ACT4R: ACTION-CONDITIONED 4D RECONSTRUCTION FOR KINEMATICS-FREE CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Precise control of movement across various robotic embodiments is a longstanding challenge. Modeling complex robotic structures requires significant time, dedicated sensors, and manual labeling. Recent advancements in 4D representation learning offer an interesting alternative: can we leverage 4D features lifted from 2D observations to densely map a robot's spatial movement based on the action signal? To this end, we propose Act4R, a framework that builds on 3D foundation model to predict a robot's 3D scene flow with any given action. Act4R introduces a Spatial-Action Encoder, which encodes the current 2D robot state and its action signals into embeddings that represent future robot state in 3D. Then, both the current and future spatial embeddings are decoded into point maps to estimate action-conditioned scene flow. To effectively learn the mapping between action and future state, Act4R introduces spatial-action fusion in its transformer-based encoder, and a spatial alignment loss that guides the encoder for future flow prediction. After training on some samples of a given embodiment, Act4R can perform inverse control by optimizing the signal to achieve the desired scene flow. We evaluate Act4R on a public real-world dataset and synthetic data collected in simulator. Experiments on various robotic platforms show that Act4R outperforms the previous SoTA in scene flow prediction. Our method also performs better in inverse control, both on the action accuracy metric and in closed-loop control in simulation.

1 INTRODUCTION

Robotic control requires a reliable mapping from raw action signals to spatial movements. While most research uses well-measured and reliable embodiment with precise kinematics, e.g., Franka (Haddadin et al., 2022), Universal Robot (Universal Robots, 2024), etc., there exist many more robots with which this mapping is not reliably specified. This can be due to manufacturing variation, limited kinematic sensing, or a design not originally intended for precise control, and is often true for low-cost or custom-built robots. Relying on bespoke modeling to map raw signals to spatial movements (Wong et al., 2025) makes it difficult to support an open-ended range of embodiments. Our goal is a common visual calibration framework that learns this mapping for each embodiment, enabling downstream tasks without a manually specified kinematic model.

Prior efforts to control robots with unmeasured kinematics have explored marker-based visual servoing and specialized machine learning architectures. Early visual servoing techniques utilized high-contrast physical markers or LED arrays to track deformation (Lagneau et al., 2020; AlBeladi et al., 2022; Bern et al., 2020); however, these methods often fail under occlusion or when the robot's surface undergoes significant warping. Subsequent machine learning approaches attempted to bypass these physical constraints by training end-to-end regressors to map pixels directly to actuator commands (Chin et al., 2020; Toshimitsu et al., 2022). Recently, Neural Jacobian Field (NJF) leverages radiance field as a representation to recover a robot's Jacobian matrix and action-conditioned scene flow at a pixel-level density (Li et al., 2025), demonstrating a promising direction to map complex kinematics at full resolution. However, training an NJF model requires substantial effort and imposes practical limitations.

Instead of learning 3D representations from scratch, integrating strong 3D priors can improve data and learning efficiency of action-conditioned scene flow estimation. Recent 3D Vision Foundation Models (VFM) (Wang et al., 2024; Leroy et al., 2024; Wang et al., 2025a)

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

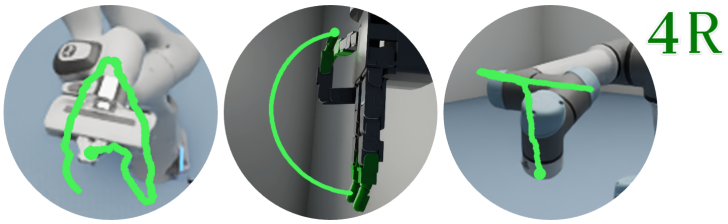


Figure 1: Robots can be controlled by Act4R without prior kinematics, as visualized by the green 2D tracks of three platforms.

are trained on a large amount of 3D datasets, allowing them to internalize rich spatial representations. These models exhibit remarkable robustness and zero-shot generalization capability across complex environments. Building upon these foundations, recent works (Zhang et al., 2025; Feng et al., 2025) have extended these capabilities to dynamic scenes, exploring the temporal evolution of geometry and motion. Particularly, St4RTrack (Feng et al., 2025) demonstrates that scene flows can be estimated effectively by finetuning 3D VFMs on a small 4D labeled dataset. This raises an interesting question: can 3D VFMs be adapted to a novel modality, i.e., the robot's raw action signal, to estimate its future scene flow and enable action-to-motion calibration across different embodiments?

To this end, we introduce **Act4R**, a method for action-to-motion calibration that predicts a robot's future *spatial* movement from the current 2D observation and an input action command. Rather than requiring a manually specified kinematic model, Act4R learns this mapping from observations and action signals for each embodiment. The main challenge for Act4R lies in finding an *efficient* and *performant* mapping from action signals to spatial representations. Previous works in action encoding, e.g., for Vision-Language-Action (VLA) models, focus on descriptive, language-based actions (Kim et al., 2025; Qu et al., 2025; Li et al., 2026). Gr00t N1 (Bjorck et al., 2025) introduces a three-layer MLP to convert action into the token space; however, these designs are ineffective at inferring spatial concepts from raw action signals. In contrast, 3D VFMs encode images into spatial embeddings, but have not been applied to action signals. Act4R introduces a novel Spatial-Action Encoder (SAE) module, which cross-attends the spatial embedding from a 2D observation and an action signal. SAE outputs an updated spatial embedding, which is then fed to a 3D VFM decoder to generate point maps for scene flow estimation. To ensure that SAE is learning a useful representation for the 3D VFM decoder, Act4R imposes a spatial alignment loss on SAE. Specifically, SAE's output should align with the spatial embedding of the *future* image.

During training, Act4R internalizes the robot's geometric deformations by observing a robot execute a diverse range of actions from various angles. For geometric supervision, we use sensor measured depth. A further ablation study also shows that depth estimated by a 3D VFM can also be used for effective control mapping, as demonstrated by the green tracks produced by different robots in Fig. 1. Once the VFM is finetuned for a specific robot, Act4R can be used for inverse control. On data collected from four real-world robotic platforms, we demonstrate that Act4R improves scene flow prediction and inverse action estimation while requiring less training time than NJF (Li et al., 2025). To evaluate the same calibration framework across additional embodiments, we collect a simulation dataset covering three robotic embodiments and establish a common evaluation protocol for unknown-embodiment control. Experiments on these embodiments demonstrate effective closed-loop inverse control, complementing the offline evaluation on real-world data.

In summary, our main contributions are:

1. We propose **Act4R**, an action-conditioned scene flow estimation framework for calibrating action-to-motion mappings across robotic embodiments without manually specified kinematic models.

2. Within Act4R, we propose a **Spatial-Action Encoder** (SAE). SAE introduces a spatial-action-fusion mechanism to ensure spatial and action embeddings are modeled jointly; a spatial alignment loss is introduced to ensure a correct mapping from action signals to spatial embeddings is learned.

3. Based on experiments over four real-world robotic platforms, our framework significantly improves upon previous approaches (Li et al., 2025). We additionally collect a simulation dataset and establish a shared evaluation protocol across three robotic embodiments, providing a portable benchmark for unknown-embodiment control.

