# AGENTIC STAR: SHAPE TRACKING AND RECONSTRUCTION FROM MONOCULAR VIDEOS

**Kirill Mazur, Nikita Karaev, Matthew Chang,**
**Jitendra Malik, Nur Muhammad "Mahi" Shafiullah**

Amazon FAR (Frontier AI and Robotics)
{makezur,nikaraev}@amazon.co.uk
{drmchang,jtnmalik,notmahi}@amazon.com

https://agenticstar.github.io

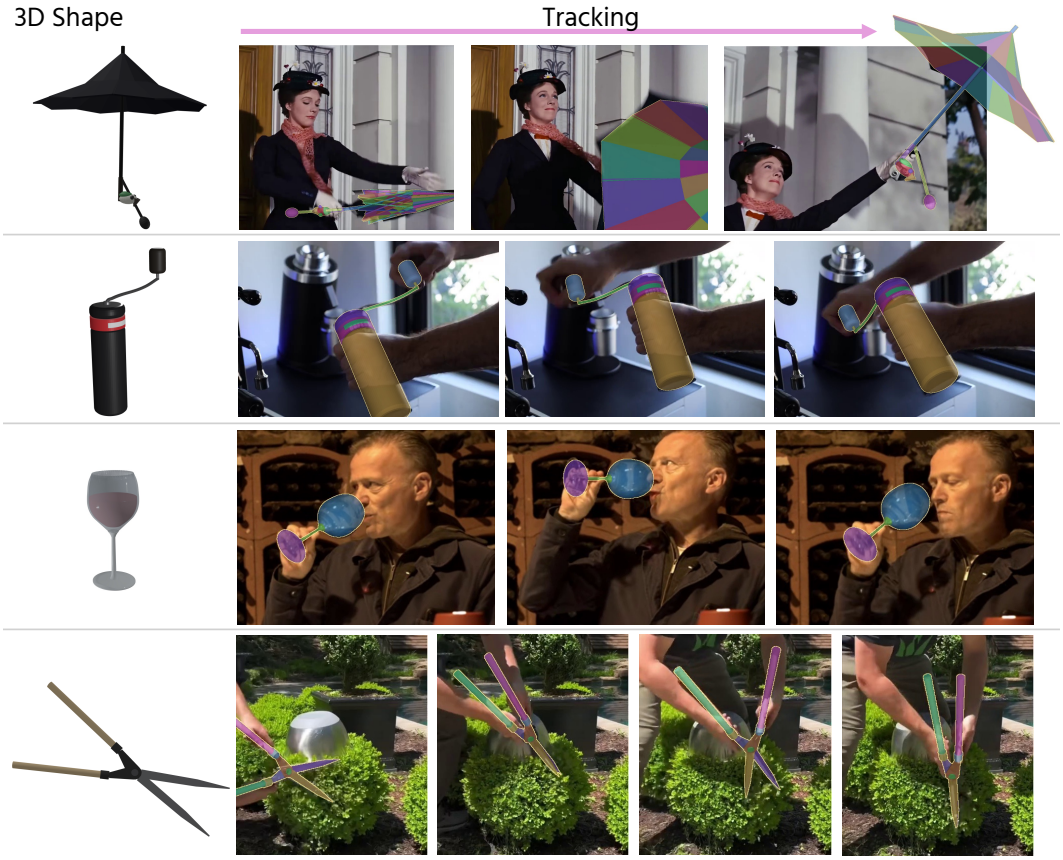


Figure 1: **Agentic shape tracking and reconstruction.** Our method jointly reconstructs object geometry and kinematics (**left**) from monocular videos and tracks the resulting 3D model through time (**right**). It recovers complex articulation (first and fourth rows), preserves fine geometric details (second row), and tracks transparent objects (third row). Consistent part colours across frames visualise temporal correspondences and tracking quality.

## ABSTRACT

The problem of reconstructing and tracking arbitrary objects from monocular videos remains a challenging one – primarily due to the wide variety in solid and articulated, textured and textureless objects in the wild. In this work, we present a method for shape reconstruction and tracking from video via agentic analysis-by-synthesis. Unlike prior methods which first estimate dense pixel correspondences and then recover object motion from them, our method infers a structured 3D object model, including its geometry and kinematic structure, and uses this model to optimise object track estimates over time. In our optimisation loop, a Vision-Language Model (VLM) agent iteratively refines shape or generalised pose

through a render-and-compare loop, combining coarse visual reasoning with numerical pose optimisation for precise state estimation. This structured formulation enables our method to track through large motion, articulation, and severe occlusion without relying on pixel-matching objectives. Quantitatively, on ARCTIC, our method substantially outperforms state-of-the-art 3D point-tracking baselines for articulated objects, and on HOT3D it outperforms all evaluated rigid-object tracking baselines.

## 1 INTRODUCTION

Bottom-up versus top-down debate in perception has a long history in computer vision. In dynamic reconstruction, the question can be put in the following way: should we first recover low-level visual evidence, such as scene flow or point trajectories, and then infer the object that generated it. Or should we first recognise the object and its structure, and use this structured representation to recover its state over time?

Most previous work on articulated object reconstruction (Liu et al., 2023b;a; Zhao et al., 2025; Peng et al., 2026; Delitzas et al., 2026) approaches the problem in a bottom up way, building on general scene-flow and correspondence methods. These methods reconstruct the 3D scene and estimate either 2D correspondence (Teed & Deng, 2020; Karaev et al., 2024; Harley et al., 2025) or 3D point trajectories (Zhang et al., 2026a; Sucar et al., 2026; Feng* et al., 2025; Xiao et al., 2025) over time. While this provides a powerful general-purpose representation of observed motion, it comes with two key limitations. First, dense correspondence estimation and tracking are themselves challenging under the occlusions and limited visual overlap typical to dynamic object interactions. Second, the resulting representation remains largely unstructured: point clouds and trajectories describe where visual evidence moves, but not the underlying objects, their parts, joints, or state variables. Such structure must therefore be inferred only after reconstruction, which becomes brittle when the recovered geometry and trajectories are imperfect. This is particularly limiting for downstream applications such as robotics, which require explicit object models that can be instantiated and manipulated in simulation.

Rather than first asking where every observed point moved, we directly ask which *structured 3D object and state sequence could have generated the video*. We therefore adopt an analysis-by-synthesis approach for dynamic object reconstruction and tracking. Our representation models the object's geometry, kinematic structure, and generalised pose over time, comprising its 6-DOF base pose and kinematic states.

Once the object's structure is known, its permissible configurations are strongly constrained, transforming dense point-wise motion estimation into a compact pose-estimation problem over base pose and articulation. The resulting states are also interpretable and semantically meaningful.

Realising this top-down approach requires a model that can infer object structure and then use that structure to inform pose estimation. Much of an object's motion is determined not by visual evidence alone, but by its internal mechanism. For example, when a person closes a book, its pages become occluded and can no longer be visually tracked, yet their possible motion remains strongly constrained by the structure of the book. To exploit knowledge of the object's mechanism, tracking should be done by the same entity that inferred the shape.

Large vision-language models (VLMs), which serve as the backbone of modern coding agents, now exhibit a high level of generalisation and are extremely good at providing coarse 3D shape estimates (Yin et al., 2026; Zhou et al., 2026) or detecting gross spatial inconsistencies, making them natural candidates for this task. On their own, however, they are poor state estimators: a VLM can reliably judge whether an object is *roughly* posed correctly (e.g., that it is upside down), but cannot recover its pose to within a few degrees. Traditional test-time optimisation methods such as Structure-From-Motion (SfM), on the other hand, are often precise but brittle.

