

Generalizable Articulated Object Reconstruction from Casually Captured RGBD Videos

Weikun Peng¹, Jun Lv², Cewu Lu², Manolis Savva¹

¹Simon Fraser University ²Shanghai Jiao Tong University

[3dlg-hcvc.github.io/video2articulation/](https://github.com/3dlg-hcvc/video2articulation/)

Abstract

Articulated objects are prevalent in daily life. Understanding their kinematic structure and reconstructing them have numerous applications in embodied AI and robotics. However, current methods require carefully captured data for training or inference, preventing practical, scalable, and generalizable reconstruction of articulated objects. We focus on reconstruction of an articulated object from a casually captured RGBD video shot with a hand-held camera. A casually captured video of an interaction with an articulated object is easy to acquire at scale using smartphones. However, this setting is quite challenging, as the object and camera move simultaneously and there are significant occlusions as the person interacts with the object. To tackle these challenges, we introduce a coarse-to-fine framework that infers joint parameters and segments movable parts of the object from a dynamic RGBD video. To evaluate our method under this new setting, we build a $20\times$ larger synthetic dataset of 784 videos containing 284 objects across 11 categories. We compare our approach with existing methods that also take video as input. Experiments show that our method can reconstruct synthetic and real articulated objects across different categories from dynamic RGBD videos, outperforming existing methods significantly.

1 Introduction

Articulated objects are prevalent in daily life. Building digital representations of such objects and understanding their kinematic structures has many applications in computer animation, embodied AI, and robotics. For example, automatically building interactable digital twins of articulated objects can benefit robotics research, since researchers can train and test manipulation policies more conveniently and efficiently in simulation prior to real-world deployment. However, to the best of our knowledge, current approaches to articulated object reconstruction impose many constraints on the input data, preventing practical and efficient deployment for acquisition of articulated objects in realistic settings. Thus, researchers still largely rely on manual annotation of articulation joint parameters [39].

Recent approaches to reconstructing articulated objects can be categorized into three families. The first uses a deformation model for implicit functions to represent the dynamics of articulated objects [26, 40, 46, 34]. However, the resulting implicit representation is not easy to use in simulation as we first need to reconstruct the articulated object explicitly and find rigid transformations of the different object parts. Thus, this is not a practical path for physically accurate animation or interaction. The second family reconstructs articulated objects by estimating joint parameters explicitly from two different states of the object [8, 16, 47, 19, 21, 27]. However, this type of approach requires aligning observations of two states to the same coordinate frame. Prediction accuracy can drop drastically due to slight errors or perturbations in the coordinate frame alignment of observations at each state. The third family of approaches generates articulated objects leveraging large language models and vision foundation models to help predict joint parameters [22, 13]. At present, these approaches either

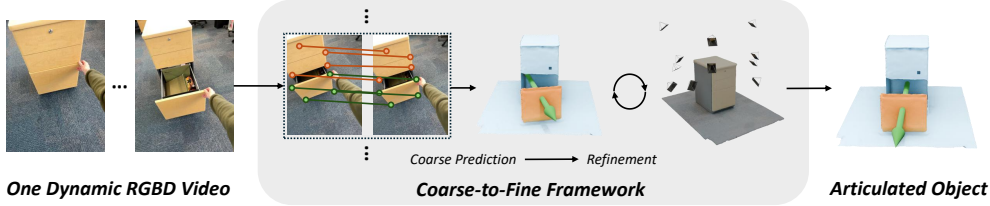


Figure 1: We propose a coarse-to-fine framework that can reconstruct an articulated object and relevant articulation parameters from a casually captured RGBD video. Our pipeline first predicts coarse joint parameters and movable part segmentation. The initial estimates are then refined with gradient-based optimization.

rely on an external library for object retrieval or require additional data for fine-tuning. Retrieving objects from a library is not easily scalable, while collecting and annotating data for fine-tuning is labor-intensive. Furthermore, these approaches do not easily generalize well to real settings after fine-tuning on limited data.

By analyzing the issues of recent approaches, we summarize three desirable features for a practical articulated object reconstruction pipeline: **1) minimal constraints on input; 2) explicit joint parameter and geometry representation for easy integration into simulators; and 3) generalization across diverse categories of articulated objects.** To address these desiderata, we propose to reconstruct articulated objects from casually captured RGBD videos of interaction with the object (i.e. a video shot from a hand-held RGBD camera while the same person recording interacts with the object). Casually captured videos, or dynamic videos, are easy to capture at scale and contain the necessary motion information to infer joint parameters. Compared to the problem setting in prior work, the input video data is far less constrained and easier to capture, the resulting explicit representation of articulated objects is easier to use for simulation, and the overall pipeline is scalable to real-world settings.