2 Related Works

2.1 Robotic Control Mapping

In conventional robotic manipulation, the control framework typically relies on a manually designed Jacobian system that maps control commands to the corresponding velocities of the end-effector (Whitney, 1969). With the rapid development of deep learning, recent works, as surveyed in (Weinberg et al., 2024) have explored modeling robotic systems and their Jacobians with neural networks. Most work require a bespoke modeling scheme for specific robotic platforms, e.g., exhaustive joint-level DoF annotations to map actuator space to Cartesian space. To bypass these modeling hurdles, some frameworks (Yip & Camarillo, 2014; Lagneau et al., 2020; Bern et al., 2020; AlBeladi et al., 2022; Toshimitsu et al., 2022) utilize marker-based visual servoing to derive low-level control signals. While effective for tracking, these methods rely on manual instrumentation, such as ArUco markers or color-coded segments, which limits their applicability to unstructured real-world scenarios where such modifications are impractical or impossible.

In recent years, Vision-Language-Action (VLA) models have been introduced to ground high-level semantic knowledge into low-level robotic control. While early VLA models relied primarily on 2D semantic alignments, recent advancements have introduced more sophisticated spatial understanding capabilities (Qu et al., 2025; Kim et al., 2025; Li et al., 2026). Huang et al. (2026) predict action-conditioned 3D scene flow using robot point-flow actions constructed from known robot geometry and forward kinematics. We note that these works are built upon rigid-body robotic platforms with well-articulated kinematics and controls and cannot be transferred to platforms where control is unknown.

Parallel to high-level VLA reasoning, Neural Jacobian Field (NJF) (Li et al., 2025) has emerged as a novel paradigm for inverse robotic control by representing kinematics as a continuous, coordinate-based Neural Field $\mathbf{J}(\mathbf{x}) = \partial\mathbf{x}/\partial\mathbf{u}$. NJF is optimized by observing robotic movement across various view angles and updates itself through ray-tracing an action-conditioned, differentiable displacement field (Pumarola et al., 2021; Park et al., 2021). By modeling every voxel of the robot, NJF provides a generalizable approach for high DoF inverse control. Act4R builds upon this paradigm but in a different way. Act4R leverages 3D VFMs to represent spatial relationship between an image and a control signal; as such the choice of representation is in point maps and not Neural Fields.

2.2 Progress in 3D Vision Foundation Models

Significant progress has been made recently in feed-forward neural networks that can estimate geometry, camera poses and feature tracks from two-view or multi-view inputs. DUSt3R and MASt3R (Wang et al., 2024; Leroy et al., 2024) introduce the point-map representation and canonical camera coordinate framework that enables efficient learning of geometry. Subsequent work extends this formulation to continuous captures (Wang et al., 2025c), multi-view (Wang et al., 2025a; 2026; Lin et al., 2025), over long sequences (Zhuo et al., 2025; Deng et al., 2025), etc. Moving beyond 3D modeling, Shape of Motion (Wang et al., 2025b) and Dynamic Point Maps (Sucar et al., 2025) emphasize the coupling of geometry and motion, providing versatile representations for reconstructing complex deformations. More recently, foundation models like MonST3R (Zhang et al., 2025) and St4RTrack (Feng et al., 2025) have simplified the estimation of geometry in the presence of motion. By leveraging large-scale visual priors, these frameworks enable simultaneous 4D reconstruction and point-to-point tracking, providing the dense supervision required for the training of our action-conditioned model. Act4R is built upon the general encoder-decoder Transformer architecture of these 3D/4D and inherits some of the prior weights; however, Act4R takes an image and an action as inputs, instead of an image pair. As such, Act4R introduces its own encoder to convert action information into spatial embeddings.

3 Methodology

Act4R seeks to map actions to a robot's spatial movement from observations only. As shown in Fig. 2, Act4R predicts future robotic state by taking the current image I_t and a control action A_t as inputs. We detail the overall problem setting and Act4R framework in Sec. 3.1. Within Act4R, we introduce a Spatial-Action Encoder (SAE), a specialized action encoder to map control commands into high-dimensional spatial embeddings, as detailed in Sec. 3.2. At training time, we use a spatial

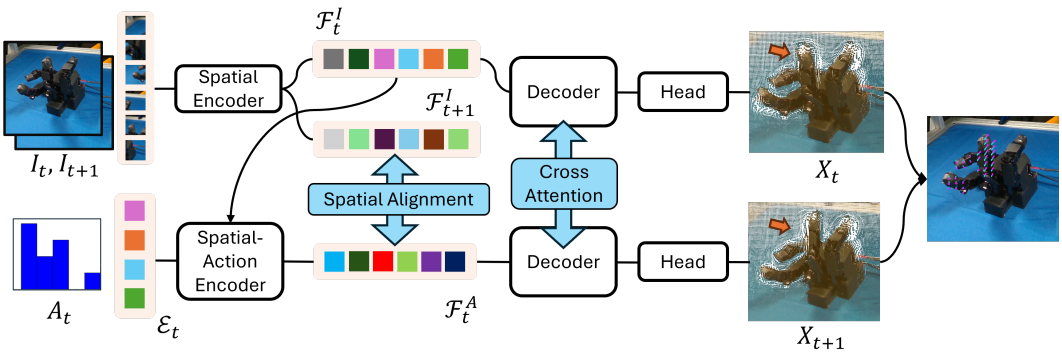


Figure 2: **Method Overview**: Given a 2D observation of the robot I_t and the action A_t, Act4R introduces a Spatial-Action Encoder to estimate the future placement of the robot at time $t+1$. A learnable action embedding $\mathcal{E}_t$ is introduced to improve efficiency. **At training time**, SAE's output feature $\mathcal{F}_t^A$ is aligned with the future spatial feature $\mathcal{F}_{t+1}^I$. At inference time, I_{t+1} is not used.

alignment loss to constrain the output from SAE with the spatial embeddings of the future image I_{t+1}, described in Sec. 3.3. Following this alignment, the decoder is trained end-to-end under dense reconstruction and tracking supervision. Finally, we invert the model to derive the optimal action required to achieve the desired robotic motion.

3.1 Problem Settings and the Act4R Framework

At time t, we define the 3D pointmap $\mathbf{X}_t$ of the robot as its system state, and the action vector A_t as the control command. In reality, it is often hard to have direct access to $\mathbf{X}_t$; instead, we can easily have access to a 2D observation I_t of the robot. The goal of Act4R is to predict the current and future state of the robot $\mathbf{X}_{t,t+1}$ by training on a series of I_t, A_t, and I_{t+1} collected over time. Subsequently, we can obtain the scene flow $\mathcal{S}$ and manipulate the robot based on predicted scene flow.