We show how to pair VLM agents with numerical optimisation tools to form an *agentic optimisation loop* for joint shape and state estimation, iteratively recovering the shape, joints, and pose of dynamic objects. The key difference from existing work is that the VLM makes gradient-free optimisation tractable, by narrowing the search to a promising optimisation basin and rejecting spurious local minima, which is often the hardest part of the problem. Once the correct basin is identified, the

compact parameter space of shape parameters, joint axes and their configurations makes refinement tractable.

To the best of our knowledge, ours is the first work to apply VLM agents to joint shape and motion reconstruction of both rigid and articulated objects from casually captured monocular videos. We propose a novel agentic render-and-compare loop, and demonstrate that our method outperforms 3D point tracking baselines for articulated object reconstruction on ARCTIC (Fan et al., 2023) and existing articulated object reconstruction methods on iTACO (Peng et al., 2026), as well as rigid object tracking systems on the challenging HOT3D (Banerjee et al., 2025) dataset. Crucially, because we model the cause of object motion rather than its visual evidence, the proposed method can reconstruct and recover motion in cases that are beyond the reach of prior methods — for example, tracking transparent objects such as glass, as well as severely occluded or only partially visible objects.

## 2 Related Work

**3D as Code.** DeepSVG (Carlier et al., 2020) modelled vector graphics as sequences of drawing commands, while DeepCAD (Wu et al., 2021) generated CAD construction sequences. Similarly, PolyGen (Nash et al., 2020) and MeshGPT (Siddiqui et al., 2024) represented polygonal meshes as discrete sequences for autoregressive generation. SceneScript (Avetisyan et al., 2024) brought this paradigm to perception, predicting a structured language of 3D scene primitives from visual observations and SfM information. More recently, LLMs, VLMs, and agentic systems have been incorporated into code-based 3D generation and inverse graphics pipelines to procedurally generate and reconstruct 3D scenes (Kulits et al., 2024; Gu et al., 2025; Sun et al., 2025; Yin et al., 2026).

Code-based representations are a natural fit for articulated objects, whose geometry and kinematics form arbitrary hierarchical structures. Starting with Real2Code (Zhao et al., 2025), subsequent systems (Le et al., 2024; Zhou et al., 2026) have used code-based representations to reconstruct or generate articulated models. These approaches primarily address model acquisition or asset generation, rather than reconstruction and tracking through video.

**Articulated Object Reconstruction.** Early work recovered articulated structure by factorising tracked trajectories into rigidly moving parts and their kinematic relations (Costeira & Kanade, 1998; Yan & Pollefeys, 2008; Tresadern & Reid, 2005). Modern bottom-up methods recover part geometry and motion from multi-view consistency, neural fields, or dense correspondences (Liu et al., 2023a;b; Deng et al., 2024; Kerr et al., 2024; Mazur et al., 2026; Delitzas et al., 2026).

Learning-based methods directly predict articulated geometry or kinematics from static or sparse observations (Jiang et al., 2022; Heppert et al., 2023; Li et al., 2026b;a). More recent approaches strengthen these priors using retrieval, generative models, and VLMs (Chen et al., 2024; Le et al., 2024; Liu et al., 2025; Zhao et al., 2025; Li et al., 2025). While these methods can infer plausible kinematic structure from limited observations, they generally reconstruct an articulated model rather than track its articulation through a video.

**Object and Point Tracking.** Dynamic tracking methods differ in the structure they represent: some estimate point-level tracks, while others impose object-level structure, such as rigid motion.

Point trackers estimate 2D (Doersch et al., 2022; 2023; Karaev et al., 2024; Harley et al., 2025) or 3D trajectories (Koppula et al., 2024; Xiao et al., 2024; 2025; Sucar et al., 2026; Zhang et al., 2026a) of observed pixels over time. While general, these predictions lack explicit object structure and often require post-processing for downstream applications.

Rigid object trackers instead assume a shared 6-DoF motion and can be divided into model-based and model-free methods. Model-based approaches such as FoundationPose (Wen et al., 2024) assume an object model. Recent methods also use models obtained through 3D generation (Nguyen et al., 2024; Lee et al., 2025), including the adaptation of SAM3D (Chen et al., 2025) for monocular object tracking (Paliwal et al., 2026). Model-free approaches (Sun et al., 2022; Wen et al., 2023; Taher et al., 2026) instead recover structure from observations, framing tracking as an object-centric multi-view geometry problem.

## 3 METHOD

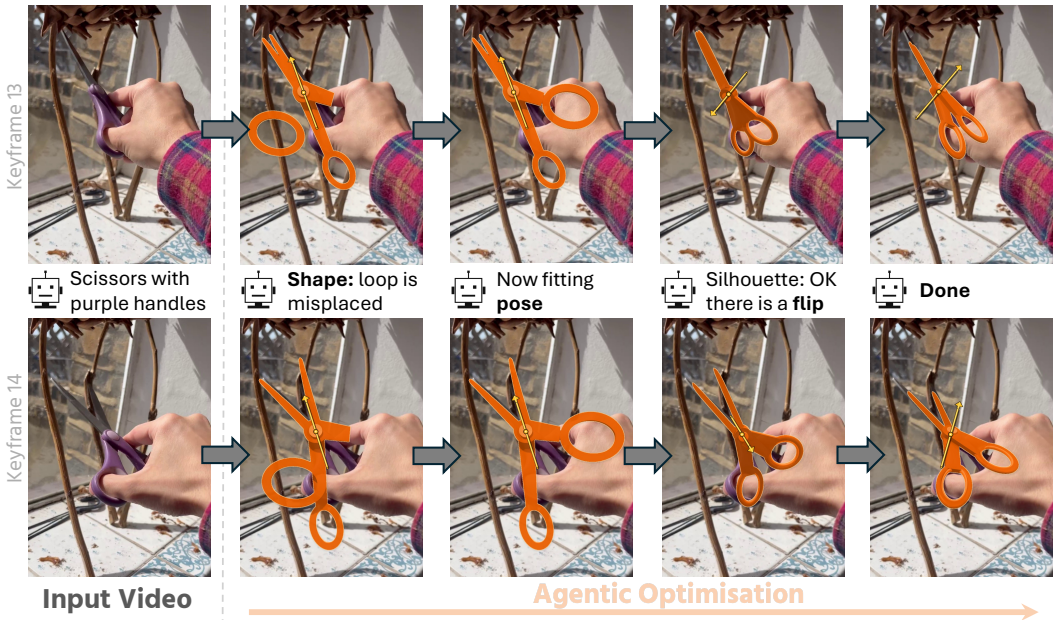


Figure 2: **Method**. Given an input video **(left)**, our agentic optimisation loop iteratively refines the shape and pose of the observed object **(right)**. The middle panels summarize the agent's reasoning throughout the optimisation process. Our harness provides the VLM with numerical optimisation tools for precise pose fitting, together with temporal diagnostics for assessing consistency across keyframes. The final output is a reconstructed object model together with its estimated generalised pose at every observed keyframe.

**Task formulation.** Our method takes as input a set of observed video frames $\{I_i \in \mathbb{R}^{H\times W\times 3}\}$, *optionally* coupled with depth maps $\{D_i \in \mathbb{R}^{H\times W}\}$, and binary masks $\{M_i \in \{0,1\}^{H\times W}\}$ for the target object. Our method can also take in hand segmentation masks $\{H_i \in \{0,1\}^{H\times W}\}$, as they often occlude the object of interest.

The masks are typically extracted by a video-segmentation model, such as SAM3 (Carion et al., 2025), or provided directly.

**Goal.** Our goal is to build a canonical object model $\mathcal{O}$, represented as code and shared across all observed frames, together with its poses and kinematic parameters $\mathcal{T}_i$ for every observed video frame $I_i$. We use an object-to-camera convention for poses and forward kinematics, so a generalised pose should map the object to its coordinates in the target camera.

**Conventions.** We assume a pinhole camera model with known calibration $K_i$ and known camera pose extrinsics $(R_i^{\text{cam}}, t_i^{\text{cam}})$ estimated by an external SLAM system or a feed-forward reconstruction system (Wang et al., 2025; 2026).

