FUNREC

Reconstructing Functional 3D Scenes from Egocentric Interaction Videos

Supplementary Material

Alexandros Delitzas[1,2] Chenyangguang Zhang[1] Alexey Gavryushin[1]
Tommaso Di Mario[1] Boyang Sun[1] Rishabh Dabral[2] Leonidas Guibas[3]
Christian Theobalt[2] Marc Pollefeys[1,4] Francis Engelmann[3,5] Daniel Barath[1]
[1]ETH Zurich [2]Max Planck Institute for Informatics [3]Stanford University [4]Microsoft [5]USI Lugano

This supplementary material complements the main paper by providing additional details about the proposed method and datasets. It also includes extended experimental results and further information on downstream applications. Finally, we discuss current limitations and outline potential directions for future work.

1. Additional method details

1.1. Proposed part pose parameterization

Instead of independently parameterizing each per-timestep part pose as a full 6-DoF rigid body transformation (*i.e.*, with unconstrained rotation and translation), we adopt a formulation that explicitly incorporates the physical constraints of articulated motion. This approach significantly reduces the number of optimizable parameters but also simplifies the optimization process. By embedding the articulation parameters directly into the transformation equations, they are optimized jointly with the geometry, ensuring that the physical constraints of the motion are inherently enforced. Each transformation $T_i^{m_k}$ represents the pose of part m_k at time i, mapping points from the canonical (part) coordinate system to the world coordinate system.

For *prismatic joints*, we define the pose of a part as:

$$T_i^{m_k} = \begin{bmatrix} I & \lambda_i \mathbf{a} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \|\mathbf{a}\| = 1 \tag{1}$$

Here, $\mathbf{a} \in \mathbb{S}^2$ is the unit articulation axis of the part m_k shared among all timesteps and $\lambda_i \in \mathbb{R}$ is the per-timestep displacement. This parameterization introduces $N + 2$ optimizable parameters in total, where N is the frame length of the interaction interval.

For *revolute joints*, we define the part pose as

$$T_i^{m_k} = \begin{bmatrix} R(\mathbf{a}, \theta_i) & (I - R(\mathbf{a}, \theta_i))\,\mathbf{p} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \|\mathbf{a}\| = 1, \mathbf{a}^\top \mathbf{p} = 0 \tag{2}$$

where $\mathbf{a} \in \mathbb{S}^2$ and $\mathbf{p} \in \mathbb{R}^3$ denote the unit articulation axis and the pivot point shared across all timesteps, and $\theta_i \in \mathbb{S}^1$ is the per-timestep angular displacement. The matrix $R(\mathbf{a}, \theta_i)$ denotes the rotation by angle θ_i around axis $\mathbf{a}$, expressed via Rodrigues' formula. The orthogonality constraint $\mathbf{a}^\top \mathbf{p} = 0$ ensures that the pivot point $\mathbf{p}$ lies on the plane orthogonal to the rotation axis $\mathbf{a}$. Geometrically, this guarantees that the part rotates around the axis passing through the pivot point, rather than inducing unintended translation along the axis direction. This parameterization introduces $N + 4$ optimizable parameters, where N is the frame length of the interaction interval.

1.2. Confidence-aware correspondence weighting

As discussed in the main paper, we modulate correspondence weights when performing rigid body fitting with SupeRANSAC [2] by incorporating confidence-aware sampling. Specifically, we use its PROSAC sampler, which prioritizes higher-confidence correspondences during hypothesis generation. We apply this strategy in two stages of our pipeline: camera pose estimation and part pose estimation. In both cases, PROSAC enables us to steer the solver toward more reliable correspondences while still allowing lower-confidence ones to contribute as soft evidence.

Camera pose estimation. Estimating camera motion from egocentric videos is particularly challenging due to the dominance of dynamic regions, most notably the hands and manipulated objects, which occupy large portions of the field of view. To mitigate their influence, we build a fragment-level pose graph and optimize it using selectively weighted RGB-D correspondences. Dense point correspondences are first extracted between consecutive and uniformly selected frames using RoMA [4]. We then filter these correspondences using per-frame hand and interacted-object masks from VISOR [3]. Matches whose pixels fall within hand masks $\{M_i^h\}$ are removed, while those in-

side interacted-object masks $\{M_i^{obj}\}$ are downweighted. Specifically, the RoMA confidence of each correspondence between frame i and j is scaled by the product of the interacted-object confidence values in the two frames, $\mathcal{C}_i^{obj} \cdot \mathcal{C}_j^{obj}$, softly suppressing correspondences that are likely to arise from moving objects. This weighting encourages the pose solver to rely primarily on stable, static scene structure when estimating camera motion.

Part pose estimation. In part pose estimation, we construct 3D-3D correspondences using the 3D tracks at different timesteps, *i.e.*, $\tau_{l,i}^m$ and $\tau_{l,j}^m$. Each correspondence pair is weighted by the product of its visibility scores $o_{l,i}^m \cdot o_{l,j}^m$.

1.3. Pixel-aligned part segmentation

To provide accurate mask prompts M_q^{sm} on keyframes for SAM2, we follow the pixel-aligned dense segmentation strategy described in the main paper. By combining projected motion tracks with SAM-based semantic grouping, we obtain a reliable moving part mask on each keyframe. Figure 1 illustrates an intuitive example, showing how projected tracks guide region selection and produce a motion-consistent mask that serves as an effective prompt for SAM2's video propagation module.