Our approach leverages 3D VFMs priors (Feng et al., 2025; Wang et al., 2024; 2025a), which train a neural network f_θ to regress dense 3D point maps from an image pair $(\mathbf{I}_i, \mathbf{I}_j)$:

$$f_\theta(\mathbf{I}_i, \mathbf{I}_j) = \{\mathbf{X}_i^i, \mathbf{X}_j^i\}, \tag{1}$$

$\mathbf{X}_j^i \in \mathbb{R}^{N\times 3}$ represents a dense set of 3D points from I_j in the camera coordinate frame of I_i. Generally speaking, 3D VFMs can be split into a spatial encoder $f_{\theta_{enc}}$ and decoder $f_{\theta_{dec}}$, i.e.:

$$f_{\theta_{enc}}(\mathbf{I}_i) = \mathcal{F}_i^I, f_{\theta_{enc}}(\mathbf{I}_j) = \mathcal{F}_j^I, f_{\theta_{dec}}(\mathcal{F}_i^I, \mathcal{F}_j^I) = \{\mathbf{X}_i^i, \mathbf{X}_j^i\}, \tag{2}$$

where $f_{\theta_{enc}}$ first lifts images into spatial embeddings, and $f_{\theta_{dec}}$ performs cross attention between two embeddings to generate 3D point maps. Exploiting the flexibility of Transformer-based architectures, which process inputs as sequences of tokens, we substitute the target image $\mathbf{I}_j$ with an action vector $\mathbf{A}_t$. This modification enables the framework to generate a 3D point map for the current observation $\mathbf{I}_t$ conditioned on $\mathbf{A}_t$, the signal that implies future robotic state. Because the camera pose remains static between observations, the coordinate frame is static, and the predicted point map varies only in the temporal dimension. Thus, our model formulation becomes:

$$f_{\theta_{enc}}(\mathbf{I}_t) = \mathcal{F}_t^I, f_{\theta_{enc}}^{\text{SAE}}(\mathbf{A}_t, \mathcal{F}_t^I) = \mathcal{F}_t^A, f_{\theta_{dec}}(\mathcal{F}_t^I, \mathcal{F}_t^A) = \{\mathbf{X}_t, \mathbf{X}_{t+1}\}. \tag{3}$$

As shown in Fig. 2, Act4R uses an SAE to lift A_t into spatial embedding. It is then fed into a 3D VFM decoder with the spatial embedding from I_t, and produces $X_{t,t+1}$.

3.2 Spatial-Action Encoder

How can we find the proper mapping from action signals A_t to the future 3D robotic state $\mathbf{X}_{t+1}$, given the current state in I_t and $\mathcal{F}_t^I$? A naive approach may be to encode action signals into action embeddings, and train a decoder architecture that cross-attends with $\mathcal{F}_t^I$. While this may work well if there exists a large amount of image-action pairs and their 3D ground truth, we generally do not have

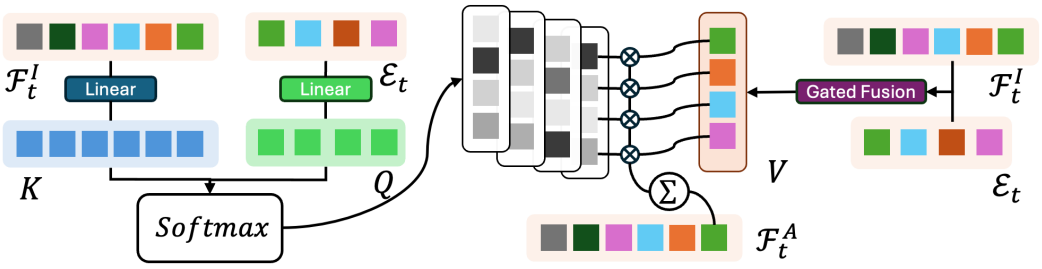


Figure 3: The architecture of Spatial-Action Encoder.

enough data to train a decoder from scratch to interpret the spatial-action relationship. As such, we leverage existing spatial prior in 3D VFMs, where the decoder are trained to take spatial embeddings encoded from two images.

As shown in Fig. 3, we move the fusion of $A_t \in \mathbb{R}^M$ and $\mathcal{F}_t^I \in \mathbb{R}^{N\times C}$ into a Spatial-Action Encoder. SAE first passes A_t through an MLP to construct the action embedding $\mathcal{E}_t \in \mathbb{R}^{M\times C}$. Similar to conventional Transformer architecture, $\mathcal{F}_t^I$ and $\mathcal{E}_t$ are linearly projected to form the Key and Query tokens. Attention weights $W \in \mathbb{R}^{M\times N}$ are computed through a dot product and softmax operation. Conventional Cross Attention mechanism typically expresses Value token in one modality, e.g. image tokens query on text token space. In our case, both action $\mathcal{E}_t$ and $\mathcal{F}_t^I$ contain important spatial information about $\mathbf{X}_{t+1}$; therefore, we construct the Value token $V \in \mathbb{R}^{N\times C}$ by a gated channel-wise fusion of the image tokens and a motion-informed context. We average the action tokens $\mathcal{E}_t \in \mathbb{R}^{M\times C}$ over the M joints, broadcast the result to the N image patches, and apply a linear projection $W_v \in \mathbb{R}^{C\times C}$ to obtain $\tilde{\mathcal{E}}_t \in \mathbb{R}^{N\times C}$. A per-patch gate $g = \sigma(\mathrm{MLP}([\mathcal{F}_t^I; \tilde{\mathcal{E}}_t])) \in \mathbb{R}^{N\times 1}$ then interpolates between the two sources as $V = (1-g)\odot \mathcal{F}_t^I + g \odot \tilde{\mathcal{E}}_t$. After multiplying the attention weights with the Value tokens, we obtain an encoded action token $\mathcal{E}_{t'} \in \mathbb{R}^{M\times C}$. We then feed $\mathcal{E}_{t'}$ as Key and $\mathcal{F}_t^I$ as Query in the next layer. In the second layer the Key already has length M, so $V \in \mathbb{R}^{M\times C}$ is formed with the same gated fusion, now interpolating $\mathcal{E}_{t'}$ and a per-joint projection of $\mathcal{E}_t$; we do not average over joints or concatenate $\mathcal{F}_t^I$. This fusion mechanism allows action embedding to be continuously integrated into the spatial embedding, rather than aggregating only on Value tokens from $\mathcal{F}_t^I$. The final output is denoted as $\mathcal{F}_t^A \in \mathbb{R}^{N\times C}$.

3.3 Future State Spatial Alignment

A direct "cold start" through end-to-end training with $\mathbf{A}_t$ and SAE risks catastrophic interference with the pre-trained weights, potentially distorting the high-fidelity spatial representations internalized during large-scale pre-training. To preserve these geometric priors, we introduce a latent alignment constraint. At *training time*, we introduce an additional spatial embedding $\mathcal{F}_{t+1}^I$, which represents the pseudo ground truth spatial embedding that we hope SAE can learn from.