However, this kind of input also brings more challenges to this problem. Since both the camera and the scene are dynamic, and the person occludes the object as they are manipulating it, decoupling camera motion and object part motion is challenging. Consequently, accurate estimation of joint parameters and movable part segmentation is difficult. To address these challenges, we develop a coarse-to-fine framework to reconstruct articulated objects from dynamic RGBD videos, illustrated in Fig. 1. Our pipeline first estimates joint parameters and a movable part segmentation via image feature matching. Then, a gradient-based optimization framework refines these initial estimates against a surface point cloud representation of the object acquired through 3D reconstruction. Our pipeline explicitly parameterizes the articulation joint parameters and other relevant parameters, satisfying the second feature. Moreover, to aid generalization and satisfy the third desideratum, our pipeline neither relies on an external library nor requires additional data for fine-tuning. Instead, it only uses pretrained models.

Since the problem setting and task setup we described has not been previously addressed, we build a new dataset for evaluation. We select 284 synthetic objects from 11 categories in the PartNet-Mobility dataset [25] to generate dynamic videos for input and evaluate the performance of our method. Compared to datasets used in prior works such as PARIS [16] and ArtGS [19], our dataset contains $20\times$ more objects and thus provides a more robust evaluation of the performance of different methods for reconstructing articulated objects from dynamic videos.

In summary, we make the following contributions:

1. We address the task of articulated object reconstruction from a casually captured RGBD video, a new problem setting that is more practical than prior work.
2. To tackle this new problem, we develop a coarse-to-fine framework that reconstructs articulated objects explicitly and does not rely on an external articulated object library or additional data for fine-tuning.
3. To evaluate our method robustly, we build a new synthetic dataset with $20\times$ more objects than prior datasets. Experiments show that our method outperforms state of the art methods on reconstruction of both synthetic and real articulated objects across different categories.

2 Related work

Our work lies at the intersection of articulated object reconstruction and dynamic scene understanding. We review recent work in both fields to provide a comprehensive background of our work.

2.1 Articulated object reconstruction

Articulated object reconstruction is a long-standing task in computer vision and robotics. One line of research focuses on the visual fidelity of the dynamics of the articulated objects [26, 40, 46, 34]. They usually apply implicit representations and deformation fields to model the articulated objects. This type of representation is not suitable for physically accurate animation and interaction. It requires extra effort to build an explicit representation to integrate with the simulator. In this paper, we aim to build a tool for reconstructing an interactable digital twin of articulated objects, where explicitly modeling the object and inferring joint parameters is essential. Thus, this line of work is not suitable for our problem setting.

Another line of work attempts to infer joint parameters and build a digital twin of articulated objects from multi-view observations of different object states [8, 16, 47, 19]. These works implicitly assume all the observations have already been aligned to the same coordinate. In practice, human operators need to manually align the camera coordinates of all the observations, which can not scale to large amounts of data. Similarly, in robotics community, researchers use robots to interact with the articulated object and reconstruct articulated objects from the interaction [21, 27, 51, 45]. Since robots usually stay still while manipulating objects, different observations at different object states can be easily aligned with the robot’s coordinate. However, robots are expensive and may damage themselves during the interaction. Therefore, it’s not an efficient way to build a digital twin of articulated objects. Robot-See-Robot-Do [10] recovers 3D motions of the articulated object from a monocular human demonstration video. Their problem setting is close to us but requires a static view video and is sensitive to the camera view.

Recently, with the advancement of large language models and generative models, some researchers try to leverage those models to directly generate code representation or kinematic graph from a single image of an articulated object [22, 13, 2, 17]. With the power of vision-language models, Articulate-Anything can also take text or video as input [13]. These works either rely on existing mesh libraries to retrieve object meshes or require extra data for fine-tuning foundation models. Thus, at the current stage, this line of work still suffers from the out-of-distribution issue, which prevents it from being fully generalizable and scalable. In contrast, our method does not rely on object retrieval or fine-tuning, and is inherently more scalable.

2.2 Dynamic scene reconstruction

Dynamic scene understanding and reconstruction is a challenging task in computer vision and is still an open problem. One line of research continues extending neural radiance field and 3D gaussian splatting to represent dynamic scenes [29, 3, 54, 20, 48, 41, 50, 43]. These works achieve impressive results in the novel view rendering of dynamic scenes while focusing less on geometry reconstruction.