1.4. Part pose graph optimization

As discussed in the main paper, we estimate the articulation parameters and per-timestep part poses by optimizing the following objective:

$$\begin{aligned}\mathcal{L}(T^m, L^m, \phi^m) = &\sum_i f(T_i^m, T_{i+1}^m, T_{i\to i+1}^m) \\ &+ \sum_{i,j} l_{ij}^m \, f(T_i^m, T_j^m, T_{i\to j}^m) \\ &+ \mu \sum_{i,j} \left(\sqrt{l_{ij}^m} - 1\right)^2 \end{aligned} \tag{3}$$

where $l_{ij}^m \in [0,1]$ are optimized loop-closure confidences, and μ controls the regularization strength. The term

$$f(T_i^m, T_j^m, T_{i\to j}^m) = e_{ij}^\top \Omega_{ij} e_{ij},$$

where $e_{ij} = \log_{\mathrm{SE}(3)}\left((T_i^m)^{-1} T_j^m T_{i\to j}^m\right)$ measures the discrepancy of the estimated relative transformation $(T_i^m)^{-1} T_j^m$ and the observed transformation $T_{i\to j}^m$, weighted by the information matrix Ω_{ij}. The logarithm $\log_{\mathrm{SE}(3)}(\cdot)$ maps the residual to the tangent space of $\mathrm{SE}(3)$.

Loss terms. The objective consists of three components:

- Consecutive-frame edges: The first term enforces temporal consistency of adjacent frames. These edges are dense and typically reliable, anchoring the overall trajectory.
- Loop-closure edges: The second term includes longer-range frame connections, modulated by confidence variables $l_{ij}^m \in [0,1]$. These help correct accumulated drift but may be noisy, requiring robust weighting.
- Confidence regularization: The final term encourages the confidences. The penalty $(\sqrt{l_{ij}^m} - 1)^2$ suppresses low-confidence edges and prevents trivial solutions in which all loop closures are rejected.

Optimization strategy. We initialize the articulation parameters ϕ^m via a least-squares fit on the part tracks τ^m. The full objective is then minimized using Ceres Solver [1], employing the Levenberg-Marquardt algorithm together with manifold parameterizations for the articulation-specific variables, as described in Section 1.1.

Uncertain loop-closure rejection. To improve robustness, we perform optimization in two stages:

1. Initial optimization: Solve the full objective to estimate the loop-closure confidences l_{ij}^m.
2. Pruning: Remove uncertain loop closures with $l_{ij}^m < \eta_l$, where η_l is a confidence threshold.
3. Final optimization: Re-optimize the pose graph using only the remaining high-confidence edges.

We set the regularization weight as $\mu = \xi \, \epsilon_d^2$ where ϵ_d is the correspondence-distance threshold used when forming 3D matches, and ξ is the average number of correspondences per edge. This scales the confidence regularization to the expected noise level of the geometric evidence.

2. Datasets

A summary of the modalities of our newly introduced datasets is provided in Table 1.

2.1. Real scenes (RealFun4D)

Our dataset contains *in-the-wild* egocentric video captures collected in 60 real-world apartments across four countries, covering a total of 351 interactions. Each sequence includes synchronized multi-modal data: (1) RGB, (2) depth, (3) camera trajectories, (4) annotated interaction intervals, and (5) textual descriptions of the interactions. Each sequence is further annotated with (6) articulation parameters, (7) per-timestep part poses, (8) 2D part segmentation masks, (9) 2D hand masks, (10) 3D reconstructions of the static scene, and (11) 3D reconstructions of each articulated part.

Collection method. We capture data using a single head-mounted Azure Kinect DK. A human operator wearing the camera explores the scene and interacts with the articulated objects in it. To protect privacy, we remove or cover identifiable personal items (*e.g.*, photographs) from the scene prior to the capture. Using the official Azure Kinect SDK, we record synchronized RGB-D data (1920×1080 at 15 FPS) along with camera intrinsics. Each capture is processed in three stages.

In the first stage, we annotate the interaction intervals (*i.e.*, the start and end frames) and provide a short textual description for each interaction (*e.g.*, "open the fridge"). In the second stage, we annotate per-frame 2D masks for both

Keyframe I_q | Projected tracks I_q | Semantic pixel grouping G_q | Mask prompt M_q^{sm}

Figure 1. **Pixel-aligned part segmentation.** From left to right: a keyframe I_q is selected; sparse moving and static 3D tracks are projected into the keyframe; SAM2's automatic mask generator produces a semantic pixel grouping G_q; regions are scored using the motion ratio computed from projected tracks, yielding a motion-consistent mask prompt $\mathcal{M}_q^{sm}$. These keyframe prompts are then propagated by SAM2's video module, producing dense, temporally consistent part masks across the entire sequence.

Dataset	#Scenes	#Interactions	RGB	Depth	Camera Poses	Hand Masks	Part Masks	Part Poses	Articulation Params	Text Desc.	Interaction Intervals	3D Static Recon.	3D Part Recon.	Scene Type
RealFun4D	60	351	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	Real-world apartment rooms
OmniFun4D	12	127	✓	✓	✓	–	✓	✓	✓	✓	✓	✓	✓	Photorealistic simulated scenes

Table 1. **Our proposed datasets.** Overview of sensing modalities, annotations, and reconstruction outputs for our newly proposed datasets, RealFun4D and OmniFun4D.

human hands and interacted object parts. Following [7], these ground-truth masks are used to filter dynamic regions and perform offline RGB-D reconstruction to recover camera trajectories and the static, non-moving scene geometry. In the third stage, we annotate the articulation joints of the interacted parts directly in 3D space which specifies the kinematic structure (axis, pivot and joint type). Once the articulation joint is known, the part pose has only a single remaining degree of freedom which is the scalar displacement along the joint motion (*e.g.*, rotation angle for revolute joints or displacement distance for prismatic joints). To recover this 1-DoF trajectory, we additionally annotate one 2D track per frame. When lifted and combined with the joint parameters, this track resolves the unknown displacement, aligns the part across canonical poses and enables accurate 3D reconstruction of the articulated part geometry.