Given object model $\mathcal{O}$ and its pose $\mathcal{T}_i$ we denote its render into the camera $i$ as $\mathcal{R}(\mathcal{O}, \mathcal{T}_i)$. Its silhouette render is denoted as $\mathcal{R}_s(\mathcal{O}, \mathcal{T}_i)$

### 3.1 OVERVIEW

Recent large models have been trained on a vast amount of data and exhibit a high level of semantic knowledge and recognition capability, including the ability to recognise shape, kinematic structure, and approximate object orientation (Zhou et al., 2026). However, these models struggle with precise quantity estimation. Estimating continuous quantities, such as object pose, has always been the strength of iterative, optimisation-based methods, such as bundle adjustment (Triggs et al., 1999).

Guided by these observations, we design an *agentic render-and-compare* optimisation loop that allows VLMs to iteratively refine both the shape and pose of the object. Akin to regular optimisation methods, our core loop is iterative. Each iteration is allowed to either modify the shape or pose only, never jointly. The exact scheduling decision is left to an agent.

**Shape step.** Shape and kinematics editing is done by a coding agent, conditioned on the past observation history, such as observations made during pose steps. All shape coding is confined to a single scene.py script, where geometry is freeform and can represent arbitrary topologies, whereas pose definition should follow the representation conventions: all pose-related fields are stored under predefined variable names in the script, so they can be automatically decoupled from the shape definition and updated independently. Diagnostic rendering tools allow the agent to inspect the resulting object from both observed and novel viewpoints.

**Pose step.** For a pose iteration, the object model $\mathcal{O}$ is frozen and rendered under its current pose estimates $\{\mathcal{T}_i\}$ at the target keyframes. Rather than directly predicting precise continuous pose updates, the agent specifies interpretable directions or search regions of the pose space to explore. Our pose optimisation tool (3.4) searches these regions numerically, renders and scores candidate poses. The top scoring candidates are then returned back to the agent for visual inspection. Pose refinement proceeds iteratively: the agent invokes the pose optimisation tool, visually inspects the returned candidates, and uses the results to specify new search directions for subsequent iterations. Since the shape is shared and fixed during these iterations, pose refinement can be parallelised across the sequence.

For video, the agent has to run a mandatory temporal diagnostic tool (3.5) that identifies implausible discontinuities in the estimated trajectory. This allows temporal smoothing to be applied selectively rather than imposed as a fixed prior.

**Tool List.** We list major utils implemented in our harness and omit infrastructure related tools:

- Blender parallel rendering utils coupled with rendering visualisation scripts;
- Silhouette scoring, see section 3.2;
- Pose tool, see section 3.4;
- Temporal residuals tool, see section 3.5;
- Turnable: a utility to render views to a sphere around the object.
- External VLM critic invocation for shape quality inspection;

### 3.2 Scoring Objective

Inspired by the shape-from-silhouette (Szeliski, 1993) line of work, we set Intersection-over-Union (IoU) of our renders' silhouettes $\mathcal{R}_s(\mathcal{O}, \mathcal{T})$ with the target object mask $M_i$ as the main numerical score $\mathcal{S}(\mathcal{O}, \mathcal{T})$ for the agent. Although IoU is an inherently local signal and is prone to many local minima, coupling it with a VLM's ability to reason "approximately" helps the agent escape these minima. We focus on dynamic objects and assume humans are manipulating them. We extract hand segmentation masks $H_i$ and exclude them from IoU, as hands often severely occlude the object of interest.

Formally,

$$\mathcal{S}(\mathcal{O}, \mathcal{T}) \; : \; = \mathrm{IoU}(\mathcal{R}_s(\mathcal{O}, \mathcal{T}) \setminus H_i; M_i \setminus H_i) \tag{1}$$

Incorporating other signals, such as pixel matching (Teed & Deng, 2020; Leroy et al., 2024; Edstedt et al., 2024), might also be viable, although the agentic optimisation loop should then reason about the failure cases of the possibly noisy signal. We therefore do not use such signals.

We also provide a variant of our system where additional supervision (score term with a blending coefficient $\alpha$) comes from depth signal. Unless stated otherwise, our method does *not* employ depth supervision.

### 3.3 OBJECT AND STATE REPRESENTATION

**Shape and Geometry.** Our shape and its kinematic structure $\mathcal{O}$ are represented in the form of Python code. Geometry is expressed via Blender shape primitives. The joint kinematic structure, its limits, and the joint positions are also expressed this way, which allows modelling arbitrary kinematic structures and permits natural updates to the shape or state space of an object of interest in the form of code diffs.

**Pose Representation.** For an articulated object, we define the *generalised pose* $\mathcal{T}_i$ as the 6-DoF pose of its base together with the states of all articulation joints:

$$\mathcal{T}_i = (T_i, j_{i,1}, \ldots, j_{i,n}), \qquad T_i = (R_i, t_i) \in \mathbb{SE}(3), \qquad j_{i,k} \in \mathbb{R}^1. \tag{2}$$

Each joint state $j_{i,k} \in \mathbb{R}^1$ is constrained by its corresponding motion limits. Throughout the paper, we use *pose* to refer to this generalised pose unless stated otherwise.

The object has a single scale $s$ shared across all frames, and we estimate object-to-camera poses. We store rotations as unit quaternions and translations as vectors. Given a canonical-frame point $\boldsymbol{p}_\mathrm{c}$ after forward kinematics, its position in camera $i$ is

$$\boldsymbol{p}_i = sR_i\boldsymbol{p}_\mathrm{c} + t_i. \tag{3}$$

**Pose Updates.** To make rotational corrections interpretable to the VLM, we explicitly distinguish between camera-centric and object-centric updates. Because rotations do not commute, left- and right-multiplication correspond to rotations expressed in different coordinate frames. Camera-centric updates left-multiply the current rotation, whereas object-centric updates right-multiply it, which is particularly useful for expressing large transformations such as symmetry flips. Assuming for clarity that the object centre after forward kinematics is at the canonical origin, the two update conventions are:

Camera-centric updates:

$$\boldsymbol{p}_i^{\mathrm{cam}} = s \cdot (\Delta R)\, R_i\, \boldsymbol{p}_\mathrm{c} + t_i + \Delta t \tag{4}$$

Object-centric updates:

$$\boldsymbol{p}_i^{\mathrm{cam}} = s \cdot R_i\, (\Delta R)\, \boldsymbol{p}_\mathrm{c} + t_i \tag{5}$$

### 3.4 VLM-GUIDED POSE OPTIMISATION

Classical pose optimisation methods can yield precise estimates, but our scoring objective $\mathcal{S}(\mathcal{O}, \mathcal{T})$ provides an inherently local signal and is therefore sensitive to initialisation and prone to spurious local optima. In contrast, modern VLM agents are effective at coarse visual reasoning, but less suited to precise continuous estimation. We combine these complementary strengths in a *VLM-guided pose optimisation loop*, where the VLM identifies a promising search region and a numerical optimiser performs precise refinement within it.

Given the current pose $\mathcal{T}$, the agent specifies a bounded search region, e.g. "vary yaw from $-15°$ to $15°$ and the hinge joint from $24°$ to $64°$." A gradient-free optimiser (Storn & Price, 1997) is then executed within this region, producing the $K$ highest-scoring pose candidates $\{\mathcal{T}^i_{\mathrm{order}}\}_{i=1}^K$. Their corresponding renders and numerical scores,

$$\left\{\mathcal{R}(\mathcal{O}, \mathcal{T}^i_{\mathrm{order}}), \mathcal{S}(\mathcal{O}, \mathcal{T}^i_{\mathrm{order}})\right\}_{i=1}^K,$$

are returned to the agent for visual inspection. The final selection is made by the VLM, which chooses the *visually* best candidate among these high-scoring solutions. Consequently, the selected pose need not be the numerical optimum under $\mathcal{S}$. The selected candidate becomes the starting pose for the next optimisation iteration.

