuning on the five human demos yields an overall success rate of **67.5%** across four tasks, whereas the *From Scratch* model fails on all tasks (0%). Despite substantial differences in embodiment, camera intrinsics, and scene appearance, the pretrained TraceGen model adapts effectively with only a small number of uncalibrated human videos, indicating that the 3D trace representation provides a practical bridge between human demonstrations and robot execution.

5.3. Role of Pretraining and Warmup

We investigate how large-scale cross-embodiment pretraining affects TraceGen's ability to adapt with few task demonstrations. We compare the pretrained model with a *From Scratch* variant that shares the same architecture but is trained only on warmup data.

Few-shot warmup and the importance of pretraining. Table 1 summarizes the effect of 5-video and 15-video warmups. With five target-robot videos, the pretrained model achieves an overall success rate of **80%**, whereas the scratch model attains **25%**. Increasing warmup to fifteen videos yields limited additional improvement for the pretrained model (**82.5%**) and marginal change for the scratch variant (**30%**). These results indicate that the majority of TraceGen's performance stems from pretraining, with warmup primarily aligning pretrained motion priors with task-specific configurations.

8

Table 1. Effect of cross-embodiment pretraining under 5-video and 15-video warmup. Pretraining significantly improves success rates compared to training from scratch.

Warm-up	Pretraining	Clothes	Ball	Brush	Block	Overall SR(%)
5 robot videos	Random init.	10/10	0/10	0/10	0/10	25.0%
	TraceGen	10/10	6/10	8/10	8/10	80%
15 robot videos	Random init.	10/10	0/10	0/10	2/10	30.0%
	TraceGen	10/10	9/10	8/10	6/10	82.5%

Effect of pretraining source. To quantify the role of pretraining data, we compare four variants trained with the same 5-video warmup set but different pretraining sources: (1) no pretraining (*From Scratch*); (2) pretraining on SSV2 (human hand–centric, 35K clips); (3) pretraining on Agibot (robot-centric, 35K clips); and (4) full TraceGen pretraining on the cross-embodiment dataset. Under identical warmup, SSV2 pretraining yields **25%** success and Agibot yields **45%**, both lower than cross-embodiment pretraining on the full dataset. These results suggest that both embodiment alignment (robot-centric data) and heterogeneous motion coverage (human + robot sources) matter, and combining them yields substantially better transfer.

Table 2. Effect of pretraining source on 5-video warmup performance. Cross-embodiment pretraining with a larger dataset (TraceGen) yields substantially higher success than single-source pretraining and full scratch training.

Task	From scratch	SSV2 only	Agibot only	TraceForge-123K
Ball	0/10	3/10	4/10	6/10
Block	0/10	2/10	5/10	8/10
Overall SR(%)	0%	25%	45%	70%

6. Conclusion

We presented 🧊**TraceGen**, a cross-embodiment world model that predicts future motion in compact 3D trace space rather than pixel space. By representing manipulation tasks as 3D traces of scene points, TraceGen achieves a unified motion representation that generalizes across diverse embodiments—from human hands to robot arms. To enable large-scale training, we introduced 🔶**TraceForge**, a data-curation pipeline that processes heterogeneous sources into consistent 3D traces by compensating for camera motion and normalizing embodiment-specific speeds. Pretrained on 123K episodes, TraceGen achieves 80–82.5% success on real-world tasks with only 5–15 demonstrations, running 50× faster than video-based approaches. These results suggest that reasoning in trace space provides an effective inductive bias for cross-embodiment learning, offering both computational efficiency and sample efficiency for robot manipulation.

7. Limitations and Future Work

Within the Stochastic Interpolant framework, we adopt linear interpolation with ODE integration. While this approach allows sampling diverse trajectories through different noise initializations, we have not yet explored alternative interpolation schedules or mechanisms to explicitly control which trajectory mode is generated for ambiguous tasks.

The quality of demonstration data varies. A portion of our source videos contains inefficient or corrective motions—where operators make exploratory movements or errors before completing tasks—introducing suboptimal supervision signals. We implemented additional filtering steps to clean the dataset, though some noisy demonstrations remain.

Moreover, TraceGen's zero-shot generation ability, while promising, is not yet fully reliable under novel embodiments or unseen environments, occasionally yielding plausible but physically infeasible trajectories. Additionally, for fine-grained manipulation tasks, the generated trajectories may lack sufficient detail for the robot to execute precise manipulation actions. Scaling to internet-scale demonstration datasets, combined with improved data filtering mechanisms, could address these issues. Finally, extending beyond human-like robot arms to very different robot types would test the limits of trace-space abstraction. Despite these challenges, we believe TraceGen's efficiency and generality represent a meaningful step toward practical cross-embodiment manipulation systems.