To this end, we first construct an oracle version of 3D VFMs f_ϕ by finetuning on the following predictions:

$$f_\phi(\mathbf{I}_t, \mathbf{I}_{t+1}) = \{\mathbf{X}_t, \mathbf{X}_{t+1}\}. \tag{4}$$

Act4R initializes its spatial encoder and decoder weights from f_ϕ, and freezes its spatial encoder during training. We extract the intermediate spatial embedding $\mathcal{F}_{t+1}^I$ by passing I_{t+1} to the spatial encoder, and force $\mathcal{F}_t^A$ to match $\mathcal{F}_{t+1}^I$. By doing so, the action encoder inherits the foundation model's visually grounded spatial logic. This ensures that control inputs are mapped into a physically meaningful 3D manifold, maintaining the integrity of the model's predictive capacity. The alignment loss is formulated as:

$$\mathcal{L}_{feat} = \left\|\mathcal{F}_t^A - \mathcal{F}_{t+1}^I\right\|_2^2. \tag{5}$$

3.4 Action-Conditioned 4D Prediction

Scene Flow Loss. To supervise the geometric deformation induced by the action, we utilize the state-of-the-art point tracking framework CoTracker (Karaev et al., 2024; 2025) as pseudo ground truth. Because the model predicts 3D coordinates while CoTracker provides 2D temporal correspondences

$(x^{gt}_{t,n}, x^{gt}_{t+1,n})$, we lift both endpoints with the depth maps $X^{\mathrm{gt}}_{t,n} = \Pi^{-1}(x^{gt}_{t,n}; D_t)$, $\mathcal{S}_{gt,n} = X^{\mathrm{gt}}_{t+1,n} - X^{\mathrm{gt}}_{t,n}$, where $\Pi^{-1}(\cdot; D)$ denotes the back-projection with depth D. The predicted scene flow at CoTracker correspondence n is $\mathcal{S}_n = X_{t+1,n} - X_{t,n}$. Since the foundation model is scale-ambiguous, we first align the scale of gt and predicted over all pixels $\mathcal{V}$ with valid depth $\hat{\mathcal{S}}_n$ and calculate loss:

$$\hat{\mathcal{S}}_n = s_{flow}\mathcal{S}_n, \qquad s_{flow} = \frac{\sum_{p\in\mathcal{V}} X^{\top}_{t,p} X^{\mathrm{gt}}_{t,p}}{\sum_{p\in\mathcal{V}} \|X_{t,p}\|_2^2}, \qquad \mathcal{L}_{flow} = \frac{1}{N}\sum_{n=1}^{N} \left|\mathcal{S}_{gt,n} - \hat{\mathcal{S}}_n\right|_2. \tag{6}$$

Trajectory Loss. We also use 2D track generated by CoTracker as a supervision directly. The post-action point map is projected onto image plane, $x_{t+1} = \Pi_{t+1}(\mathbf{X}_{t+1})$. The same scale ambiguity remains in the image plane, so we rescale the predicted track about the image center c and then calculate the trajectory loss $\mathcal{L}_{traj}$:

$$\hat{x}_{t+1} = (x_{t+1} - c)\, s_{traj} + c, s_{traj} = \frac{1}{N}\sum_{n=1}^{N} \frac{\left\|x^{gt}_{t+1,n} - c\right\|_2}{\left\|x_{t+1,n} - c\right\|_2}, \mathcal{L}_{traj} = \frac{1}{N}\sum_{n=1}^{N} \left\|x^{gt}_{t+1,n} - \hat{x}_{t+1,n}\right\|^2. \tag{7}$$

Additionally, we follow Feng et al. (2025) in applying a depth loss $\mathcal{L}_{depth}$ to supervise 3D geometry, and a consistency loss $\mathcal{L}_{cons}$ to promote temporal stability. More details are included in the Appendix. The overall loss during training is:

$$\mathcal{L} = \lambda_{feat}\mathcal{L}_{feat} + \lambda_{flow}\mathcal{L}_{flow} + \lambda_{traj}\mathcal{L}_{traj} + \lambda_{depth}\mathcal{L}_{depth} + \lambda_{cons}\mathcal{L}_{cons} \tag{8}$$

3.5 Inverse Control from Future Scene Flow

Once Act4R is trained to model the future scene flow, we can use it for inverse control. Unlike conventional kinematics methods, Act4R models a robot's movement at every point and applies to a wide range of robots without further efforts. Consider that we want the robot to move from a current state to a desired state. This can be done by establishing the desired 2D or 3D flow $\mathcal{S}_{tgt}$ from the current observation I_t. We use $\ell 2$ optimization objective that finds the best A^* to achieve $\mathcal{S}_{tgt}$ through f_θ. In 3D form, this is:

$$A^*_t = \underset{A_t}{\operatorname{argmin}} \sum_{n=1}^{N} \left\|\Delta^{3D}(f_\theta(I_t, A_t)) - \mathcal{S}^{3D}_{tgt,n}\right\|_2^2, \Delta^{3D}(X_t, X_{t+1}) = X_{t+1,n} - X_{t,n}. \tag{9}$$

We can also project the 3D flow from Act4R to fit the 2D tracks.

$$A^*_t = \underset{A_t}{\operatorname{argmin}} \sum_{n=1}^{N} \left\|\Delta^{2D}(f_\theta(I_t, A_t)) - \mathcal{S}^{2D}_{tgt,n}\right\|_2^2, \Delta^{2D}(X_t, X_{t+1}) = \Pi(X_{t+1,n}) - \Pi(X_{t,n}) \tag{10}$$

We can implement this by freezing the network parameters f_θ and use popular optimizers, e.g., Adam (Kingma & Ba, 2014) to minimize Eq. 9 or Eq. 10. During inverse control, only the current observation and the desired motion flow are required.

4 Experiments

4.1 Experiment Setup

	3D Flow Error (mm) ↓				
Robot	Act4R	Act4R$_{\mathrm{NSA}}$	Act4R$_{\mathrm{N3D}}$	Act4R$_{\mathrm{Gr00t}}$	NJF (Li et al., 2025)
Allegro	**6.715**	7.947	112.113	10.704	11.607
HSA	**5.136**	5.740	58.704	5.757	7.420
Toy Arm	**6.820**	8.199	43.755	8.153	9.398
Pneumatic	**4.603**	5.130	43.955	5.257	10.744
Allegro Sim	**3.765**	4.283	74.590	8.333	17.666
Franka Sim	**6.675**	6.766	9.988	14.571	35.155
UR5e Sim	**4.787**	11.295	146.750	15.040	32.246
Mean	**5.500**	7.051	69.979	9.688	17.748

Table 1: Performance comparison of 3D flow error.

Real World Dataset. We use the dataset released by Li et al. (2025), which provides a collection of robotic platforms and sensor data. The dataset features four distinct robot types to evaluate generalization across varied embodiments: 1. Allegro Hand (Wonik Robotics, n. d.), a commercial four-fingered robotic hand, 2. HSA (Lipton et al., 2018), a unique platform utilizing four deformable, helical-actuator-driven legs, 3. Pneumatic Hand (Matusik et al., 2024), a 3D-printed pneumatic hand combining structural rigidity with soft-robotic flexibility, and 4. a low-cost, multi-jointed robotic arm designed for educational use (Lapeyre, 2015).