2.2. Simulated scenes (OmniFun4D)

We leverage the OmniGibson [5] simulator to record a total of 127 simulated, photorealistic interactions in 12 different scenes from the iGibson [14] environments, resulting in the OmniFun4D dataset. We provide the following modalities: (1) RGB (1920×1080 at 15 FPS), (2) depth, (3) camera trajectories, (4) annotated interaction intervals, (5) textual descriptions of corresponding interactions, (6) articulation parameters, (7) per-timestep part poses, (8) 2D part segmentation masks, (9) 3D model of the static scene, and (10) 3D model of each articulated part.

Collection method. The interactions for each scene are recorded by a human operator's pass through the simulated environment. The operator uses the keyboard and mouse to control a floating camera and sequentially trigger a set of scripted interactions in the scene. After recording the camera pose and triggered interactions at each timestep of the operator's pass, we replay the trajectory offline. Data for all aforementioned modalities is captured during this replay. Freed from the constraint of interactiveness, we improve the render quality using NVIDIA's photorealistic RTX Path Tracing renderer [9] and stochastically perturb the camera pose through a stateful Gaussian walk to imitate natural pose imperfections in human-recorded videos. We encourage the reader to watch the attached video for qualitative examples of interactions from OmniFun4D.

Figure 2. **Misalignment example in the HOI4D dataset.** Ground-truth 4D point clouds in HOI4D occasionally exhibit noticeable misalignment with the object mesh transformed using the provided part-pose annotations. To ensure a reliable evaluation set, we manually inspect all sequences and retain only those scenes that demonstrate consistent alignment between the reconstructed geometry and the annotated part poses.

2.3. HOI4D

We convert the original HOI4D dataset [7] into our format and retain 30 videos with high-quality annotations after filtering. We provide the following modalities: (1) RGB (1920×1080 at 15 FPS), (2) depth, (3) camera trajectories, (4) annotated interaction intervals, (5) textual descriptions of corresponding interactions, (6) articulation parameters, (7) per-timestep part poses, (8) 2D part segmentation masks, (9) 3D model of the interacted object's static part, and (10) 3D model of the interacted object's articulated part.

Collection method. We process the original dataset to ex-

tract ground-truth articulation parameters as defined in Section 1.1 from the dataset's timestep-wise, per-part 6DoF poses. Specifically, to estimate articulation parameters for revolute articulations, we estimate the best-fitting plane using PCA, project 10,000 randomly sampled mesh surface points to that plane, set the initial estimate of the circle's center and radius to the mean and the mean distance to the center, then refine these parameters using least squares. For prismatic articulations, we perform a least-squares fitting of the line induced by the movement of the surface point closest to the object's center, from a set of 10,000 randomly sampled mesh surface points.

We note that HOI4D's ground-truth 4D point cloud occasionally exhibits alignment issues with the object mesh transformed according to HOI4D's ground-truth part pose data. This misalignment is already present in the original HOI4D release (Figure 2). To construct a high-quality subset for our evaluation, we hence perform a manual inspection and select well-aligned scenes from the original dataset, further ensuring a high-quality articulation estimation from our previously described pipeline for each selected scene.

3. Experiments

3.1. Further details on the baselines

We compare against three representative categories of prior work: (1) 4D reconstruction pipelines (MonST3R [17], SpatialTrackerV2 [16]); (2) 6D object pose tracking methods (BundleSDF [15]); and (3) articulated object reconstruction approaches (ArtGS [8]). Below we describe the baseline implementations in greater detail.

MonST3R [17]. Given an input egocentric video, we first use MonST3R to obtain camera poses, estimated depth maps, and dynamic masks. We adopt the window-wise optimization implementation of MonST3R to process videos of arbitrary length. The dynamic parts are then backprojected into the 3D world space using the estimated camera poses and depth maps (optionally, we use ground truth depth for fair comparison). For part-level pose estimation, we consider two alternative implementations. The first approach uses ICP with RANSAC to register the relative part poses to the first frame. The second approach lifts 2D tracks, estimated by CoTracker3 [6] within the dynamic mask of each frame, into 3D using depth information. Using these tracks, part poses can be registered via a RANSAC based regression. After part poses are estimated, articulation parameters are obtained via RANSAC-based line fitting.

SpatialTrackerV2 [16]. Unlike MonST3R, SpatialTrackerV2 directly provides 3D tracks in world space but does not generate dynamic masks. Thus, we apply a distance threshold to determine which tracks are dynamic. The dynamic 3D tracks are projected to 2D, and the resulting dynamic pixels are fed into SAM2 to propagate dynamic masks across frames. Using these masks together with the estimated (or optionally ground-truth) depth maps and camera poses, we backproject the dynamic points in each frame into a 3D point cloud. Finally, we estimate part poses from these 3D dynamic tracks using RANSAC. The dynamic point clouds are canonicalized with respect to the recovered part poses, and articulation parameters are extracted by performing RANSAC-based line (prismatic joints) or circle (revolute joints) fitting to these poses. Since SpatialTrackerV2 lacks a window-wise mechanism for arbitrarily long videos, we downsample each sequence to 30 frames due to GPU limits and linearly interpolate the predictions for the remaining frames.

BundleSDF [15]. BundleSDF requires dynamic masks as input and outputs both the reconstructed part geometry in canonical space and the transformation matrices from canonical to camera space. However, in practice, we observe that it is prone to failure when the provided masks are not sufficiently accurate. Thus, we use the ground truth dynamic masks in our experiments. Additionally, we use ground truth camera poses to decouple camera motion and estimate the part pose results. The same RANSAC-based line or circle fitting is utilized to obtain the articulation parameters from the estimated part poses.

ArtGS [8]. ArtGS is a two-state articulation reconstruction method that trains a 3D model using Gaussian Splatting from multi-view static RGB-D observations. It requires ground-truth camera poses and two articulation states (before and after interaction) as input. Since ArtGS cannot natively handle dynamic video, we substitute the frames captured before any motion and those at the ground-truth maximum articulation state as its two required static states. Due to its lack of temporal tracking, ArtGS produces only the static reconstruction and articulation parameters, but cannot provide state tracks across dynamic frames.