Acknowledgements

References

[1] Abbas Abdolmaleki, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Ashwin Balakrishna, Nathan Batchelor, Alex Bewley, Jeff Bingham, Michael Bloesch, et al. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *arXiv preprint arXiv:2510.03342*, 2025. 3

[2] Michael S Albergo, Nicholas M Boffi, and Eric Vanden-

Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023. 6

[3] Jose Barreiros, Andrew Beaulieu, Aditya Bhat, Rick Cory, Eric Cousineau, Hongkai Dai, Ching-Hsin Fang, Kunimatsu Hashimoto, Muhammad Zubair Irshad, Masha Itkina, et al. A careful examination of large behavior models for multitask dexterous manipulation. *arXiv preprint arXiv:2507.05331*, 2025. 2

[4] Homanga Bharadhwaj, Roozbeh Mottaghi, Abhinav Gupta, and Shubham Tulsiani. Track2act: Predicting point tracks from internet videos enables generalizable robot manipulation. In *ECCV*, 2024. 3

[5] Homanga Bharadhwaj, Jay Vakil, Mohit Sharma, Abhinav Gupta, Shubham Tulsiani, and Vikash Kumar. Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking. In *ICRA*, 2024. 4

[6] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025. 2

[7] Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Xindong He, Xu Huang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. In *IROS*, 2025. 4

[8] Jeremy A Collins, Loránd Cheng, Kunal Aneja, Albert Wilcox, Benjamin Joffe, and Animesh Garg. Amplify: Actionless motion priors for robot learning from videos. *arXiv preprint arXiv:2506.14198*, 2025. 3

[9] Zichen Cui, Hengkai Pan, Aadhithya Iyer, Siddhant Haldar, and Lerrel Pinto. Dynamo: In-domain dynamics pretraining for visuo-motor control. In *NeurIPS*, 2024. 3

[10] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. The epic-kitchens dataset: Collection, challenges and baselines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):4125–4141, 2020. 4

[11] Shivin Dass, Alaa Khaddaj, Logan Engstrom, Aleksander Madry, Andrew Ilyas, and Roberto Martín-Martín. Datamil: Selecting data for robot imitation learning with datamodels. *arXiv preprint arXiv:2505.09603*, 2025. 4

[12] Ben Eisner, Harry Zhang, and David Held. Flowbot3d: Learning 3d articulation flow to manipulate articulated objects. In *RSS*, 2022. 4

[13] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *CoRL*, 2017. 4

[14] Chongkai Gao, Haozhuo Zhang, Zhixuan Xu, Cai Zhehao, and Lin Shao. Flip: Flow-centric generative planning as general-purpose manipulation world model. In *ICLR*, 2025. 3

[15] Ankit Goyal, Arsalan Mousavian, Chris Paxton, Yu-Wei Chao, Brian Okorn, Jia Deng, and Dieter Fox. Ifor: Iterative flow minimization for robotic object rearrangement. In *CVPR*, 2022. 4

[16] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *ICCV*, 2017. 4

[17] Jiayuan Gu, Sean Kirmani, Paul Wohlhart, Yao Lu, Montserrat Gonzalez Arenas, Kanishka Rao, Wenhao Yu, Chuyuan Fu, Keerthana Gopalakrishnan, Zhuo Xu, et al. Rt-trajectory: Robotic task generalization via hindsight trajectory sketches. In *ICLR*, 2024. 4

[18] Guangqi Jiang, Yifei Sun, Tao Huang, Huanyu Li, Yongyuan Liang, and Huazhe Xu. Robots pre-train robots: Manipulation-centric robotic representation from large-scale robot datasets. In *ICLR*, 2025. 4

[19] Nikita Karaev, Yuri Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker3: Simpler and better point tracking by pseudo-labelling real videos. In *ICCV*, 2025. 5

[20] Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models. In *ICML*, 2024. 6, 7, 14

[21] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. In *RSS*, 2024. 4

[22] Hanjung Kim, Jaehyun Kang, Hyolim Kang, Meedeum Cho, Seon Joo Kim, and Youngwoon Lee. Uniskill: Imitating human videos via cross-embodiment skill representations. In *CoRL*, 2025. 3

[23] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, et al. Openvla: An open-source vision-language-action model. In *CoRL*, 2025. 2

[24] Po-Chen Ko, Jiayuan Mao, Yilun Du, Shao-Hua Sun, and Joshua B Tenenbaum. Learning to act from actionless videos through dense correspondences. In *ICLR*, 2024. 7

[25] Pulkit Kumar, Shuaiyi Huang, Matthew Walmer, Sai Saketh Rambhatla, and Abhinav Shrivastava. Trokens: Semantic-aware relational trajectory tokens for few-shot action recognition. In *ICCV*, 2025. 5

[26] Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, Winson Han, Wilbert Pumacay, Angelica Wu, Rose Hendrix, Karen Farley, Eli VanderBilt, Ali Farhadi, Dieter Fox, and Ranjay Krishna. Molmoact: Action reasoning models that can reason in space. *arXiv preprint arXiv:2508.07917*, 2025. 3