**Pose Search Space.** The agent specifies bounded search intervals over interpretable pose variables, including yaw, pitch, roll, image-plane translation, depth, and articulation states. We describe the pose-update parameterisation and transformation conventions in section 3.3.

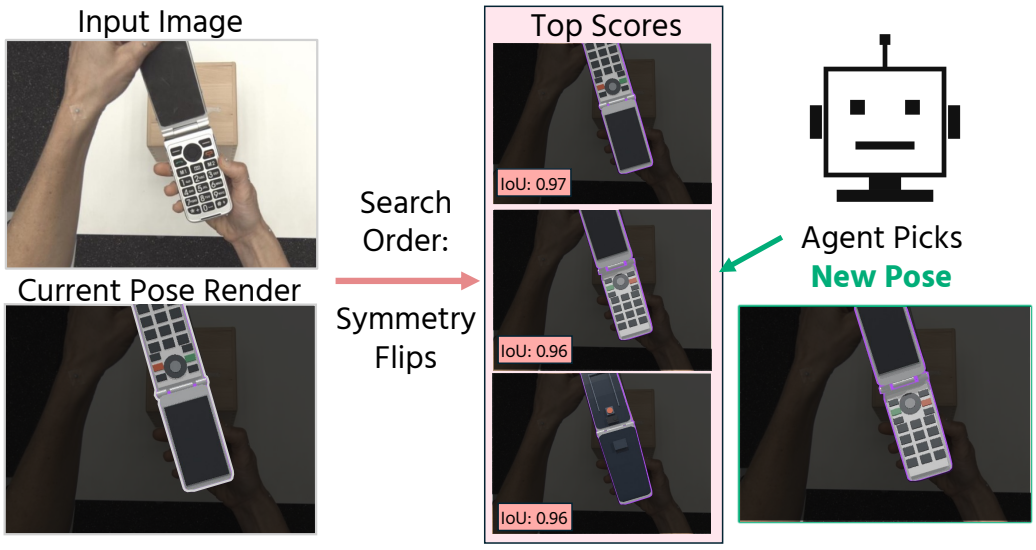


Figure 3: **VLM-guided pose optimisation.** Given the current pose estimate, a VLM agent defines a bounded search subspace, such as symmetry flips around a chosen axis. A numerical optimiser searches this subspace and returns the highest-scoring pose candidates. Because the silhouette-based objective is prone to spurious local optima, the VLM visually inspects the top candidates and selects the best pose, which becomes the starting point for the next optimisation iteration.

### 3.5 Sequence-level Pose Estimation

When fitting an object model to a video, estimating poses independently across frames can produce temporally inconsistent trajectories. For example, for symmetric objects, adjacent frames may converge to different symmetry-equivalent poses, resulting in discontinuous 3D point trajectories. Classical state-estimation methods (Dellaert & Kaess, 2017) often address this with temporal smoothness factors between consecutive estimates. However, a fixed smoothness prior can suppress genuine rapid motion, which is common in real-world videos. We therefore use *agentic smoothing*: temporal inconsistencies are reported to the agent as diagnostics, while the agent can retain rapid motion when it is supported by the visual observations.

**Temporal Residuals.** Given a sequence of generalised poses $\{\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_n\}$, we compute temporal residuals for the object's 6-DoF base pose and its joints. Specifically, we measure velocities and accelerations component-wise rather than defining a single metric over the full configuration space. For one-dimensional joints, these quantities are given by first- and second-order differences.

For the monocular setting with known camera poses, the dynamic object retains an independent scale gauge. We therefore reparameterise its translation as $t_i = s\tau_i$:

$$\boldsymbol{p}_i^{world} = s(R_i^{\text{cam}} R_i)\boldsymbol{p} + (s\, R_i^{\text{cam}} \tau_i + t_i^{\text{cam}})\,. \tag{6}$$

For temporal residuals, we compare the camera-motion-compensated rotation $R_i^{\text{cam}} R_i$ and the scale-normalised translation $R_i^{\text{cam}} \tau_i$, expressed in object units. We omit $t_i^{\text{cam}}$ because the camera trajectory's translation gauge need not be compatible with the scale of the dynamic object reconstruction.

## 4 Experiments

### 4.1 Experimental Details

Unless stated otherwise, we use the Codex harness with our tool suite and GPT-5.6 Sol at medium reasoning effort. We employed the same set of tooling and prompts for all experiments, unless stated otherwise. We budget 10 hours for each optimisation loop, however some agents can report termination earlier.

### 4.2 ARTICULATED OBJECT RECONSTRUCTION: ARCTIC

Table 1: **Performance on ARCTIC. (Left)** 3D tracking quality compared with state-of-the-art 3D point trackers: V-DPM (Sucar et al., 2026), OpenD4RT (Zhang et al., 2026a), and SpatialTrackerV2 (Xiao et al., 2025). We evaluate points queried in the first keyframe and track their 3D trajectories across all subsequent keyframes. **(Right)** Geometry reconstruction quality compared with V-DPM, the best-performing tracking baseline in the left panel. Errors are reported in cm.

| Metric | SpatialTrackerV2 | OpenD4RT | V-DPM | Ours |
|---|---|---|---|---|
| 3D EPE (cm) ↓ | 10.79 | 8.77 | 7.65 | **5.59** |

**(a) 3D Tracking**

| Metric | V-DPM | Ours |
|---|---|---|
| Chamfer (cm) ↓ | 4.72 | **3.36** |

**(b) Geometry**

ARCTIC (Fan et al., 2023) contains humans interacting with articulated objects undergoing substantial articulation and rapid motion. It provides high-quality ground-truth geometry and motion captured using a motion-capture rig.

For evaluation, we use the egocentric-camera sequences of subject $s1$, yielding $24$ sequences in total. We discard the first $100$ frames of each sequence, which typically correspond to capture setup, contain little or no object motion, and often include severely underexposed frames. We divide the remaining frames into chunks of $300$ frames and select every $10$th frame as a keyframe. Reconstruction and tracking are performed on these keyframes, and all methods are evaluated on the same set of keyframes.

The predicted geometry and trajectories of all methods are recovered only up to an unknown global scale. We resolve this scale through depth alignment; see section A.4 for details.

**3D Point Tracking.** Because these sequences are too challenging for existing articulated-object reconstruction methods, we compare against state-of-the-art 3D point-tracking methods. Most baselines are pixel-anchored: given a query pixel $q$ in the first keyframe $I_0$, they estimate the 3D trajectory of the corresponding scene point over time. Ground-truth trajectories are obtained from the ground-truth articulated mesh poses. We evaluate 3D End-Point Error (EPE) (Liu et al., 2019) for query pixels lying inside the ground-truth object mask in the first keyframe.

Table 2: **Harness ablation.** 3D EPE for point tracking.

| Variant | EPE (cm) ↓ |
|---|---|
| No Harness | 11.26 |
| IoU Objective | 151.46 |
| GT mesh + VLM-free opt. | 14.60 |
| No temporal | 6.15 |
| Ours (GPT 5.6 Sol) | 5.59 |
| Ours (Fable 5) | **4.95** |

Our representation, in contrast, is a structured 3D mesh and does not necessarily contain a predicted point corresponding to every query pixel in the ground-truth mask. We therefore establish correspondences between query pixels and points on the predicted mesh. In the first keyframe, each query pixel is matched to the nearest projected mesh point. The matched point is then tracked through the sequence and used to compute 3D EPE.

**Geometry Quality.** We report Chamfer distance between the predicted and ground-truth geometry, averaged across all keyframes. For our method, this is computed between the predicted and ground-truth meshes. For V-DPM, we instead compute Chamfer distance between its predicted 3D point cloud at each timestamp and the ground-truth mesh.