3.2. Implementation details

Our method runs on a single NVIDIA GeForce RTX 4090 GPU (24 GB). For motion clustering, we use thresholds $\epsilon_f = 0.01$, $\epsilon_s = 0.01$ and $\eta_m = 0.5$. In experiments with simulated data, where human hand cues are not available for selecting the moving-part cluster, we instead compute the per-cluster score based on the maximum visible deformation of the tracks within each cluster. For pixel-aligned part segmentation, we select keyframes every $\Delta_Q = 10$ frames. For part pose estimation, we use an edge-pruning threshold of $\eta_l = 0.3$ and a correspondence distance threshold of $\epsilon_d = 0.01$.

3.3. Additional quantitative results

We provide additional results on camera pose estimation in Table 2.

Methods	OmniFun4D			HOI4D			RealFun4D		
	ATE (m) ↓	RPE trans (m) ↓	RPE rot (°) ↓	ATE (m) ↓	RPE trans (m) ↓	RPE rot (°) ↓	ATE (m) ↓	RPE trans (m) ↓	RPE rot (°) ↓
MonST3R [17]	0.0001	**0.0000**	0.2638	0.0096	0.0066	0.3512	0.0537	0.0163	0.7425
SpatialTrackerV2 [16]	**0.0000**	**0.0000**	1.4721	0.0254	0.0101	1.5213	0.1318	0.0287	1.9036
FunREC (Ours)	**0.0000**	**0.0000**	**0.2312**	**0.0071**	**0.0038**	**0.1684**	**0.0063**	**0.0027**	**0.1456**

Table 2. **Camera pose estimation results across all datasets.** We report absolute trajectory error (ATE), relative pose error in translation (RPE trans), and relative pose error in rotation (RPE rot) for OmniFun4D, HOI4D, and RealFun4D. **Best** , second-best , and third-best .

Ablation settings	Articulation estimation		6D part pose		Segm.
	Axis (°) ↓	Pos (m) ↓	ADD-S (%) ↑	ADD (%) ↑	mIoU (%) ↑
(A1) w/o VLM-predicted articulation type	23.6/24.7	0.19	53.12	46.85	58.6
(A2) w/ SE(3)*-only part poses*	19.4/ 9.6	0.10	69.88	62.41	71.2
(A3) w/o articulation-aware clustering	25.2/20.5	0.22	50.94	44.12	52.9
(A4) w/o voting-based semantic association	6.3/ 5.9	0.06	73.84	65.21	53.7
(A5) w/o part pose graph	23.9/10.8	0.15	58.74	51.06	74.3
(A6) w/o confidence-aware part pose estimation	8.7/ 6.3	0.10	70.11	62.95	74.4
FunREC (Ours, full)	**5.6/ 5.5**	**0.05**	**75.62**	**68.11**	**74.8**

Table 3. **Ablation study.** We evaluate the effect of key method components on the RealFun4D dataset.

3.4. Additional qualitative comparisons

We provide further qualitative comparisons in Figure 3 and Figure 4.

3.5. Ablation study

We analyze the contribution of key components in Table 3. Ablation (A1) infers the articulation type from motion only, without VLM predictions. (A2) removes the articulation-centric parameterization and predicts unconstrained per-frame SE(3) motions for the interacted parts. Without the physically plausible kinematic structure enforced by a shared articulation axis, the estimated motion drifts across frames and becomes less consistent. (A3) replaces articulation-aware clustering with standard clustering. (A4) removes our voting-based semantic association and relies solely on raw projected moving tracks to prompt SAM2 [12], without image-level semantic grouping or aggregating evidence across regions via motion ratios. Without the voting mechanism, sparse or noisy tracks often activate incorrect image regions, causing fragmented or overly large masks. In contrast, our full approach leverages region-level consensus over moving and static track counts, yielding significantly more accurate and stable pixel-aligned masks. (A5) removes the part pose graph optimization. Without loop closures and global consistency constraints, drift accumulates across frames, weakening both the recovered part poses and their coupling to the articulation parameters. (A6) uses uniform weighting instead of confidence-aware part pose estimation.

4. Further details on the applications

Simulation-ready export. The functional reconstruction provides both the geometry of the environment and the articulation structure of all interactive parts. It lacks physical properties and needs to be transformed into a physics-enabled file format. We first generate a URDF (Unified Robot Description Format) model from the reconstruction, which jointly encodes the geometry of each component and the joint configuration connecting all movable parts in the scene. In our setup, the static environment serves as the root link, and all interactable elements are attached via either prismatic or revolute joints. Using the step-wise articulated poses observed during interaction, we also extract joint limits for each part.

For basic physical parameters such as mass and inertia, we select an image where the object of interest is clearly visible and query GPT-5 using the following prompt:

```
You are a physical property estimation model for
    robotics.
Given (1) an input image and (2) the name or description
     of the object of interest,
you estimate the object's physical properties based only
     on visual evidence,
standard material assumptions, and common real-world
    statistics.

Your job is NOT to identify the correct values but to
    produce reasonable,
self-consistent estimates suitable for physics
    simulation (e.g., Isaac Sim, PyBullet, MuJoCo).

Follow these rules:
1. First, identify the object's geometry (shape,
    dimensions).
2. Infer likely material(s) based on texture, color, and
     context.
3. Estimate density using standard physical values.
4. Compute mass using geometry x density.
5. Compute inertia tensor using classical rigid body
    formulas.
6. State all assumptions explicitly.
7. Return all outputs in SI units.
8. Never reference that this is a guess - just give the
    best estimate.
9. Output must be in the following format:

{
  "object_name": <object name>,
  "assumed_geometry": { "shape": "...", "dimensions_m":
      { ... } },
  "assumed_material": "...",
  "estimated_density_kg_m3": ...,
  "estimated_mass_kg": ...,
  "inertia_tensor_kg_m2": {
     "Ixx": ..., "Iyy": ..., "Izz": ...,
     "Ixy": 0.0, "Ixz": 0.0, "Iyz": 0.0
  },
  "notes": "..."
}

You must always return valid JSON.
```