[27] Hongyu Li, Lingfeng Sun, Yafei Hu, Duy Ta, Jennifer Barry, George Konidaris, and Jiahui Fu. Novaflow: Zero-shot manipulation via actionable flow from generated videos. *arXiv preprint arXiv:2510.08568*, 2025. 2, 3, 4, 7

[28] Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Caelan Reed Garrett, Fabio Ramos, Dieter Fox, Anqi

Li, Abhishek Gupta, and Ankit Goyal. Hamster: Hierarchical action models for open-world robot manipulation. In *ICLR*, 2025. 2, 3

[29] 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 *CVPR*, 2025. 5, 14

[30] Yongyuan Liang, Tingqiang Xu, Kaizhe Hu, Guangqi Jiang, Furong Huang, and Huazhe Xu. Make-an-agent: A generalizable policy network generator with behavior-prompted diffusion. In *NeurIPS*, 2024. 4

[31] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In *NeurIPS*, 2023. 4

[32] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *CoRL*, 2018. 4

[33] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *CoRL*, 2021. 4

[34] Dantong Niu, Yuvan Sharma, Haoru Xue, Giscard Biamby, Junyi Zhang, Ziteng Ji, Trevor Darrell, and Roei Herzig. Pretraining auto-regressive robotic models with 4d representations. In *ICML*, 2025. 3

[35] 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. In *ICRA*, 2024. 4

[36] Delin Qu, Haoming Song, Qizhi Chen, Zhaoqing Chen, Xianqiang Gao, Xinyi Ye, Qi Lv, Modi Shi, Guanghui Ren, Cheng Ruan, et al. Embodiedonevision: Interleaved vision-text-action pretraining for general robot control. *arXiv preprint arXiv:2508.21112*, 2025. 2

[37] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020. 6

[38] Kanchana Ranasinghe, Xiang Li, E-Ro Nguyen, Cristina Mata, Jongwoo Park, and Michael S Ryoo. Pixel motion as universal representation for robot control. *arXiv preprint arXiv:2505.07817*, 2025. 3

[39] Daniel Seita, Yufei Wang, Sarthak J Shetty, Edward Yao Li, Zackory Erickson, and David Held. Toolflownet: Robotic manipulation with tools via predicting tool flow from point clouds. In *CoRL*, 2023. 4

[40] Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025. 6

[41] Yanchao Sun, Shuang Ma, Ratnesh Madaan, Rogerio Bonatti, Furong Huang, and Ashish Kapoor. Smart: Self-supervised multi-task pretraining with control transformers. In *ICLR*, 2023. 3

[42] Mel Vecerik, Carl Doersch, Yi Yang, Todor Davchev, Yusuf Aytar, Guangyao Zhou, Raia Hadsell, Lourdes Agapito, and Jon Scholz. Robotap: Tracking arbitrary points for few-shot visual imitation. In *ICRA*, 2024. 4

[43] Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *CoRL*, 2023. 4

[44] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *CVPR*, 2025. 5, 14

[45] Chuan Wen, Xingyu Lin, John So, Kai Chen, Qi Dou, Yang Gao, and Pieter Abbeel. Any-point trajectory modeling for policy learning. In *RSS*, 2024. 3

[46] 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 *ICCV*, 2025. 5, 14

[47] Mengda Xu, Zhenjia Xu, Yinghao Xu, Cheng Chi, Gordon Wetzstein, Manuela Veloso, and Shuran Song. Flow as the cross-domain manipulation interface. In *CoRL*, 2024. 3

[48] Jianwei Yang, Reuben Tan, Qianhui Wu, Ruijie Zheng, Baolin Peng, Yongyuan Liang, Yu Gu, Mu Cai, Seonghyeon Ye, Joel Jang, et al. Magma: A foundation model for multimodal ai agents. In *CVPR*, 2025. 3

[49] 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. In *ICLR*, 2025. 6

[50] Sarah Young, Dhiraj Gandhi, Shubham Tulsiani, Abhinav Gupta, Pieter Abbeel, and Lerrel Pinto. Visual imitation made easy. In *CoRL*, 2021. 4

[51] Kelin Yu, Sheng Zhang, Harshit Soora, Furong Huang, Heng Huang, Pratap Tokekar, and Ruohan Gao. Genflowrl: Shaping rewards with generative object-centric flow in visual reinforcement learning. In *ICCV*, 2025. 4

[52] Peihong Yu, Amisha Bhaskar, Anukriti Singh, Zahiruddin Mahammad, and Pratap Tokekar. Sketch-to-skill: Bootstrapping robot learning with human drawn trajectory sketches. In *RSS*, 2025. 4

[53] Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. Robopoint: A vision-language model for spatial affordance prediction for robotics. In *CoRL*, 2024. 2

[54] Yifu Yuan, Haiqin Cui, Yibin Chen, Zibin Dong, Fei Ni, Longxin Kou, Jinyi Liu, Pengyi Li, Yan Zheng, and Jianye Hao. From seeing to doing: Bridging reasoning and decision for robotic manipulation. *arXiv preprint arXiv:2505.08548*, 2025.