Another line of approaches attempts to reconstruct dynamic scenes from monocular depth estimation and optical flow. Robust-CVD proposes to first estimate the depth of each frame and then optimize camera poses and depth jointly [12]. CasualSAM leverages optical flow to train a depth and movement prediction network and optimize camera parameters [56]. MegaSaM proposes an optimization framework for dynamic scene reconstruction based on DROID-SLAM [14, 38]. Recently, DUS3R [44] provides a powerful foundation model for static scene reconstruction in point cloud representation. Based on DUS3R, better methods come out recently. MonST3R fine-tunes DUS3R on dynamic scene datasets to recover dynamic 3D scenes [55]. CUT3R introduces a continuous 3D perception model that can perform online dense 3D reconstruction [42]. Easi3R identifies moving objects in the video from DUS3R’s attention maps [1]. Aether proposes a unified model for reconstruction, planning, and prediction on dynamic scene [36]. Our coarse prediction pipeline directly benefits from MonST3R [55], and our refinement framework is heavily inspired by the optimization framework in CasualSAM [56] and Mega-SaM [14].

Another important task in dynamic scene understanding is segmenting the moving map, images indicating which pixels are moving in the scene at each frame, in the video. Recent approaches are

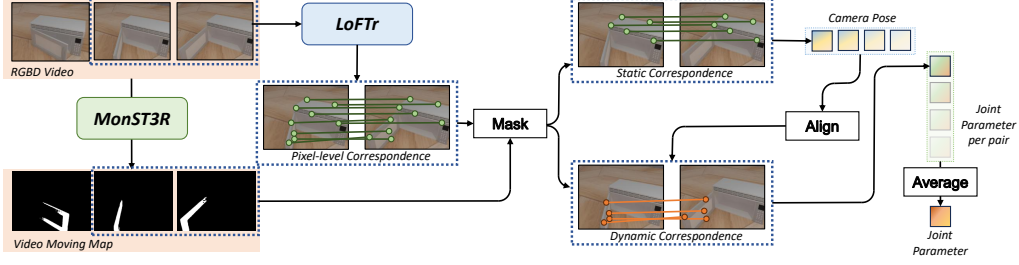


Figure 2: An overview of our coarse prediction pipeline. We first use feature matching in the static regions to estimate relative camera poses and align all observations to the same coordinate. Then, we compute the rigid transformation using feature matching in the dynamic regions to estimate joint parameters. Finally, we average out all the results to produce a joint parameter estimation.

mainly based on optical flow or point tracking [52, 24, 23, 9, 5, 7]. However, we find that optical flow or point tracking-based methods struggle with textureless objects and are sensitive to the points’ location. Therefore, we don’t use them in our work. We further discuss this in Appendix C.5.

3 Method

In this section, we first state the problem formulation. Then, we introduce our coarse-to-fine reconstruction pipeline in the next two subsections.

3.1 Problem formulation

In this paper, we proposed to reconstruct articulated object from a casually capture RGBD video $\mathcal{V} = \{\mathcal{I}_t\}_{t=0}^T$, where T denotes the video length and each frame $\mathcal{I}_t \in \mathbb{R}^{H \times W \times 4}$ represent an RGBD image with height H and width W . To achieve accurate reconstruction of an articulated object, it is essential to estimate two key components: 1) The joint parameters $\mathcal{J} \in \mathbb{R}^7$, including the type of joint (e.g., revolute or prismatic), as well as the position and orientation of the joint axis. 2) The movable parts of the object. Since it’s challenging to segment movable parts on the 3D objects directly, we instead estimate the moving map of each video frame $\mathcal{M}_t = \{0, 1\} \in \mathbb{R}^{H \times W}$, which indicates the moving parts of the object in the video. Then, we reproject the moving map to 3D space with depth values to identify movable parts of the object.

For the input video \mathcal{V} , we assume it is casually captured using handheld devices such as mobile phones. Therefore, we do not assume that the camera remains stationary during recording, nor do we assume that the camera trajectory can be easily obtained. Both camera pose $p_t^{cam} \in \mathbb{R}^{4 \times 4}$ and the joint state $s_t \in \mathbb{R}$ at each frame change simultaneously, which also needs to be estimated during the reconstruction. Normally, a human operator manipulates one joint at a time. Thus, we assume only one part of the object is moving in the video.