Simulation Dataset. Following the same settings, we collect three additional embodiments in simulators. In IsaacSim simulator (Gao et al., 2026), we use 1. Franka (Haddadin et al., 2022), 2. Allegro Hand (Wonik Robotics, n. d.) and 3. UR5e (Universal Robots, 2024) robots as the robotic platforms.

All real and simulated datasets include multi-view RGB video frames and depth maps, along with the control signal for each robot. The depth maps are measured by dedicated sensors for real data, and environment ground truth for synthetic data. For each robotic platform, the dataset comprises several hundred individual action sequences, every action sequence consists of 8-10 frames. We use 90% of the sequences as training and validation set and the rest as test set.

Training and Inference. All experiments are conducted with four NVIDIA RTX A40 GPUs. We leverage St4RTrack (Feng et al., 2025), a recent work on 4D reconstruction based on 3D VFMs (Wang et al., 2024; Leroy et al., 2024), as our base model. The Act4R architecture is trained for 10 epochs. We train our model individually for every robotic platform, taking approximately 16 hours. For baselines, NJF (Li et al., 2025) is trained for 32 hours.

4.2 Flow Prediction Accuracy

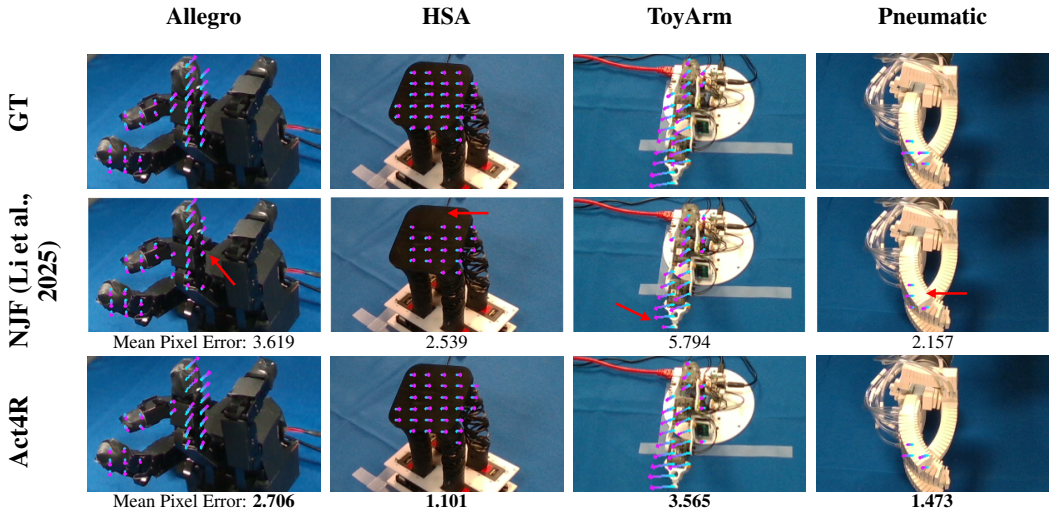


Figure 4: Predicted tracks across different experimental setups. GT denotes ground truth. Compared to Li et al. (2025), Act4R is much more accurate at predicting action-conditioned flow.

For scene flow evaluation, we follow a bidirectional protocol, evaluating every image pair in both forward and backward directions. During testing, the model takes I_t and A_t to predict the resulting 3D scene flow. To evaluate performance, we derive the ground-truth trajectories from the current and subsequent images in the dataset. Our primary evaluation metric is the 3D Flow Error between the predicted and ground truth trajectories. While CoTracker samples points uniformly across the image, we observed that most pixels remain stationary. To ensure our metric reflects the important motion tracking accuracy and not be overly smoothed, we compute the error exclusively for pixels where the ground truth exhibits a displacement. Same as Eq. 6, we lift the track from CoTracker to 3D and compare with predicted 3D Flow.

The comparative results are summarized in Tab. 1. Act4R$_{\text{NSA}}$ deploys no spatial alignment in Sec. 3.3, Act4R$_{\text{N3D}}$ does not use 3D flow loss, Act4R$_{\text{Gr00t}}$ uses an action encoder structure from Gr00t (Bjorck et al., 2025), which only self-attends on action signal with no spatial context. As shown in

Tab. 1, we observe that Act4R significantly outperforms Act4R$_{\text{Gr00t}}$, demonstrating the effectiveness of fusing action and spatial embeddings in SAE. Spatial alignment reduces the 3D flow error on all seven platforms. Finally, compared to NJF (Li et al., 2025) and as visualized in Fig. 4 in terms of 2D tracks, Act4R demonstrates significant improvement in flow accuracy. Compared to a neural field representation, which has to voxelize the entire space, a point-map representation on the robot surface can be more compact; furthermore, Act4R leverages spatial prior from 4D VFMs whereas NJF optimizes neural fields from scratch, leading to improvement in performance while reducing the training time. We also provide evaluations for 2D pixel track error in the Appendix.

4.3 Inverse Control Accuracy

Inverse control based on 2D tracks. In the dataset, each frame includes an input action attribute representing the robot's configuration. We compute $\mathbf{A}_t$ as the difference in the action space between two consecutive frames. To ensure numerical stability, we perform a linear normalization of these action vectors into a range of $[-1, 1]$. This normalization prevents gradient imbalances caused by varying actuator scales across the seven different robotic platforms. Based on the optimization objective in Eq. 10, we extract 2D flow $\mathcal{S}_{tgt}$ by computing the track between I_t and I_{t+1} and use it to obtain optimal control A_t^*. The experiment results are shown in Tab.2, where we use cosine similarity between A_t^* and groud truth action $\hat{A}_t^*$ to measure direction correctness. Act4R$_{\text{VGGT}}$ is the experiment that uses depth estimated by VGGT (Wang et al., 2025a). Magnitude is evaluated by $\ell 2$ error in the Appendix. With better scene flow estimation, Act4R outperform NJF (Li et al., 2025) on all platforms in terms of obtaining the correct actions to drive the robot.

Inverse control based on 3D tracks. While 2D tracks are easy to obtain based on camera observations and consistent with prior evaluation protocol (Li et al., 2025), robots move in 3D and should be able to follow 3D guidance. To this end, we also evaluate inverse control based on 3D tracks, which can be obtained by back-project 2D tracks into 3D space based on ground truth depth. We can then optimize Eq. 9 for the desired actions. Despite reasonable performance in following 2D tracks, Act4R$_{\text{N3D}}$ completely fails inverse control from 3D tracks, achieving only 0.085 on average in action accuracy, while Act4R and NJF Li et al. (2025) maintain similar performance. Act4R$_{\text{VGGT}}$ performs worse than Act4R; this may be due to inconsistent depth estimation that leads to noise in optimization. A detailed table is included in the Appendix.