[55] Yifu Yuan, Haiqin Cui, Yaoting Huang, Yibin Chen, Fei Ni, Zibin Dong, Pengyi Li, Yan Zheng, and Jianye Hao.

Embodied-r1: Reinforced embodied reasoning for general robotic manipulation. *arXiv preprint arXiv:2508.13998*, 2025. 2

[56] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *ICCV*, 2023. 6

[57] Bowei Zhang, Lei Ke, Adam W. Harley, and Katerina Fragkiadaki. Tapip3d: Tracking any point in persistent 3d geometry. In *NeurIPS*, 2025. 5

[58] Haoyu Zhen, Qiao Sun, Hongxin Zhang, Junyan Li, Siyuan Zhou, Yilun Du, and Chuang Gan. Learning 4d embodied world models. In *ICCV*, 2025. 3

[59] Ruijie Zheng, Xiyao Wang, Yanchao Sun, Shuang Ma, Jieyu Zhao, Huazhe Xu, Hal Daumé III, and Furong Huang. Taco: Temporal latent action-driven contrastive loss for visual reinforcement learning. In *NeurIPS*, 2023. 3

[60] Ruijie Zheng, Yongyuan Liang, Xiyao Wang, Shuang Ma, Hal Daumé III, Huazhe Xu, John Langford, Praveen Palanisamy, Kalyan Shankar Basu, and Furong Huang. Premier-taco is a few-shot policy learner: Pretraining multitask representation via temporal action-driven contrastive loss. In *ICML*, 2024. 3

[61] Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. In *ICLR*, 2025. 4

[62] Hongyan Zhi, Peihao Chen, Siyuan Zhou, Yubo Dong, Quanxi Wu, Lei Han, and Mingkui Tan. 3dflowaction: Learning cross-embodiment manipulation from 3d flow world model. *arXiv preprint arXiv:2506.06199*, 2025. 3, 7

[63] Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. In *ICML*, 2025. 3

[64] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *CoRL*, 2023. 4

Appendix

A. Prompts for Task Instruction Generation

To obtain consistent and diverse task instructions from video segments in ◆TraceForge, we use a vision-language model (VLM) to transform representative video frames into three complementary instruction styles. As described in the main manuscript, each event chunk is paired with (i) a concise imperative command, (ii) a stepwise manipulation instruction, and (iii) a natural, human-like request. When human-written instructions are already available, we preserve them and augment them with these additional variants. Otherwise, we sample frames from the beginning, middle, and end of the chunk and prompt the VLM to generate instructions following the structured specification below.

Sample Prompt for Instruction Generation

You are an expert image analyzer. You will receive a sequence of frames from a single video episode that records a simple manipulation task. The frames are ordered chronologically from initial state to final state.

GOAL Infer the most likely task being performed in this episode and return the OUTPUT FORMAT below.

TASK INSTRUCTIONS
- IMPORTANT: Do **NOT** generate a descriptive sentence like "The agent is trying to grab the sink faucet."
- Instead, generate instructions as if a human is directly commanding a robot or agent to make this situation happen.
- Provide exactly three instructions, one for each category:
  - `instruction_1`: Direct, simple imperative command e.g., "Turn on the faucet."
  - `instruction_2`: Step-by-step explicit manipulation e.g., "Move your gripper toward the faucet handle."
  - `instruction_3`: Natural human-like request e.g., "Can you turn on the sink for me?"
- All instructions must explicitly state **what** object is manipulated and **how**.

Instruction length rules
- `instruction_1`: $\leq$ 10 words
- `instruction_2`: $\leq$ 20 words
- `instruction_3`: $\leq$ 15 words

The operator can be a human, robot, or tool.

OUTPUT (JSON ONLY)

```
{
```

```
"instruction_1":
    "<=10 words
    imperative command>",
"instruction_2":
    "<=20 words
    stepwise/explicit command>",
"instruction_3":
    "<=15 words
    natural human-like request>"
}
```

POLICIES
- Do NOT invent objects that are not visible.
- Prefer specific names if clear (banana), else generic ones (container).
- All strings must be in English.
- Return only valid JSON—no markdown or extra text.

B. 3D Trace Extraction Accuracy of TraceForge

As discussed in the main text, ◆TraceForge provides the large-scale 3D motion supervision used to train TraceGen. To ensure that this supervision is reliable, we report a quantitative sanity check of TraceForge's 3D trace extraction accuracy.

We evaluate whether TraceForge recovers 3D trajectories that faithfully reflect real robot motion by comparing its predicted traces with ground-truth end-effector trajectories obtained via forward kinematics across nine teleoperated episodes. Since TraceForge represents motion as a 20×20 grid of point trajectories, we identify, for each episode, the predicted point whose 2D projection is closest to the end-effector in the first frame and treat its 3D path as the corresponding predicted trace.