### 4.3 RIGID OBJECT TRACKING: HOT3D

**Protocol.** Our method also applies to model-free rigid-object 6-DoF pose estimation from monocular video. We evaluate on the challenging HOT3D (Banerjee et al., 2025) dataset, which provides high-quality motion-capture ground-truth poses and contains rapid object motion. We use the validation split and retain sequences containing a single dynamic target object, resulting in $93$ sequences total. Each sequence contains $150$ frames. Our method operates on every $10$th frame, yielding $15$ keyframes per sequence. All methods are evaluated on exactly these keyframes. Video-tracking baselines are additionally allowed to process all $150$ frames, including the intermediate frames, but their metrics are computed only at the selected keyframes.

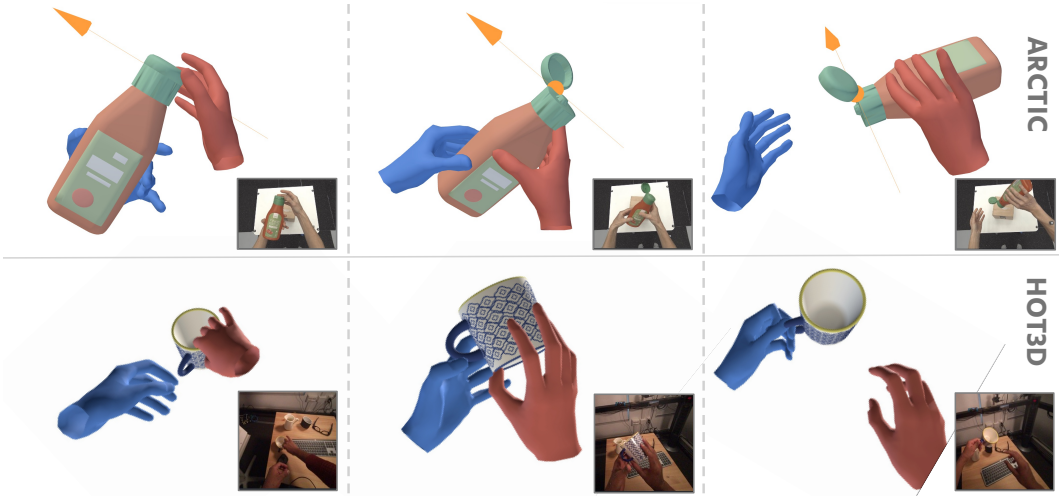


Figure 4: **Reconstructions on ARCTIC and HOT3D.** We show our reconstructions at three different timestamps alongside ground-truth hands, demonstrating the quality of our 3D pose alignment in the world frame. **(Top)** Our method recovers fine articulations, such as the ketchup-bottle cap, while tracking large and rapid pose changes. **(Bottom)** Our method accurately tracks objects with fine geometric details and repetitive textures.

Table 3: **Pose Tracking on HOT3D.** We report per-trajectory mean and median translation and rotation errors, averaged across trajectories. Methods that use the ground-truth CAD model are marked with *. Our method outperforms all evaluated baselines, including those with access to the ground-truth CAD model. Translation errors for ProxyPose are omitted because degenerate trajectories make the required alignment unstable.

| **Method** | **Trans. Err. (cm)** ↓ | | **Rot. Err. (°)** ↓ | |
|---|---|---|---|---|
| | **Mean** | **Median** | **Mean** | **Median** |
| ProxyPose (Zhang et al., 2026b) | – | – | 52.7 | 43.4 |
| GigaPose (Nguyen et al., 2024)* | 11.25 | 8.16 | 62.8 | 50.9 |
| GigaPose* + GoTrack (Nguyen et al., 2025) | 16.46 | 9.92 | 55.4 | 37.9 |
| FoundationPose (Wen et al., 2024)* + VGGT-Ω | 7.56 | 5.57 | 54.6 | 49.1 |
| SAM3D-Tracker (Paliwal et al., 2026) | 6.41 | 5.30 | 51.6 | 43.0 |
| Ours | **3.04** | **2.42** | **37.6** | **26.3** |

**Metrics.** Our goal is to evaluate temporally consistent object tracking rather than frame-wise pose estimation. We therefore do not independently quotient out object symmetries at each frame. In particular, a prediction that switches between symmetry-equivalent poses over time is considered incorrect, since such a switch can induce a different 3D point trajectory.

For model-free methods, the predicted canonical object frame is arbitrary, giving rise to a global $\mathrm{Sim}(3)$ gauge ambiguity. We resolve this ambiguity by estimating a single global rotation and translation between the predicted and ground-truth trajectories, which are then fixed for the entire sequence. For methods that also reconstruct an object (ours and SAM3D-Tracker), we estimate a single global scale factor separately by aligning rendered object depth with ground-truth depth. The same scale-alignment procedure is applied to all methods without metric-depth input. Further details are provided in the appendix.

After alignment, we compute per-frame translation and rotation errors. We compute the mean and median error within each trajectory, and report their averages across the evaluation set.

**Results.** As shown in table 3, our method outperforms all evaluated baselines in both translation and rotation error. Rotation errors remain relatively high for all methods, reflecting the particularly

rapid motion in HOT3D: object orientation can change by as much as $180^\circ$ between consecutive evaluation keyframes. This regime is substantially more challenging than rigid-object tracking settings dominated by smooth inter-frame motion.

### 4.4 Method Study

A natural first question is how much of the final performance comes from our agentic optimisation rather than from the underlying VLM alone. We therefore ablate the main components of our harness in table 2. A pure agent given the same inputs, task description, and output format, but without our harness (**No Harness**), performs substantially worse than the full system.

Providing the agent only with our numerical IoU score (**IoU Objective**) tooling is not sufficient either: instead, the agent frequently exploits the objective by producing flat, silhouette-like geometry that achieves high IoU without faithfully reconstructing the object, resulting in extremely high reconstruction and tracking errors.

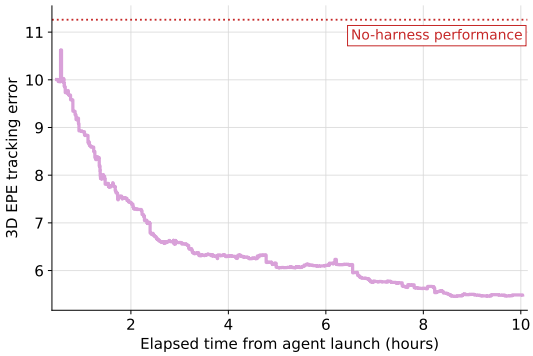


Figure 5: **Agentic optimisation timelapse on ARCTIC.** Tracking error decreases over the course of optimisation. The dashed line denotes the no-harness baseline.

We next isolate two components of the optimisation procedure. To evaluate the importance of VLM-guided pose search, we replace it with differential evolution over the full pose space while providing ground-truth geometry (**GT mesh + VLM-free opt.**). Despite this substantial advantage, the variant performs considerably worse than our full method, highlighting the importance of VLM guidance in identifying useful pose-search regions. Removing the temporal reasoning tool (**No temporal**) also degrades 3D point tracking, demonstrating the benefit of sequence-level reasoning.

**Optimisation dynamics and robustness.** In fig. 5, tracking error decreases steadily over optimisation, showing convergence behaviour similar to conventional iterative optimisation. As agentic pipelines are typically stochastic, we also report standard deviation of our tracking 3D EPE on the whole dataset obtained over 3 independent runs: 0.06cm. Lastly, we swap the underlying agent model to Claude Fable 5 with the same tools and Claude Code harness (**Fable 5**).