We insert the estimated mass and inertia values into the URDF and then import the resulting model into Isaac Sim. After enabling rigid-body properties for all articulated com-

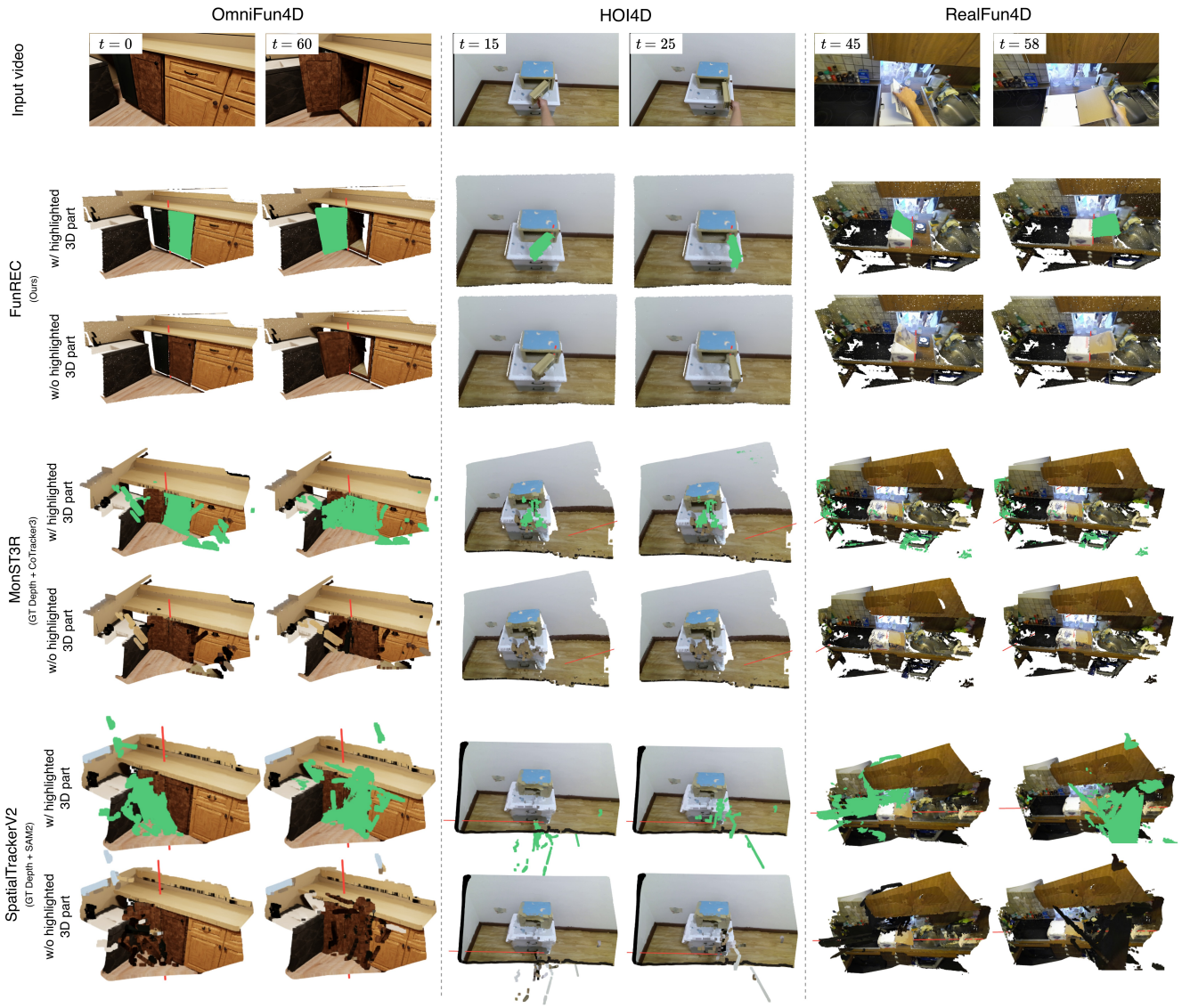

Figure 3. **Additional qualitative comparisons**. We show qualitative comparisons between baselines and our method. For each method, we accumulate the reconstructed point clouds of both the articulated part and the static scene across all timesteps, and visualize them under two selected scene states. Green indicates the articulated part, and red lines denote the estimated articulation axes. Each reconstruction is shown both with the articulated part highlighted (green), and without highlighting to show the RGB color.

ponents, the objects become physically interactive and can be manipulated by applying forces to their bodies (Figure 5).

To enable robot-scene interaction within this simulated environment, we additionally load a predefined robot model, such as a Franka arm, into Isaac Sim and implement an inverse-kinematics-based end-effector controller to command its motion. This allows the robot to interact with the reconstructed scene and opens the door to a range of downstream applications, including robot learning with high-fidelity digital twins captured from the real world.