Across episodes (13–24.5 seconds each, with an average displacement of 70.96 cm), TraceForge achieves sub–2.3 cm endpoint error on all axes (Table 3). These results indicate that the TraceForge extraction pipeline produces centimeter-level motion accuracy, providing reliable supervision for training TraceGen.

Table 3. Absolute endpoint error along the x, y, z axes in camera coordinate between predicted and ground-truth trajectories.

Error (cm)	x	y	z
Mean	1.66	1.79	2.26
Std	0.82	1.82	2.69

C. Model Training Details

This section provides comprehensive details on Trace-Gen's training configuration, complementing the architecture overview in the main manuscript. TraceGen employs a multi-encoder architecture with DINOv2 and SigLIP for RGB feature extraction, a depth encoder with a learnable stem adapter for geometric information, and T5 for text encoding.

C.1. Encoder Freezing Strategy

To leverage pretrained representations efficiently, we keep all encoders (DINOv2, SigLIP, T5) frozen throughout training and train only the fusion layer and decoder. This design choice follows Prismatic VLM [20], which demonstrated that finetuning visual backbones significantly degrades performance on vision-language tasks. Their analysis revealed that updating pretrained vision encoders during task-specific finetuning leads to catastrophic forgetting of the rich visual priors learned during large-scale pretraining.

By maintaining frozen encoders, TraceGen retains:
- **Visual features from DINOv2**, trained via self-supervised learning on large-scale natural images, providing strong geometric and semantic understanding
- **Vision-language alignment from SigLIP**, enabling effective conditioning on text instructions
- **Linguistic representations from T5**, capturing task semantics and manipulation goals

The trainable fusion layer and flow-based decoder learn to combine and map these frozen representations to the 3D trace prediction task. This approach substantially reduces the number of trainable parameters, accelerates training, and improves generalization to unseen manipulation scenarios.

D. Evaluations

D.1. Evaluation setup

We evaluate whether 3D traces predicted by TraceGen enable effective robot manipulation. Experiments are conducted on a Franka Research 3 robot across four tasks:
- folding a garment (Clothes)
- inserting a tennis ball into a box (Ball)
- sweeping trash into a dustpan with a brush (Brush)
- placing a LEGO block in the purple region (Block)

Given a single RGB-D observation and a text instruction, TraceGen predicts trajectories using 100-step ODE integration with classifier-free guidance (guidance scale 2) for *Brush* and *Clothes*, and without guidance for *Block* and *Ball*. The predicted trajectories are converted into joint-space commands through inverse kinematics after transforming them from the camera frame to the robot base frame $T_{\text{base} \leftarrow \text{cam}_{\text{ref}}}$. For all methods, the predicted z values are rescaled to match the measured depth maps.

D.2. Depth rescaling

To align the predicted trace depths with the original sensor depth, we apply a depth-rescaling procedure. Similar to NovaFlow, which computes a single scaling factor based on the median depth of the initial ground-truth map, we also begin by estimating a global scaling factor between the predicted depth and the sensor depth.

However, unlike the environments used in NovaFlow, our settings exhibit much larger variations and more frequent movements along the depth axis. We observed that using a single median-based scalar often leads to substantial depth estimation errors in such scenarios.

To address this, instead of relying on a single global statistic, we compute a pixel-wise depth rescaling map by directly comparing the predicted and sensor depth maps across all pixels. We then apply a Gaussian blur to this map to obtain a smooth depth-rescaling field, and multiply this smoothed map with the predicted 3D trace to correct its z-values.
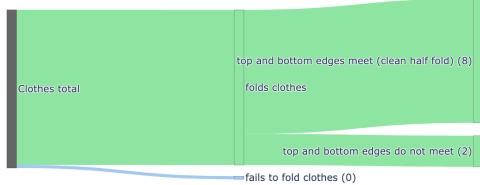
D.3. Implementation details for the baselines

3DFlowaction The official 3DFlowAction implementation relies on a filtering pipeline to extract object masks. This pipeline combines a language-conditioned object detector with a heuristic process designed to remove the robot gripper. However, when only a single image is provided as input, the detector often fails to identify the target object reliably or to filter out the robot end-effector. As a result, the predicted masks frequently include parts of the robot itself.
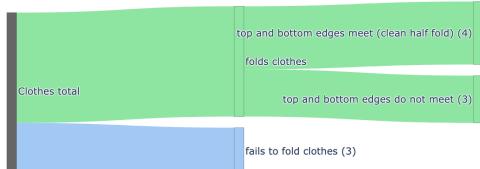
To mitigate this issue, and consistent with the official implementation, we provide a short sequence of images containing minimal robot motion. These slight temporal changes help the filtering pipeline correctly identify the robot gripper, allowing the final filtered region to closely match the ground-truth object mask. To ensure the generated object mask aligned with the expected robot movement, we manually checked whether the detected bounding box is aligned with the ground truth bounding box of the target objects.