	Action Cosine Similarity ↑					
Robot	Act4R	Act4R$_{\text{VGGT}}$	Act4R$_{\text{NSA}}$	Act4R$_{\text{N3D}}$	Act4R$_{\text{Gr00t}}$	NJF (Li et al., 2025)
Allegro	**0.679**	0.653	0.639	0.649	0.392	0.269
HSA	0.493	0.492	0.488	0.496	**0.498**	0.290
Toy Arm	**0.536**	0.531	0.526	0.529	0.501	0.273
Pneumatic	**0.376**	0.373	0.371	0.365	0.329	0.243
Allegro Sim	**0.475**	0.429	0.115	0.426	0.121	0.177
Franka Sim	**0.399**	0.384	0.355	0.383	0.288	0.185
UR5e Sim	**0.623**	0.616	0.603	0.616	0.447	0.503
Mean	**0.512**	0.497	0.442	0.495	0.368	0.277

Table 2: Performance comparison of action cosine similarity.

Closed-loop control. Beyond evaluation of single-step inverse control, we want to understand if Act4R is generalizable and can successfully move robots across space through a sequence of actions. Real robots are often inaccessible and hard to be configured consistently across experiments; to this end, we evaluate long-sequence control through three simulated robotic embodiments: Allegro, Franka, and UR5e. We note that the kinematics of these robots are unknown to the system.

To ensure that the desired control is achievable, we first generate a sequence of ground truth movement frames. We select the next ground-truth frame as target each step. Similar to before, we estimate the desired flow from the current observation of the robot with target frame and obtain the desired control through Eq. 10. This process is repeated till robots observation is close enough with target, then advance to the next ground-truth frame. As visualized in Fig. 5, Act4R can faithfully follow the instruction provided by GT tracks, while NJF (Li et al., 2025) often leads to incomplete movement (the pinky finger in Allegro) or very noisy movement (Franka and UR5e). We measure the trajectory

accuracy by the cumulative joint position error:

$$E_{\text{CJPE}} = \sum_{t=1}^{T} \sum_{i=1}^{J} \left| q_{i,t} - q_{i,t}^{\text{gt}} \right| \tag{11}$$

where $q_{i,t}^{\text{gt}}$ is the ground-truth position of joint i at frame t in radians, and $q_{i,t}$ is the corresponding joint position once the observation is close enough to that target, before advancing to the next frame. T is the number of ground-truth frames and J is the number of joints. Interestingly, Act4R trained on VGGT-estimated depth instead of ground truth depth can still lead to similar performance, despite slightly lower action accuracies in Tab. 2. For more visualizations of closed-loop control, we refer to Fig. 1, where various robots can be successfully guided to draw desired letters.

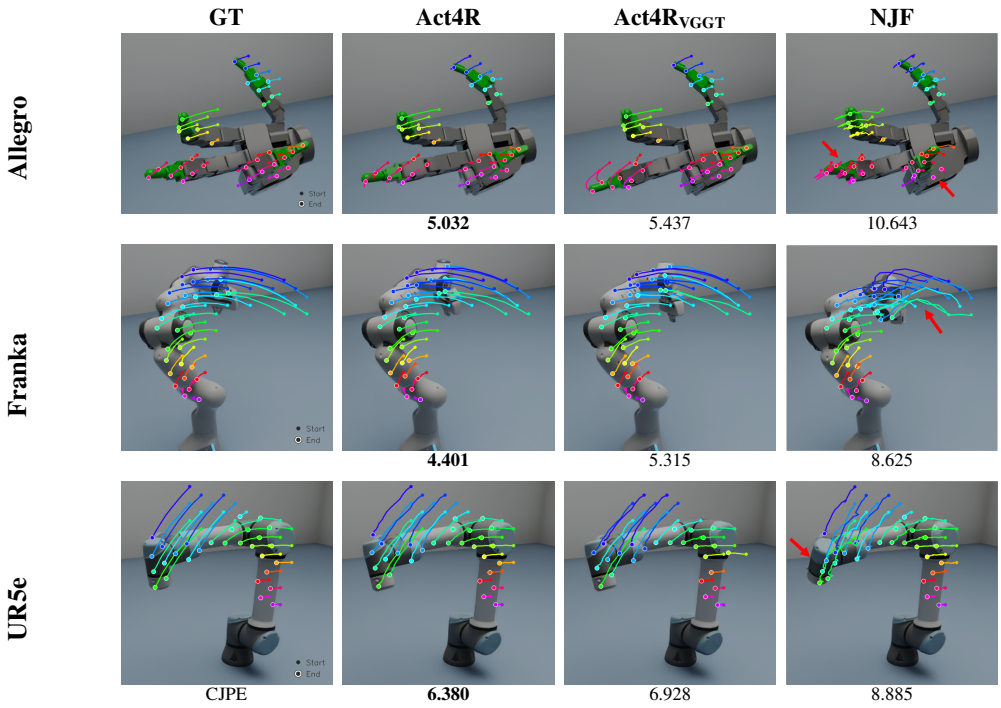


Figure 5: Long sequence inverse control visualization on simulated robots. GT is the ground-truth motion, Act4R is our method, Act4R$_{\text{VGGT}}$ uses VGGT depth, and NJF is the baseline. CJPE stands for Cumulative Joint Position Error, measured in radians.

5 Conclusion and Discussion

We propose Act4R, an algorithm designed to estimate future movements of robots given current 2D observation and an action signal. Act4R leverages strong spatial prior from 3D VFM by inheriting its encoder-decoder design; particularly, Act4R introduces a Spatial-Action Encoder to capture the spatial-action relationship. To effectively learn the future spatial embedding, SAE models the Value Token from the joint embedding of action and spatial embeddings; furthermore, we propose a Future State Spatial Alignment strategy to help SAE learn more effectively. Once trained, our model can be used to infer the required action given a desired scene flow through optimization. Experiments show that our method significantly outperforms NJF (Li et al., 2025) in scene flow prediction while incurring lower training cost. If measured depth is unavailable, depth predicted by a vision model can still be used. Despite its promising performance, Act4R still has room to improve. When the spatial displacement between images is small, vision-based inverse control can be noisy. Furthermore, more modern 4D Foundation Models (Luo et al., 2026; Zhang et al., 2026) are available as the backbone, though at much higher cost. Finally, the Act4R formulation can be used to model more complex interactions, e.g., with soft tissues. Future work will focus on addressing these challenges and evaluate on more complex settings.

AI USE STATEMENT

In this work, we used generative AI tools to improve the language, clarity, and readability of the manuscript. All scientific contributions, including the core ideas, methodology, experimental design, and interpretation of the results, are entirely the original work of the authors. All authors have carefully reviewed the final manuscript and take full responsibility for its content and claims.

REFERENCES

A. AlBeladi, E. Ripperger, S. Hutchinson, and G. Krishnan. Hybrid eye-in-hand/eye-to-hand image based visual servoing for soft continuum arms. *IEEE Robotics and Automation Letters*, 7(4): 11298–11305, 2022.

J. M. Bern, Y. Schnider, P. Banzet, N. Kumar, and S. Coros. Soft robot control with a learned differentiable model. In *Proceedings of the 2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)*, pp. 417–423. IEEE, 2020.

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.

K. Chin, T. Hellebrekers, and C. Majidi. Machine learning for soft robotic sensing and control. *Advanced Intelligent Systems*, 2(6):1900171, 2020.