Hand-guided affordance mapping. We visualize our data together with scene-embedded hand meshes in Figure 6. Given the RGB-D input, off-the-shelf hand pose reconstruction methods, such as [10, 11], can be used to obtain the 2D hand keypoints as well as hand pose of the agent interacting with the scene. The hand pose can be transformed into a hand mesh using MANO [13], featuring 3D hand joints. From the depth value at a given 2D hand keypoint, we can calculate the position of the corresponding 3D hand keypoint in space. This allows us to estimate a transformation of the reconstructed hand mesh into the 3D scene. From the scene-embedded hand mesh, the minimum Euclidean dis-

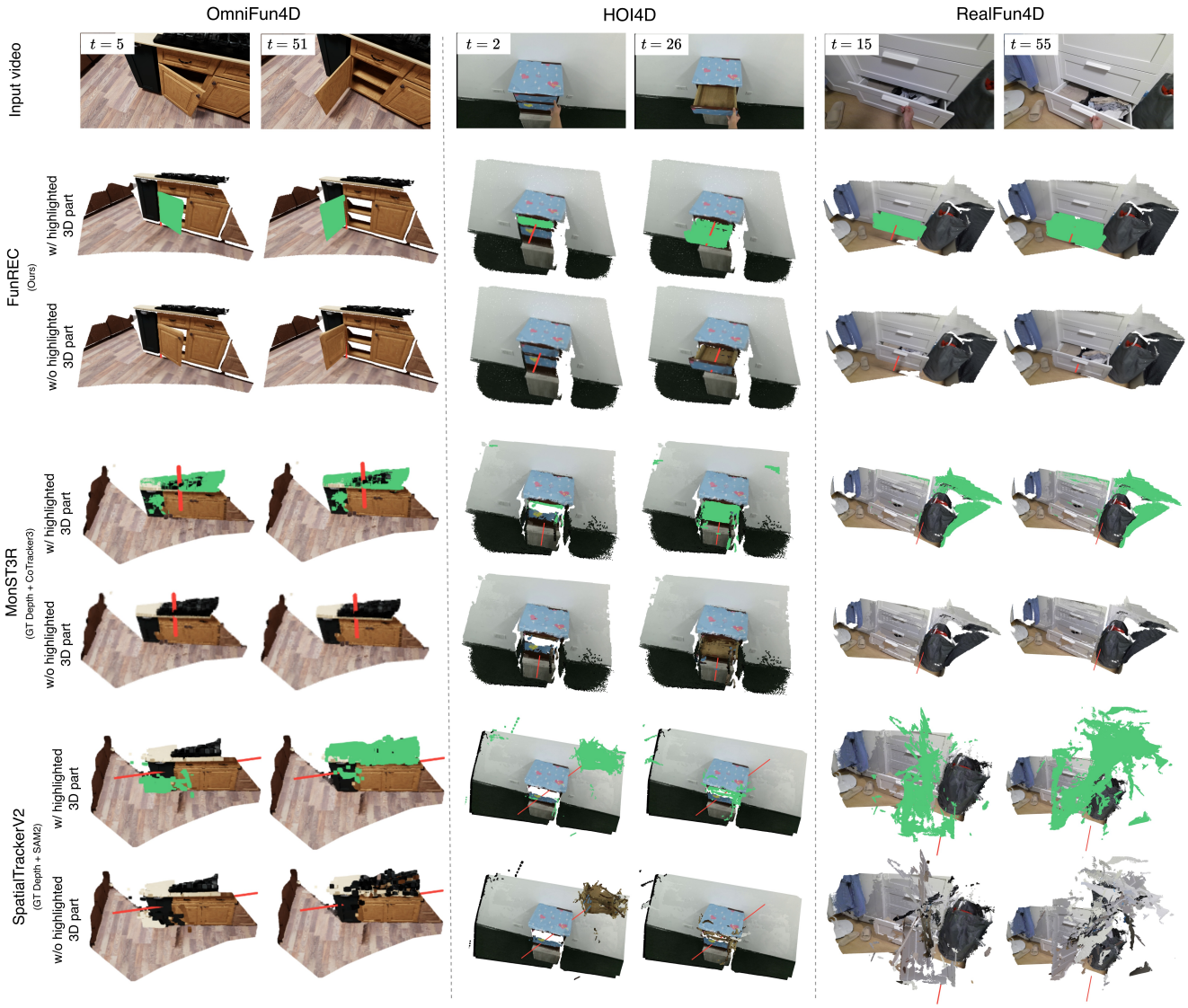

Figure 4. **Additional qualitative comparisons**. We show qualitative comparisons between baselines and our method. For each method, we accumulate the reconstructed point clouds of both the articulated part and the static scene across all timesteps, and visualize them under two selected scene states. Green indicates the articulated part, and red lines denote the estimated articulation axes. Each reconstruction is shown both with the articulated part highlighted (green), and without highlighting to show the RGB color.

tance from each fingertip vertex to any vertex of a dynamic part in the scene can be calculated. For a given timestep, dynamic parts sufficiently close to the fingertips can be considered as actively interacted parts.

Robot-scene interaction from human demonstration. We evaluate our method in an indoor apartment-like environment using a Boston Dynamics Spot robot equipped with an arm. The environment contains common household furniture such as cabinets, a refrigerator, and desks. We first record a single human demonstration of interacting with articulated objects from a first-person perspective, for example opening cabinet doors. From the recorded egocentric RGB-D video, we recover the articulation model and the per-timestep object part poses. We then estimate the contact point using the hand-guided affordance described in the previous subsection. Using the extracted contact point together with the articulation parameters, we generate a reference trajectory and anchor it in the 3D space for the robot's end effector. The trajectory consists of a sequence of 6D poses. Before executing the trajectory, the robot opens its gripper, moves to the first pose, and then closes the gripper to grasp the object handle. We then command Spot's impedance arm controller through its API to track the reference trajectory. Figure 7 shows examples that compare