NovaFlow The original NovaFlow pipeline includes a grasp-proposal module and a trajectory-planning module. Since our tasks do not involve grasping, we remove the grasp-proposal stage and initialize the robot in a state where it is already holding the object, ensuring a fair comparison.
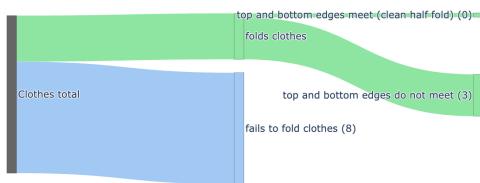
In addition, the official NovaFlow implementation uses MegaSaM [29] with TAPIP3D for camera pose estimation and depth prediction. For fair comparison, we replace the MegaSaM component with the same fine-tuned VGGT [44] depth and pose predictor from SpatialTrackerV2 [46] that we use in TraceForge. This VGGT-based predictor achieves similar accuracy while providing substantially faster infer-
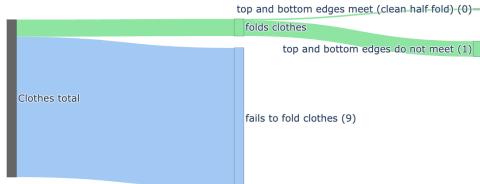
(a) TraceGen (Warmed up with robot demo)

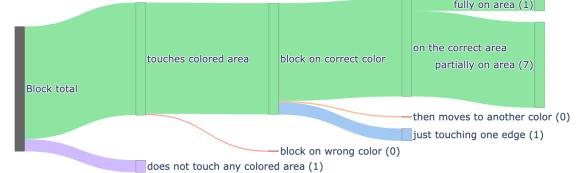(b) TraceGen (Warmed up with human demo)

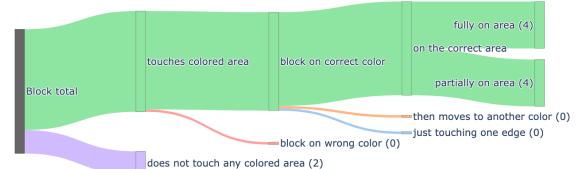
(c) NovaFlow (Veo3.1)

(d) NovaFlow (Wan2.2)

Figure 9. Failure-mode breakdown for the *Clothes* task.

(a) TraceGen (Warmed up with robot demo)

(b) TraceGen (Warmed up with human demo)

(c) NovaFlow (Veo3.1)

(d) NovaFlow (Wan2.2)

Figure 10. Failure-mode breakdown for the *Block* task.

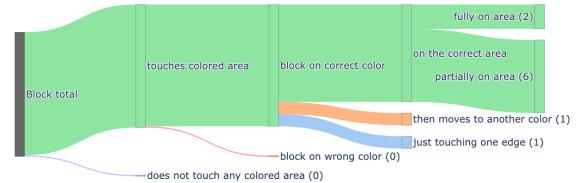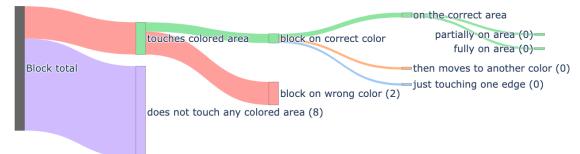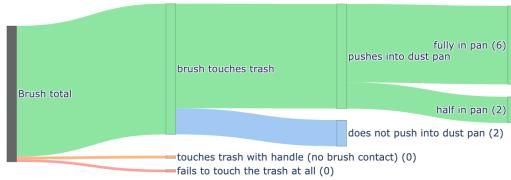ence by avoiding expensive 3D optimization. Also, for each NovaFlow, we use the following prompt:

- Veo3.1 (Clothes): The robot smoothly picks up the pants leg and folds the garment in half.
- Veo3.1 (Brush): The robot arm moves the broom toward the yellow trash, sweeps it forward, and guides it into the dustpan.
- Veo3.1 (Block): In the picture you see robot, blue cube, red paper, and purple paper. The blue cube is right underneath the robot arm. These are the only things that you need to pay attention. The robot arm grabs the blue cube and put it on the top of the purple paper.
- Veo3.1 (Ball): The robot arm in the image moves to grab the tennis ball and put it into the box in the image.
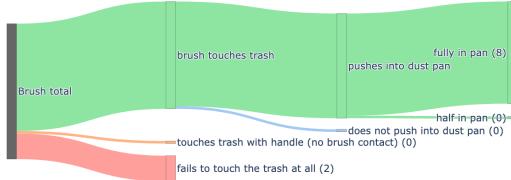- Wan2.2 (Clothes): The robot's end effector grasps the

black pants, positions them flat, then folds them in half to create a compact folded shape.
- Wan2.2 (Brush): The robot's end effector grips the broom handle and sweeps the yellow trash into the dustpan with deliberate strokes..
- Wan2.2 (Block): The robot's end effector grasps the LEGO block, lifts it upward, moves it above the purple notebook, then lowers it onto the notebook center.
- Wan2.2 (Ball): The robot's end effector grasps the tennis ball, lifts it upward, then moves it horizontally toward the box and lowers it inside.