### 4.5 iTACO

Table 4: **Results on the iTACO benchmark.** We evaluate our method on the simulated RGB-D iTACO benchmark against Articulate-Anything (Le et al., 2024), Robot See Robot Do (Kerr et al., 2024), and iTACO (Peng et al., 2026). Following iTACO, we report mean $\pm$ standard deviation for each metric. Our method performs competitively on geometry while substantially outperforming the baselines on kinematic estimation.

| | Metric | Articulate-Anything | Robot See Robot Do | iTACO | **Ours** |
|---|---|---|---|---|---|
| Geometry | CD-w ($m^2$) $\downarrow$ | $0.11 \pm 0.22$ | $3.39 \pm 21.50$ | $\mathbf{0.01 \pm 0.01}$ | $0.02 \pm 0.03$ |
| | CD-m ($m^2$) $\downarrow$ | $0.59 \pm 0.73$ | $0.60 \pm 0.60$ | $0.13 \pm 0.26$ | $\mathbf{0.02 \pm 0.06}$ |
| | CD-s ($m^2$) $\downarrow$ | $0.07 \pm 0.18$ | $0.17 \pm 0.44$ | $0.06 \pm 0.19$ | $\mathbf{0.02 \pm 0.04}$ |
| Revolute | Axis (rad) $\downarrow$ | $0.82 \pm 0.79$ | $1.16 \pm 0.52$ | $0.32 \pm 0.56$ | $\mathbf{0.08 \pm 0.11}$ |
| | Position (m) $\downarrow$ | $0.81 \pm 0.40$ | $1.18 \pm 1.21$ | $0.13 \pm 0.25$ | $\mathbf{0.05 \pm 0.06}$ |
| | State (rad) $\downarrow$ | N/A | $1.05 \pm 0.59$ | $0.25 \pm 0.46$ | $\mathbf{0.15 \pm 0.26}$ |
| Prismatic | Axis (rad) $\downarrow$ | $0.92 \pm 0.78$ | $1.27 \pm 0.44$ | $0.24 \pm 0.33$ | $\mathbf{0.07 \pm 0.09}$ |
| | State (m) $\downarrow$ | N/A | $0.63 \pm 0.41$ | $0.08 \pm 0.22$ | $\mathbf{0.04 \pm 0.03}$ |

Following iTACO (Peng et al., 2026), we evaluate our method on its simulated RGB-D benchmark using the same keyframes and evaluation protocol. For this experiment, we use the RGB-D variant of our method and augment the scoring objective in equation 1 with an $l_1$ depth term. After optimisation, we scale-align the reconstructed meshes to the input depth maps.

As shown in table 4, our method substantially outperforms existing baselines on kinematic estimation, while remaining competitive on geometry. In particular, we achieve the best results on all joint-axis, joint-position, and joint-state metrics, as well as on two of the three geometry metrics. Baseline method's results are taken from iTACO.

In 2 of 73 sequences (storage furniture objects), our method predicts an additional moving kinematic axis that is absent from the ground truth; this mode is not captured by the benchmark metrics.

## 5 Conclusion and Limitations

We present Agentic STAR, an approach for shape tracking and reconstruction via agentic optimisation. By combining the coarse visual and structural reasoning of VLMs with numerical optimisation tools, Agentic STAR can recover objects with complex kinematics and track them under challenging conditions that remain difficult for feed-forward systems, including low visual overlap, severe occlusion, and rapid motion.

The main limitation of our approach is its inference-time computational cost. However, systems such as Agentic STAR could serve as compute-intensive teachers for the next generation of feed-forward perception models, generating structured reconstructions and trajectories for training. More broadly, alternating between agentic optimisation and feed-forward learning could provide a path toward iterative self-improvement.

## REFERENCES


Armen Avetisyan, Christopher Xie, Henry Howard-Jenkins, Tsun-Yi Yang, Samir Aroudj, Suvam Patra, Fuyang Zhang, Duncan Frost, Luke Holland, Campbell Orme, Jakob Engel, Edward Miller, Richard Newcombe, and Vasileios Balntas. Scenescript: Reconstructing scenes with an autoregressive structured language model. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

Prithviraj Banerjee, Sindi Shkodrani, Pierre Moulon, Shreyas Hampali, Shangchen Han, Fan Zhang, Linguang Zhang, Jade Fountain, Edward Miller, Selen Basol, Richard Newcombe, Robert Wang, Jakob Julian Engel, and Tomas Hodan. HOT3D: Hand and object tracking in 3D from egocentric multi-view videos. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

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

Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. Deepsvg: A hierarchical generative network for vector graphics animation. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, Aohan Lin, Jiawei Liu, Ziqi Ma, Anushka Sagar, Bowen Song, Xiaodong Wang, Jianing Yang, Bowen Zhang, Piotr Dollár, Georgia Gkioxari, Matt Feiszli, and Jitendra Malik. Sam 3d: 3dfy anything in images, 2025. URL https://arxiv.org/abs/2511.16624.

Zoey Chen, Aaron Walsman, Marius Memmel, Kaichun Mo, Alex Fang, Karthikeya Vemuri, Alan Wu, Dieter Fox, and Abhishek Gupta. Urdformer: A pipeline for constructing articulated simulation environments from real-world images. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

Jo ao Paulo Costeira and Takeo Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision (IJCV)*, 1998.

Alexandros Delitzas, Chenyangguang Zhang, Alexey Gavryushin, Tommaso Di Mario, Boyang Sun, Rishabh Dabral, Leonidas Guibas, Christian Theobalt, Marc Pollefeys, Francis Engelmann, and Daniel Barath. Reconstructing Functional 3D Scenes from Egocentric Interaction Videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

F. Dellaert and M. Kaess. Factor Graphs for Robot Perception. *Foundations and Trends in Robotics*, 6(1–2):1–139, 2017.

Jianning Deng, Kartic Subr, and Hakan Bilen. Articulate your nerf: Unsupervised articulated object modeling via conditional view synthesis. In *Neural Information Processing Systems (NeurIPS)*, 2024.

Carl Doersch, Ankush Gupta, Larisa Markeeva, Adria Recasens, Lucas Smaira, Yusuf Aytar, Joao Carreira, Andrew Zisserman, and Yi Yang. TAP-vid: A benchmark for tracking any point in a video. *Neural Information Processing Systems (NeurIPS)*, 35, 2022.

Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, Joao Carreira, and Andrew Zisserman. TAPIR: Tracking any point with per-frame initialization and temporal refinement. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2023.

Johan Edstedt, Qiyu Sun, Georg Bökman, Mårten Wadenbäck, and Michael Felsberg. RoMa: Robust Dense Feature Matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Zicong Fan, Omid Taheri, Dimitrios Tzionas, Muhammed Kocabas, Manuel Kaufmann, Michael J. Black, and Otmar Hilliges. ARCTIC: A dataset for dexterous bimanual hand-object manipulation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Haiwen Feng*, Junyi Zhang*, Qianqian Wang, Yufei Ye, Pengcheng Yu, Michael J. Black, Trevor Darrell, and Angjoo Kanazawa. St4rtrack: Simultaneous 4d reconstruction and tracking in the world. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Y. Gu, I. Huang, J. Je, G. Yang, and L. Guibas. Blendergym: Benchmarking foundational model systems for graphics editing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Adam W Harley, Yang You, Xinglong Sun, Yang Zheng, Nikhil Raghuraman, Yunqi Gu, Sheldon Liang, Wen-Hsuan Chu, Achal Dave, Suya You, et al. Alltracker: Efficient dense point tracking at high resolution. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2025.

Nick Heppert, Muhammad Zubair Irshad, Sergey Zakharov, Katherine Liu, Rares Andrei Ambrus, Jeannette Bohg, Abhinav Valada, and Thomas Kollar. Carto: Category and joint agnostic reconstruction of articulated objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Zhenyu Jiang, Cheng-Chun Hsu, and Yuke Zhu. Ditto: Building digital twins of articulated objects from interaction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

Justin Kerr, Chung Min Kim, Mingxuan Wu, Brent Yi, Qianqian Wang, Ken Goldberg, and Angjoo Kanazawa. Robot see robot do: Imitating articulated object manipulation with monocular 4d reconstruction. In *Conference on Robot Learning (CoRL)*, 2024.