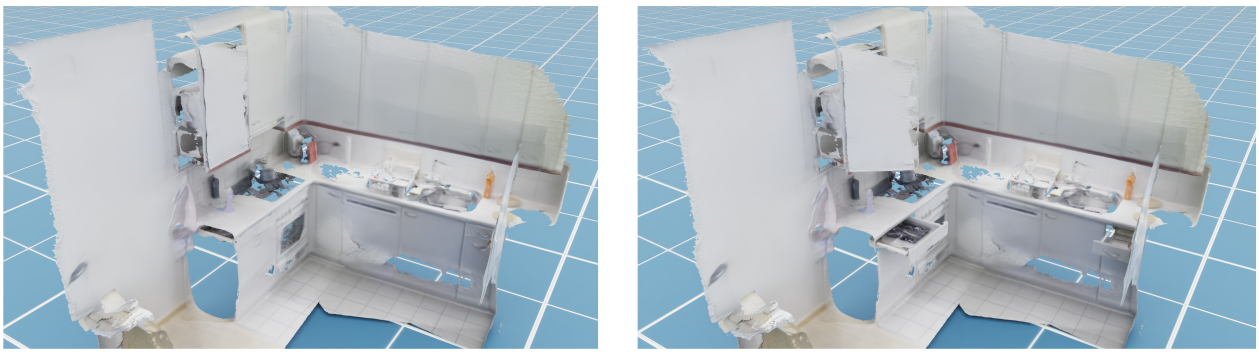

Figure 5. **Isaac Sim example.** Left: A USD-format reconstruction loaded in Isaac Sim with ground and lighting added. Right: After enabling physics and rigid-body properties for the interactive objects, they respond to external forces applied to their bodies.

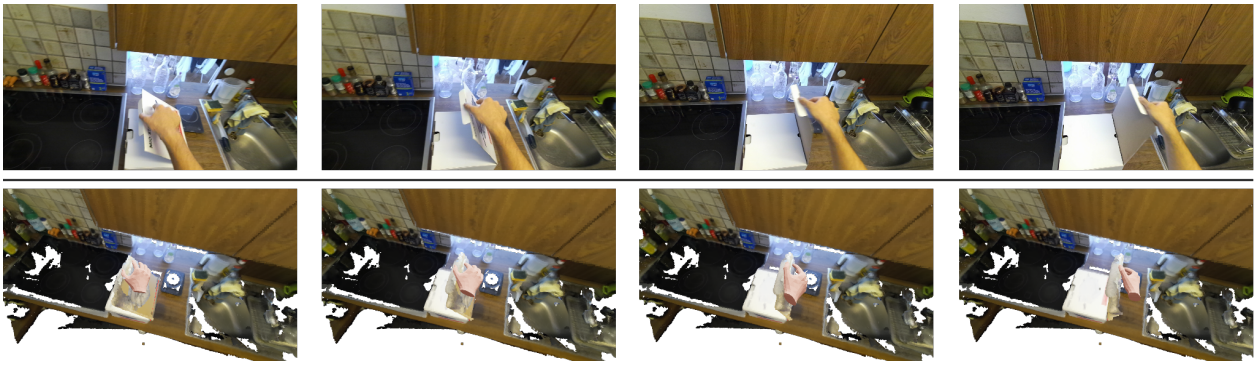

Figure 6. **Hand-object interaction example.** We reconstruct the hand during an interaction sequence and embed it into the scene. Top row: Original RGB frames. Bottom row: Functional 3D scene reconstruction with the embedded hand mesh.

the human demonstration and the resulting robot interaction following our workflow.

5. Limitations and future work

Despite achieving state-of-the-art results, FunREC's performance can still be influenced by the quality of input depth maps. Because our approach relies on stable and consistent 3D point trajectories to track the interacted part, inaccurate depth measurements, such as those caused by flickering, reflective surfaces, or transparent materials, may lead to failure cases. Incorporating monocular depth priors could help reduce this dependency and move toward a fully RGB-only pipeline.

A second limitation is that our method depends on pretrained models and therefore inherits their biases and failure modes. As 3D foundation models continue to improve, we expect the robustness of our system to benefit correspondingly.

Finally, our method currently assumes that only a single articulated part is being manipulated at a time. However, real-world interactions may involve multiple simultaneous actions, for instance opening a drawer while placing a rigid object inside. Extending FunREC to support multiple independently moving parts and additional object types (e.g., rigid objects) is an important direction for future work. This would require adapting our motion clustering module to handle multiple motion groups and generalizing our part-pose estimation to full 6-DoF modeling within the optimization framework. Additionally, as our method relies on multiple sequential optimization stages, errors may accumulate across steps; developing a feedforward formulation to mitigate this is another promising direction.