AVDC For AVDC, we follow the exact evaluation setup used in the NovaFlow paper. Specifically, we isolate the video-generation component from AVDC and combine it

(a) TraceGen (Warmed up with robot demo)

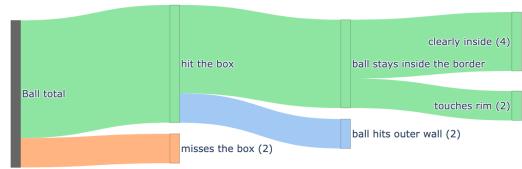(b) TraceGen (Warmed up with human demo)

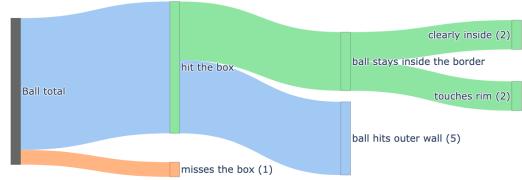(c) NovaFlow (Veo3.1)

(d) NovaFlow (Wan2.2)

Figure 11. Failure-mode breakdown for the *Brush* task.

(a) TraceGen (Warmed up with robot demo)

(b) TraceGen (Warmed up with human demo)

(c) NovaFlow (Veo3.1)

(d) NovaFlow (Wan2.2)

Figure 12. Failure-mode breakdown for the *Ball* task.

with the same 3D point-tracking and depth-estimation modules used in both TraceForge and NovaFlow, ensuring consistency across all baselines.
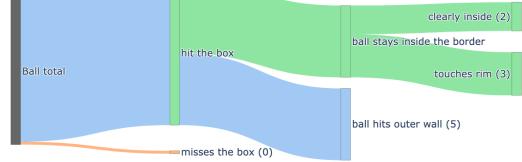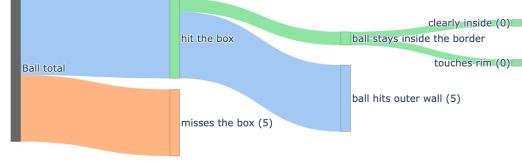
Inference latency measurement. All baseline runtimes were measured on an NVIDIA RTX A5000 except Wan 2.2 and Veo 3.1. Wan 2.2 could not fit on a single A5000, so we enabled inference using multiple GPUs—an unavoidable choice that in fact favors the baseline. Veo 3.1 is closed-source, and its latency is reported based on average API response time, which again places the baseline at an advantage.

E. Failure modes analysis

To better understand the behavior of each method beyond success rates, we provide a detailed failure-mode analysis across all four tasks and four model configurations: (i) **TraceGen** with robot-domain warmup, (ii) **TraceGen** with human video warmup, (iii) **NovaFlow** (Veo3.1), and (iv) **NovaFlow** (Wan2.2). For each combination of task and method, we collect every executed trial and categorize the outcome into fine-grained success and failure types based on object interaction quality and task completion criteria.

We visualize these distributions in Sankey diagrams, which reveal how trials progress from the total pool of attempts (left) into distinct outcome modes (right). This representation highlights the long-tail structure of failure patterns—showing whether errors arise early (e.g., incorrect
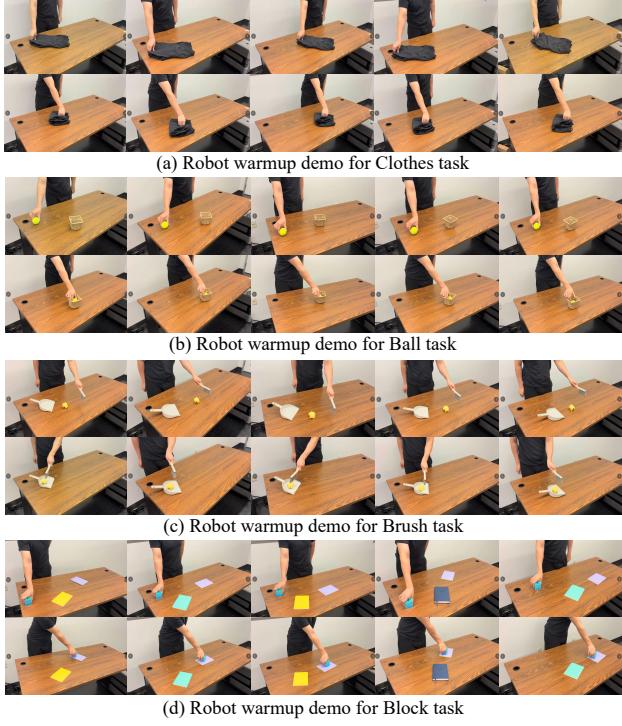
(a) Robot warmup demo for Clothes task

(b) Robot warmup demo for Ball task

(c) Robot warmup demo for Brush task

(d) Robot warmup demo for Block task

Figure 13. Human warmup demonstrations for all four tasks, showing first (top) and final (bottom) frames of each handheld video.