Skanda Koppula, Ignacio Rocco, Yi Yang, Joe Heyward, João Carreira, Andrew Zisserman, Gabriel Brostow, and Carl Doersch. TAPVid-3D: A benchmark for tracking any point in 3D. *Advances in Neural Information Processing Systems*, 2024.

P. Kulits, H. Feng, W. Liu, V. F. Abrevaya, and M. J. Black. Re-thinking inverse graphics with large language models. *Transactions on Machine Learning Research*, 2024.

Long Le, Jason Xie, William Liang, Hung-Ju Wang, Yue Yang, Yecheng Jason Ma, Kyle Vedder, Arjun Krishna, Dinesh Jayaraman, and Eric Eaton. Articulate-anything: Automatic modeling of articulated objects via a vision-language foundation model. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

Taeyeop Lee, Bowen Wen, Minjun Kang, Gyuree Kang, In So Kweon, and Kuk-Jin Yoon. Any6D: Model-free 6d pose estimation of novel objects. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, 2025.

Vincent Leroy, Yohann Cabon, and Jerome Revaud. Grounding image matching in 3d with mast3r. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

Ruining Li, Yuxin Yao, Chuanxia Zheng, Christian Rupprecht, Joan Lasenby, Shangzhe Wu, and Andrea Vedaldi. Particulate: Feed-forward 3d object articulation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026a.

Zhe Li, Xiang Bai, Jieyu Zhang, Zhuangzhe Wu, Che Xu, Ying Li, Chengkai Hou, and Shanghang Zhang. URDF-anything: Constructing articulated objects with 3d multimodal language model. In *Neural Information Processing Systems (NeurIPS)*, 2025.

Zizhang Li, Cheng Zhang, Zhengqin Li, Henry Howard-Jenkins, Zhaoyang Lv, Chen Geng, Jiajun Wu, Richard Newcombe, Jakob Engel, and Zhao Dong. Art: Articulated reconstruction transformer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026b.

Jiayi Liu, Ali Mahdavi-Amiri, and Manolis Savva. PARIS: Part-level reconstruction and motion analysis for articulated objects. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2023a.

Jiayi Liu, Denys Iliash, Angel X. Chang, Manolis Savva, and Ali Mahdavi-Amiri. SINGAPO: Single image controlled generation of articulated parts in object. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

Shaowei Liu, Saurabh Gupta, and Shenlong Wang. Building rearticulable models for arbitrary 3d objects from 4d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023b.

Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Kirill Mazur, Marwan Taher, and Andrew Davison. 4d primitive-mache: Glueing primitives for persistent 4d scene reconstruction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter W. Battaglia. Polygen: An autoregressive generative model of 3d meshes. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

Van Nguyen Nguyen, Thibault Groueix, Mathieu Salzmann, and Vincent Lepetit. Gigapose: Fast and robust novel object pose estimation via one correspondence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Van Nguyen Nguyen, Christian Forster, Bugra Tekin, Sindi Shkodrani, Vincent Lepetit, Cem Keskin, and Tomáš Hodaň. Gotrack: Generic 6dof object pose refinement and tracking. *Computer Vision and Patern Recognition Workshops (CVPRW)*, 2025.

Bhawna Paliwal, Haritheja Etukuru, William Liang, Pieter Abbeel, Nur Muhammad Mahi Shafiullah, and Jitendra Malik. Do as i do: Dexterous manipulation data from everyday human videos. *arXiv preprint arXiv:2606.19333*, 2026.

Weikun Peng, Jun Lv, Cewu Lu, and Manolis Savva. iTACO: Interactable Digital Twins of Articulated Objects from Casually Captured RGBD Videos. In *3DV*, 2026.

Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. Meshgpt: Generating triangle meshes with decoder-only transformers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 1997.

Edgar Sucar, Eldar Insafutdinov, Zihang Lai, and Andrea Vedaldi. V-dpm: 4d video reconstruction with dynamic point maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

C. Sun, J. Han, W. Deng, X. Wang, Z. Qin, and S. Gould. 3d-gpt: Procedural 3d modeling with large language models. In *2025 International Conference on 3D Vision (3DV)*, 2025.

Jiaming Sun, Zihao Wang, Siyu Zhang, Xingyi He, Hongcheng Zhao, Guofeng Zhang, and Xiaowei Zhou. OnePose: One-shot object pose estimation without CAD models. *CVPR*, 2022.

Richard Szeliski. Rapid octree construction from image sequences. *CVGIP: Image Understanding*, 1993.

Marwan Taher, Ignacio Alzugaray, Kirill Mazur, Xin Kong, and Andrew J Davison. Kv-tracker: Real-time pose tracking with transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

Phil Tresadern and Ian Reid. Articulated structure from motion by factorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle Adjustment — A Modern Synthesis. In *Proceedings of the International Workshop on Vision Algorithms, in association with ICCV*, 1999.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. $\pi^3$: Scalable permutation-equivariant visual geometry learning. In *ICLR*, 2026.

Bowen Wen, Jonathan Tremblay, Valts Blukis, Stephen Tyree, Thomas Muller, Alex Evans, Dieter Fox, Jan Kautz, and Stan Birchfield. Bundlesdf: Neural 6-dof tracking and 3d reconstruction of unknown objects. *CVPR*, 2023.

Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. FoundationPose: Unified 6d pose estimation and tracking of novel objects. In *CVPR*, 2024.

Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

Yuxi Xiao, Qianqian Wang, Shangzhan Zhang, Nan Xue, Sida Peng, Yujun Shen, and Xiaowei Zhou. Spatialtracker: Tracking any 2d pixels in 3d space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Yuxi Xiao, Jianyuan Wang, Nan Xue, Nikita Karaev, Iurii Makarov, Bingyi Kang, Xin Zhu, Hujun Bao, Yujun Shen, and Xiaowei Zhou. Spatialtrackerv2: 3d point tracking made easy. In *ICCV*, 2025.

Jingyu Yan and Marc Pollefeys. A factorization-based approach for articulated nonrigid shape, motion and kinematic chain recovery from video. *TPAMI*, 2008.

Shaofeng Yin, Jiaxin Ge, Zora Zhiruo Wang, Xiuyu Li, Michael J. Black, Trevor Darrell, Angjoo Kanazawa, and Haiwen Feng. Vision-as-inverse-graphics agent via interleaved multimodal reasoning, 2026. URL https://arxiv.org/abs/2601.11109.

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 Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026a.

Ruihang Zhang, Felix Taubner, Pooja Ravi, Kiriakos N. Kutulakos, and David B. Lindell. Proxypose: 6-dof pose tracking via video-to-video translation. *arXiv preprint arXiv:2607.06555*, 2026b.

Mandi Zhao, Yijia Weng, Dominik Bauer, and Shuran Song. Real2code: Reconstruct articulated objects via code generation. In *International Conference on Learning Representations*, 2025.

Matt Zhou, Ruining Li, Xiaoyang Lyu, Zhaomou Song, Zhening Huang, Chuanxia Zheng, Christian Rupprecht, Andrea Vedaldi, and Shangzhe Wu. Articraft: An agentic system for scalable articulated 3d asset generation. *arXiv preprint arXiv:2605.15187*, 2026.

# A Appendix

## A.1 Sub-agent parallelisation

Visual inspection for pose estimation creates a computational bottleneck because image inspection is computationally expensive and most agents can inspect only a limited number of images. Image inspection is typically more expensive than the other operations, namely tool invocation and reasoning.