Kai Deng, Zexin Ti, Jiawei Xu, Jian Yang, and Jin Xie. Vggt-long: Chunk it, loop it, align it–pushing vggt's limits on kilometer-scale long rgb sequences. *arXiv preprint arXiv:2507.16443*, 2025.

H. Feng, J. Zhang, Q. Wang, Y. Ye, P. Yu, M. J. Black, T. Darrell, and A. Kanazawa. St4rtrack: Simultaneous 4d reconstruction and tracking in the world. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8503–8513, Honolulu, HI, 2025. IEEE/CVF.

Sicong Gao, Maurice Pagnucco, Tomasz Bednarz, and Yang Song. Nvidia isaac sim: Enabling scalable, gpu-accelerated simulation for robotics. *arXiv preprint arXiv:2606.03551*, 2026.

Sami Haddadin, Sven Parusel, Lars Johannsmeier, Saskia Golz, Simon Gabl, Florian Walch, Mohamadreza Sabaghian, Christoph Jähne, Lukas Hausperger, and Simon Haddadin. The franka emika robot: A reference platform for robotics research and education. *IEEE robotics & automation magazine*, 29(2):46–64, 2022.

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.

N. Karaev, Y. Makarov, J. Wang, N. Neverova, A. Vedaldi, and C. Rupprecht. Cotracker3: Simpler and better point tracking by pseudo-labelling real videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6013–6022. IEEE/CVF, 2025.

Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together. In *Proc. ECCV*, 2024.

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 *Conference on Robot Learning*, pp. 2679–2713. PMLR, 2025.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

R. Lagneau, A. Krupa, and M. Marchal. Active deformation through visual servoing of soft objects. In *Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8978–8984. IEEE, 2020.

M. Lapeyre. *Poppy: Open-Source, 3D Printed and Fully-Modular Robotic Platform for Science, Art and Education*. PhD thesis, Université de Bordeaux, Bordeaux, France, 2015.

V. Leroy, Y. Cabon, and J. Revaud. Grounding image matching in 3d with mast3r. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 71–91, Cham, Switzerland, 2024. Springer Nature Switzerland.

Fuhao Li, Wenxuan Song, Han Zhao, Jingbo Wang, Pengxiang Ding, Donglin Wang, Long Zeng, and Haoang Li. Spatial forcing: Implicit spatial representation alignment for vision-language-action model. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026.

Sizhe Lester Li, Annan Zhang, Boyuan Chen, Hanna Matusik, Chao Liu, Daniela Rus, and Vincent Sitzmann. Controlling diverse robots by inferring jacobian fields with deep networks. *Nature*, 643 (8070):89–95, 2025.

Haotong Lin, Sili Chen, Junhao Liew, Donny Y Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

J. I. Lipton, R. MacCurdy, Z. Manchester, L. Chin, D. Cellucci, and D. Rus. Handedness in shearing auxetics creates rigid and compliant structures. *Science*, 360(6389):632–635, 2018.

Yihang Luo, Shangchen Zhou, Yushi Lan, Xingang Pan, and Chen Change Loy. 4rc: 4d reconstruction via conditional querying anytime and anywhere. *arXiv preprint arXiv:2602.10094*, 2026.

H. Matusik, C. Liu, and D. Rus. Directly 3d printed, pneumatically actuated multi-material robotic hand. In *Proceedings of the 2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2986–2992, Yokohama, Japan, 2024. IEEE.

K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5865–5874. IEEE/CVF, 2021.

A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10318–10327. IEEE/CVF, 2021.

D. Qu, H. Song, Q. Chen, Y. Yao, X. Ye, Y. Ding, Z. Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. arXiv preprint arXiv:2501.15830, 2025.

E. Sucar, Z. Lai, E. Insafutdinov, and A. Vedaldi. Dynamic point maps: A versatile representation for dynamic 3d reconstruction. arXiv preprint arXiv:2503.16318, 2025.

Y. Toshimitsu, K. Kawaharazuka, A. Miki, K. Okada, and M. Inaba. Dije: Dense image jacobian estimation for robust robotic self-recognition and visual servoing. In *Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2219–2226. IEEE, 2022.

Universal Robots. *UR5e User Manual*. Universal Robots A/S, 2024. URL `https://www.universal-robots.com/manuals/`.

J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5294–5306. IEEE/CVF, 2025a.

Q. Wang, V. Ye, H. Gao, W. Zeng, J. Austin, Z. Li, and A. Kanazawa. Shape of motion: 4d reconstruction from a single video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9660–9672. IEEE/CVF, 2025b.

Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 10510–10522, 2025c.

S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20697–20709. IEEE/CVF, 2024.

Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Permutation-equivariant visual geometry learning. In *International Conference on Learning Representations*, volume 2026, pp. 10481–10497, 2026.

Abraham Itzhak Weinberg, Alon Shirizly, Osher Azulay, and Avishai Sintov. Survey of learning-based approaches for robotic in-hand manipulation. *Frontiers in Robotics and AI*, 11:1455431, 2024.

Daniel E Whitney. Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on man-machine systems*, 10(2):47–53, 1969.

K. Wong, M. Stölzle, W. Xiao, C. Della Santina, D. Rus, and G. Zardini. Contact-aware safety in soft robots using high-order control barrier and lyapunov functions. arXiv preprint arXiv:2505.03841, 2025.

Wonik Robotics. Allegro hand. `http://wonikrobotics.com/robot-hand`, n. d. Accessed: 2024-06-17.

M. C. Yip and D. B. Camarillo. Model-less feedback control of continuum manipulators in constrained environments. *IEEE Transactions on Robotics*, 30(4):880–889, 2014.

Chuhan Zhang, Guillaume Le Moing, Skanda Koppula, Ignacio Rocco, Liliane Momeni, Junyu Xie, Shuyang Sun, Rahul Sukthankar, Joëlle K Barral, Raia Hadsell, et al. Efficiently reconstructing dynamic scenes one d4rt at a time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7382–7392, 2026.

J. Zhang, C. Herrmann, J. Hur, V. Jampani, T. Darrell, F. Cole, D. Sun, and M.-H. Yang. MonST3R: A simple approach for estimating geometry in the presence of motion. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

Dong Zhuo, Wenzhao Zheng, Jiahe Guo, Yuqi Wu, Jie Zhou, and Jiwen Lu. Streaming 4d visual geometry transformer. *arXiv preprint arXiv:2507.11539*, 2025.

A Appendix

A.1 Additional Loss

Depth Loss. To supervise 3D geometry, we fetch the depth z dimension of point cloud $\mathbf{X}_{t+1}$ and use reference depth z_{gt}:

$$\mathcal{L}_{depth} = \frac{1}{N}\sum_{i=1}^{N} \left\| \alpha \cdot z_i - z_{gt,i} \right\|^2, \alpha = \frac{\sum(z \cdot z_{gt})}{\sum z^2}. \tag{12}$$

Consistency Loss. To ensure temporal stability and maintain geometric consistency across the predicted states, we introduce a consistency loss $\mathcal{L}_{cons}$. Let $\mathbf{X'}_t$ and $\mathbf{X'}_{t+1}$ denote the predicted 3D coordinates of corresponding points at static points of both time steps. The consistency objective is defined as:

$$\mathcal{L}_{cons} = \sum_{n=1}^{N} \left\| \mathbf{X'}_t - \mathbf{X'}_{t+1} \right\|_2 \tag{13}$$

This loss ensures that the geometry remains invariant across the predicted transition, forcing the model to represent the robot's motion as a displacement of a single, coherent 3D structure rather than two independent reconstructions.