(a) Robot warmup demo for Clothes task

(b) Robot warmup demo for Ball task

(c) Robot warmup demo for Brush task

(d) Robot warmup demo for Block task

Figure 14. Robot warmup demonstrations for all four tasks. Top row shows the first frame of each demo; bottom row shows the final frame.

approach) or late in the trajectory (e.g., partial completion, drift during final alignment).

E.1. Warmup data

In this section, we visualize all warmup demonstrations used in our experiments. As described in the main text, TraceGen is adapted to each task using a lightweight warmup stage, which serves to translate the embodiment-agnostic 3D traces into the action space of the target robot or tasks.

We consider two warmup regimes:

- **Robot→Robot (same-embodiment warmup).** Five in-domain robot demonstrations are provided for each task. These clips differ from the evaluation setting in object layout and initial robot pose, ensuring that warmup does not simply memorize target configurations.

- **Human→Robot (cross-embodiment warmup).** Five uncalibrated human videos (3–4 seconds each) are captured per task using a handheld phone. These clips differ substantially from the robot setting in background, lighting, embodiment, and object placement.

Each figure below shows all warmup demonstrations for the four tasks (*Clothes*, *Block*, *Brush*, *Ball*). For each demo, the **top row displays the first frame** and the **bottom row displays the final frame**.
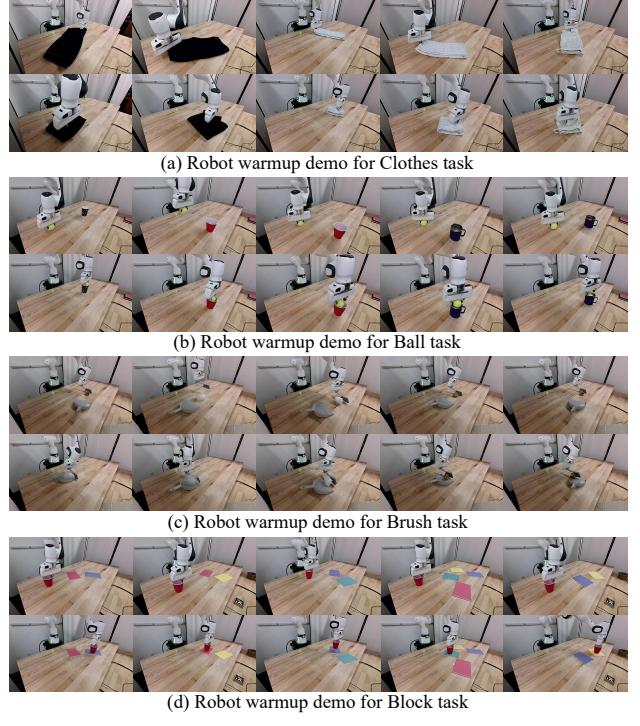
E.2. Long horizon experiments

To assess whether TraceGen's predicted trace can be composed into longer multi-step behaviors, we evaluate the model on a long-horizon *Sorting* task. The goal is to separate blocks from white trash items: each block must be placed in a designated green region, and each trash item must be placed in the red region. We collect five human teleoperation demonstrations of the full sorting process and segment them into four primitive subtasks, which serve as warmup data for TraceGen.

The sorting procedure consists of the following four consecutive subtasks:

1. Place the left trash on the red paper.
2. Place the pink LEGO block on the green paper.
3. Place the blue LEGO block on the green paper.
4. Place the right trash on the red paper.

Completing all four subtasks in sequence constitutes a successful sorting episode.

Use of scripted grasping. Because TraceGen models only the 3D trace component of manipulation and does not include an external grasping module, we assume access to a pre-defined scripted policy for picking up each object. This policy moves the robot to a preset grasping pose, enabling the placing skill generated by TraceGen to begin from a

Figure 15. Visualization of the long-horizon Sorting task, showing the four sequential placement subtasks from left to right.

consistent home configuration.

Results. We compare TraceGen initialized with pretraining from the TraceForge-123k dataset ("Warmed up from TraceGen") against a *From Scratch* model trained only on the four warmup segments. Table 4 reports per-step success rates across 10 rollouts. While the pretrained model occasionally fails the first trash placement, it maintains high performance on all subsequent subtasks. In contrast, the scratch model exhibits compounding errors over time, with success rates degrading markedly in the later steps.

Table 4. Long-horizon Sorting task: per-subtask success rates (left to right indicates temporal order).

Model	Left Trash $\rightarrow$	Pink Block $\rightarrow$	Blue Block $\rightarrow$	Right Trash
Warmed up from TraceGen	0.8	0.8	0.8	0.8
From Scratch	1.0	0.8	0.5	0.4

Overall, these long-horizon results show that Trace-Gen's pretrained motion priors enable stable composition of primitive placing behaviors, mitigating error accumulation across sequential subtasks. Although the model was not explicitly optimized for extended planning, its compact 3D trace representation supports reliable stitching of skills over longer task horizons.