Shape and kinematic structure are centralised decisions. They depend on all frames. Once the shape is coherent enough (so that the notion of pose *makes sense*), we argue that pose estimates are parallelisable.

We adopt a simple schema for motion estimation parallelisation: we split the sequence into $M$ non-overlapping subsequences and spawn a pose *subagent* for each; its sole task is pose estimation for its chunk of ownership. Temporal inconsistencies between the seams are handled by the orchestrator via the same temporal residual tool. Each pose subagent receives a free-form brief indicating whether the current poses are in the correct basin or, e.g., require a flip.

## A.2 Rotation Representation

We parameterise rotational increments using an ordered composition of rotations around the camera-frame axes for VLM interpretability, since VLM agents excel at reasoning in terms such as "the object has to be slightly tilted to the left". Let $\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$ denote the camera-frame basis vectors. Given incremental angles $(\alpha, \beta, \gamma)$, we define:

$$\Delta R(\alpha, \beta, \gamma) = \exp(\alpha[\mathbf{e}_z]_\times) \exp(\beta[\mathbf{e}_y]_\times) \exp(\gamma[\mathbf{e}_x]_\times) \,. \tag{7}$$

Thus, $\Delta R$ is parameterised by successive rotations about the $Z$, $Y$, and $X$ axes of the camera frame.

## A.3 Temporal Residual Derivations

Our pose maps canonical object coordinates into camera coordinates:

$$\boldsymbol{p}_i^{camera} = sR_i\boldsymbol{p} + t_i, \qquad t_i = s\tau_i, \tag{8}$$

where the scale $s$ is shared across frames. Using the camera-to-world pose $(R_i^{\text{cam}}, t_i^{\text{cam}})$, eq. (6) gives:

$$\boldsymbol{p}_i^{world} = s(R_i^{\text{cam}} R_i)\boldsymbol{p} + (sR_i^{\text{cam}}\tau_i + t_i^{\text{cam}}) \,. \tag{9}$$

Our temporal tool defines the translational and rotation components of the object's world pose as:

$$W_i = R_i^{\text{cam}} R_i, \qquad u_i = R_i^{\text{cam}}\tau_i. \tag{10}$$

Here, $W_i$ is the object's world orientation, while $u_i$ is the camera-to-object offset expressed in world-oriented axes and object units. We exclude $t_i^{\text{cam}}$ because the camera tracker's translation gauge may not be compatible with the reconstruction scale.

For the step from frame $i-1$ to frame $i$, we define angular and translational velocities as:

$$\Omega_i = W_{i-1}^\top W_i, \qquad v_i = u_i - u_{i-1}. \tag{11}$$

The reported rotational velocity is the geodesic rotation angle $r_i^v = \|\log(\Omega_i)\|$ expressed in degrees, while the reported translational velocity is $v_i^t = \|v_i\|$ expressed in object units per step.

At frame $i$, translational acceleration is therefore:

$$a_i = v_{i+1} - v_i = u_{i+1} - 2u_i + u_{i-1}, \tag{12}$$

Rotational acceleration is defined as:

$$r_i^a = \left\|\log\left(\Omega_i^\top \Omega_{i+1}\right)\right\| . \tag{13}$$

**Radial Acceleration and Velocity.** Radial velocity and acceleration measure changes in the object's depth, which is typically the least constrained direction for monocular methods. They are computed along the camera-to-object direction $\hat{u}_i = u_i/\|u_i\|$. The signed radial velocity and acceleration are $v_i^{\text{rad}} = \hat{u}_i^\top v_i$ and $a_i^{\text{rad}} = \hat{u}_i^\top a_i$, respectively. Positive values indicate motion or acceleration away from the camera.

**Joints.** For joint $m$ with state $q_i^m$, the tool reports joint velocity $\Delta_i^m = q_i^m - q_{i-1}^m$ and joint acceleration $A_i^m = \Delta_{i+1}^m - \Delta_i^m$. Joint quantities are expressed in their native units: degrees for revolute joints and canonical object units for prismatic joints.

All quantities are reported as differences per step. When camera poses are unavailable, base-pose rotation and translation are computed directly in the camera frames.

### A.4 Scale Alignment

Monocular reconstruction is generally recovered up to an unknown global scale. Independently moving objects may have a separate scale gauge, even when the static scene is metric.

For each sequence chunk, we estimate one shared scale factor as:

$$\hat{s} = \exp\big(\text{median}[\log D_{\text{gt}}^t(\mathbf{u}) - \log D_{\text{pred}}^t(\mathbf{u})]\big)\,,$$

over pixels where both depths are valid. Pooling all frames and using the median makes the estimate robust to outliers. We apply the same alignment to all methods that do not receive metric depth as input.

### A.5 Gauge Alignment for HOT3D

**Model free $\mathbb{S}\text{im}(3)$ Gauge alignment.** For model-free 3D object tracking there's a natural $\mathbb{S}\text{im}(3)$ gauge for all predicted estimates, which corresponds to the choice of canonical object coordinate frame with respect to which the object is tracked. Since methods do not observe depth, the scale of the reconstruction and trajectory is not well constrained and is aligned for all methods.

$$p_{\text{gt}} = sGp_{\text{pred}} + c, \qquad G \in \mathbb{SO}(3)\,, \quad c \in \mathbb{R}^3, \quad s > 0. \tag{14}$$

Here, $s$ converts the arbitrary predicted length unit to metric units.

The predicted and ground-truth object-to-camera mappings are

$$q_{\text{pred}} = R_{\text{pred}}p_{\text{pred}} + t_{\text{pred}}, \qquad q_{\text{gt}} = R_{\text{gt}}p_{\text{gt}} + t_{\text{gt}}. \tag{15}$$

Because both the predicted geometry and camera-space translation are expressed in the same arbitrary unit, the complete predicted camera-space point must be scaled by $s$. Thus,

$$s\left(R_{\text{pred}}p_{\text{pred}} + t_{\text{pred}}\right) = R_{\text{gt}}\left(sGp_{\text{pred}} + c\right) + t_{\text{gt}} \tag{16}$$

$$= sR_{\text{gt}}Gp_{\text{pred}} + R_{\text{gt}}c + t_{\text{gt}}. \tag{17}$$

Equating the linear and translation terms gives:

$$R_{\text{pred}} \simeq R_{\text{gt}}G, \qquad st_{\text{pred}} \simeq t_{\text{gt}} + R_{\text{gt}}c. \tag{18}$$

We estimate the canonical rotation gauge using $\mathbb{SO}(3)$ Procrustes:

$$G^\star = \arg\min_{G\in\mathbb{SO}(3)} \sum_i \|R_{\text{pred},i} - R_{\text{gt},i}G\|_F^2\,. \tag{19}$$

For a fixed scale, the canonical translation is estimated by:

$$c^\star = \arg\min_c \sum_i \|st_{\text{pred},i} - t_{\text{gt},i} - R_{\text{gt},i}c\|_2^2\,. \tag{20}$$

While the scaling factor $s$ can also be jointly solved in the same optimisation problem, for methods that also predict a 3D geometric model $s$ is estimated from rendered predicted depth and metric capture depth alignment.

Therefore, the aligned gauge-aligned trajectory prediction is:

$$\hat{R}_{\text{pred}} = R_{\text{pred}}G^{\star\mathsf{T}}, \qquad \hat{t}_{\text{pred}} = st_{\text{pred}} - \hat{R}_{\text{pred}}c^\star. \tag{21}$$

**Model-based baselines.** For model based-baselines, such as FoundationPose that consume the metric GT CAD model as input, the gauge ambiguity is not present. However, to ensure the most fair evaluation we align the trajectory if scene object exhibits a natural symmetry (the group of symmetries is provided by the HOT3D dataset natively). We select the symmetry element minimising the rotational error in the first visible frame and apply this same element to every predicted pose. The selected symmetry is fixed across the entire sequence, and no additional pose alignment is performed.