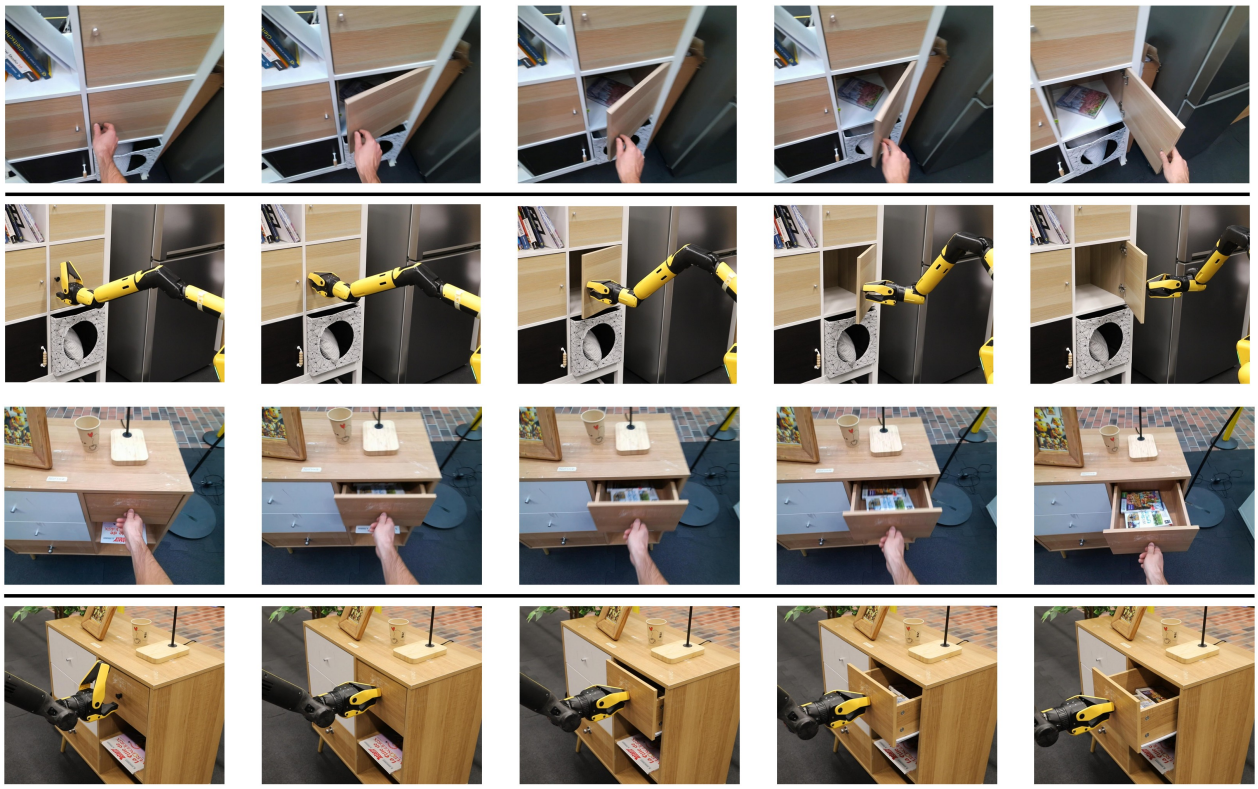

Figure 7. **Robot-scene interaction examples.** We present both human demonstrations and robot executions. Top two rows: Interaction with a cabinet door featuring a revolute joint. Bottom two rows: Interaction with a drawer featuring a prismatic joint.

References

[1] Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. Ceres Solver, 2023. 2

[2] Daniel Barath. Superansac: One ransac to rule them all. *arXiv preprint arXiv:2506.04803*, 2025. 1

[3] Ahmad Darkhalil, Dandan Shan, Bin Zhu, Jian Ma, Amlan Kar, Richard Higgins, Sanja Fidler, David Fouhey, and Dima Damen. EPIC-KITCHENS VISOR Benchmark: VIdeo Segmentations and Object Relations. In *International Conference on Neural Information Processing Systems (NeurIPS)*, 2022. 1

[4] Johan Edstedt, Qiyu Sun, Georg Bökman, Mårten Wadenbäck, and Michael Felsberg. RoMa: Robust Dense Feature Matching. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1

[5] Chengshu Li et al. BEHAVIOR-1K: A Human-Centered, Embodied AI Benchmark with 1,000 Everyday Activities and Realistic Simulation. *arXiv preprint arXiv:2403.09227*, 2024. 3

[6] Nikita Karaev, Iurii Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker3: Simpler and Better Point Tracking by Pseudo-Labelling Real Videos. In *International Conference on Computer Vision (ICCV)*, 2025. 4

[7] Yunze Liu, Yun Liu, Che Jiang, Kangbo Lyu, Weikang Wan, Hao Shen, Boqiang Liang, Zhoujie Fu, He Wang, and Li Yi. HOI4D: A 4D Egocentric Dataset for Category-Level Human-Object Interaction. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[8] Yu Liu, Baoxiong Jia, Ruijie Lu, Junfeng Ni, Song-Chun Zhu, and Siyuan Huang. Building Interactable Replicas of Complex Articulated Objects via Gaussian Splatting. In *International Conference on Learning Representations (ICLR)*, 2025. 4

[9] NVIDIA. NVIDIA® RTX Path Tracing, 2023. 3

[10] Georgios Pavlakos, Dandan Shan, Ilija Radosavovic, Angjoo Kanazawa, David Fouhey, and Jitendra Malik. Reconstructing Hands in 3D With Transformers. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 6

[11] Rolandos Alexandros Potamias, Jinglei Zhang, Jiankang Deng, and Stefanos Zafeiriou. Wilor: End-to-end 3d hand localization and reconstruction in-the-wild. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 6

[12] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment Anything in Images and Videos. *arXiv preprint arXiv:2510.11340*, 2024. 5

[13] Javier Romero, Dimitrios Tzionas, and Michael J. Black.

Embodied Hands: Modeling and Capturing Hands and Bodies Together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6), 2017. 6

[14] Bokui Shen, Fei Xia, Chengshu Li, Roberto Martín-Martín, Linxi Fan, Guanzhi Wang, Claudia Pérez-D'Arpino, Shyamal Buch, Sanjana Srivastava, Lyne P. Tchapmi, Micael E. Tchapmi, Kent Vainio, Josiah Wong, Li Fei-Fei, and Silvio Savarese. iGibson 1.0: A Simulation Environment for Interactive Tasks in Large Realistic Scenes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021. 3

[15] 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. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 4

[16] Yuxi Xiao, Jianyuan Wang, Nan Xue, Nikita Karaev, Yuri Makarov, Bingyi Kang, Xing Zhu, Hujun Bao, Yujun Shen, and Xiaowei Zhou. Spatialtrackerv2: Advancing 3d point tracking with explicit camera motion. In *International Conference on Computer Vision (ICCV)*, 2025. 4, 5

[17] Junyi Zhang, Charles Herrmann, Junhwa Hur, Varun Jampani, Trevor Darrell, Forrester Cole, Deqing Sun, and Ming-Hsuan Yang. MonST3R: A Simple Approach for Estimating Geometry in the Presence of Motion. In *International Conference on Learning Representations (ICLR)*, 2025. 4, 5