A.2 2D Pixel flow

Tab. 3 reports the 2D pixel error. All methods are evaluated on the same tracked pixels, and the error is the distance between the predicted and ground-truth endpoints, measured in pixels.

Under the 2D track metric, Act4R achieves the lowest error, and is the best method on five of seven robots. On HSA it is only slightly worse, which may come from the small displacement in that dataset. On Allegro Sim, Act4R$_{\text{NSA}}$ is better, and Act4R is nearly tied with NJF. The gap to NJF is large on the soft robots. Adding the 3D flow loss lowers the mean from 1.579 to 1.485, which indicates the 3D loss also improves 2D performance.

	Track Mean Pixel Error ↓				
Robot	Act4R	Act4R$_{\text{NSA}}$	Act4R$_{\text{N3D}}$	Act4R$_{\text{Gr00t}}$	NJF (Li et al., 2025)
Allegro	**1.959**	2.173	2.127	5.309	3.618
HSA	1.712	1.727	**1.710**	**1.710**	4.768
Toy Arm	**1.433**	1.490	1.492	1.629	2.683
Pneumatic	**1.502**	1.534	1.597	1.665	4.105
Allegro Sim	1.620	**1.487**	1.683	3.478	1.628
Franka Sim	**1.142**	1.890	1.354	2.504	1.950
UR5e Sim	**1.028**	1.098	1.093	2.260	1.388
Mean	**1.485**	1.628	1.579	2.651	2.877

Table 3: Performance comparison of track mean pixel error.

Fig. 6 shows the track prediction under different experimental settings.

A.3 Additional Action Metrics

Cosine similarity does not capture magnitude error. Tab. 4 shows the $\ell 2$ error metric. On HSA, Act4RGr00t is slightly lower, and on the pneumatic hand, Act4RNSA is a little bit better compared with Act4R. NJF is much worse on every robot. Replacing measured depth with VGGT depth raises the mean only from 0.121 to 0.123. This suggests estimated depth is a viable substitute if measured depth is unavailable.

We also recover actions from 3D scene flow using Eq. 9 and report the cosine similarity in Tab. 5. Act4R achieves the highest average similarity and performs best on six of the seven robots, while NJF

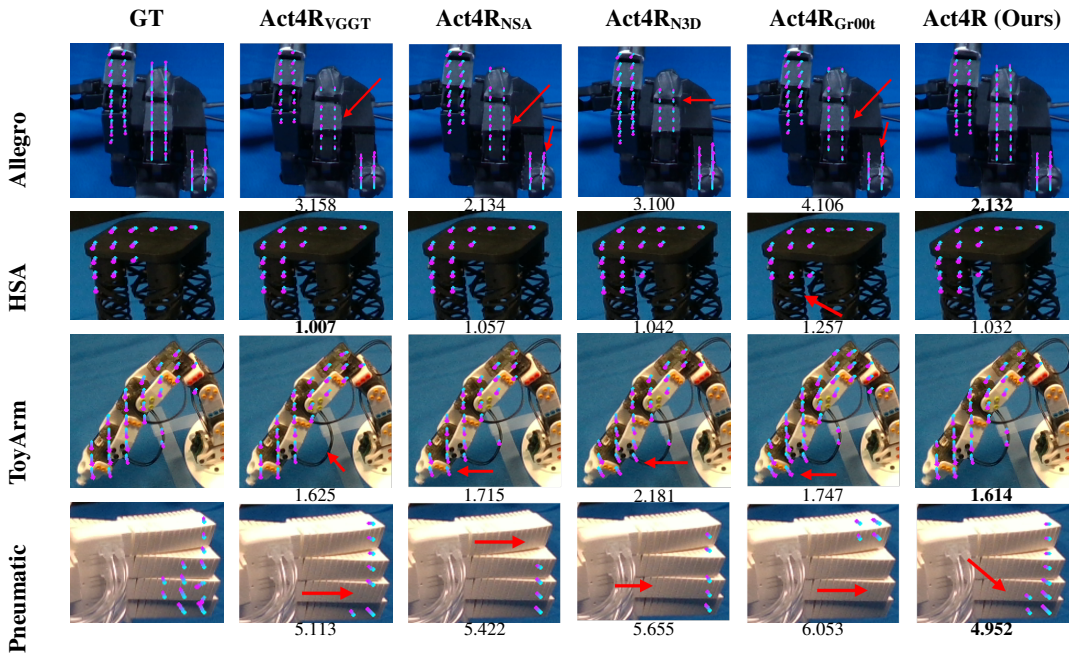


Figure 6: Visualization results for the ablation study. Values below each image indicate the Mean Pixel Error.

slightly outperforms it on HSA. Overall similarities are lower than those obtained from 2D tracks (Tab. 2), with the largest degradation observed on Allegro Sim. Replacing ground-truth depth with VGGT depth reduces the mean similarity from 0.446 to 0.377. Furthermore, removing the 3D flow loss causes Act4R$_{N3D}$ to drop sharply to 0.085, highlighting the importance of this loss when the target motion is specified in 3D.

	Action $\ell 2$ Error ↓					
Robot	Act4R	Act4R$_{VGGT}$	Act4R$_{NSA}$	Act4R$_{N3D}$	Act4R$_{Gr00t}$	NJF (Li et al., 2025)
Allegro	**0.129**	0.134	0.136	0.135	0.167	0.900
HSA	0.158	0.159	0.162	0.158	**0.155**	1.231
Toy Arm	**0.162**	**0.162**	0.163	0.163	0.166	0.965
Pneumatic	0.229	0.229	**0.228**	0.231	0.234	1.220
Allegro Sim	**0.069**	0.070	0.278	0.073	0.261	0.644
Franka Sim	**0.064**	0.066	0.068	0.067	0.075	0.485
UR5e Sim	**0.037**	0.040	0.051	0.045	0.041	0.093
Mean	**0.121**	0.123	0.155	0.125	0.157	0.791

Table 4: Performance comparison of action $\ell 2$ error.

	Action Cosine Similarity 3D ↑			
Robot	Act4R	Act4R$_{\text{VGGT}}$	Act4R$_{\text{N3D}}$	NJF (Li et al., 2025)
Allegro	**0.665**	0.573	0.104	0.523
HSA	0.489	0.427	0.040	**0.503**
Toy Arm	**0.570**	0.470	0.230	0.354
Pneumatic	**0.302**	0.268	0.067	0.130
Allegro Sim	**0.149**	0.115	0.060	0.072
Franka Sim	**0.396**	0.301	0.036	0.063
UR5e Sim	**0.551**	0.483	0.061	0.191
Mean	**0.446**	0.377	0.085	0.262

Table 5: Performance comparison of 3D action cosine similarity.