3.2 Coarse prediction

To address this issue, we first design a coarse prediction pipeline as shown in Fig. 2. The idea is to decouple the problem by firstly finding the moving part \mathcal{M}_t in the video \mathcal{V} . Once we identify the dynamic region and static region in the video frames, we can use point motion in the static region to estimate the camera pose p_t^{cam} at each frame. We align all the observations to the same coordinate, and compute rigid transformations of movable parts of the object using points motion in the dynamic region to estimate joint parameters \mathcal{J} and joint state $\{s_t\}_{t=0}^T$.

In practice, we use MonST3R [55] to provide a rough moving map \mathcal{M}_t of each video frame. Then we reproject pixels in \mathcal{I}_t to 3D point cloud \mathcal{P}_t with camera intrinsics, and utilize LoFTr [35] to compute pixel-level feature matching between two video frames. We use feature matching in static regions identified by \mathcal{M}_t to compute relative camera pose. Then, we align all observations to the same camera coordinate with the relative camera poses of each frame. Then, we use feature matching in the dynamic regions to compute a rigid transformation of the movable parts of the object. We can

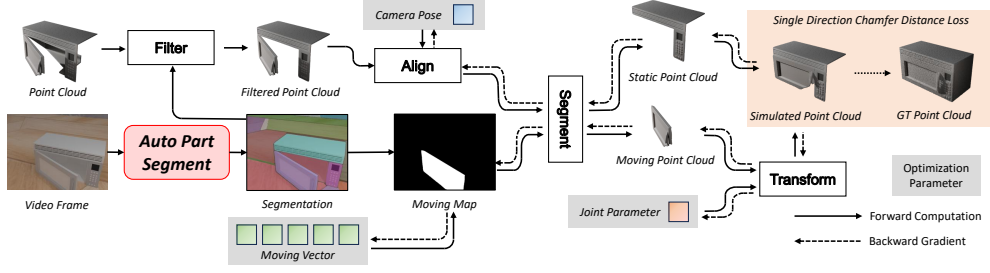


Figure 3: An overview of our refinement pipeline. We transform observations in the video back to the initial stage with camera poses and joint parameters. We then compute the chamfer distance from the transformed observation to the object surface as a loss function and optimize relevant parameters.

further estimate joint parameters from this rigid transformation. We estimate joint parameters from multiple video frame pairs and average the results as the coarse prediction of the joint parameters and states.

Through the coarse prediction stage, the proposed pipeline obtains rough results on moving map segmentation and joint parameters. However, this method has two disadvantages: 1) Since the moving map is inaccurate, camera poses and joint parameters estimation are inaccurate. 2) Feature matching provides a reliable prediction only when two video frames are visually similar. In other words, the joint states do not change much, and the point motion in dynamic regions will be minimal. In this situation, it's hard to tell which joint types lead to this kind of point motion. Joint type prediction will be inaccurate as well.

3.3 Refinement

We further refine the estimation results through gradient-based optimization. It is based on a simple idea: if the estimation of the joint parameters and camera poses is accurate, the object states observed in the video should be able to transform back perfectly to the initial state. In other words, the transformed point cloud should align with the given object surface point cloud at the initial state. An overview of our pipeline is shown in Fig. 3.

Concretely, we divide the optimization target into static part and moving part. For the static part, we transform the point cloud \mathcal{P}_t to the same coordinate with camera poses p_t^{cam} . We compute the **single directional chamfer distance** to the given object surface point cloud \mathcal{P}^O in Eq. (1).

$$\mathcal{L}_{\text{static}} = \frac{1}{T} \sum_{t=1}^T (1 - \mathcal{M}_t) \cdot \text{Chamfer}(p_t^{cam} \mathcal{P}_t, \mathcal{P}^O) \quad (1)$$

To compute the chamfer distance of the moving part, we first compute the transformation matrix k_t for the moving part with \mathcal{J} and s_t . Then, we transform \mathcal{P}_t with camera pose p_t^{cam} and k_t . We compute the **single directional chamfer distance** to \mathcal{P}^O in Eq. (2).

$$\mathcal{L}_{\text{dynamic}} = \frac{1}{T} \sum_{t=1}^T \mathcal{M}_t \cdot \text{Chamfer}(k_t p_t^{cam} \mathcal{P}_t, \mathcal{P}^O) \quad (2)$$

Combining these two equations produces the final optimization objective $\mathcal{L} = \mathcal{L}_{\text{static}} + \mathcal{L}_{\text{dynamic}}$. Note that this computation graph is differentiable. Thus, we can simply use gradient-based optimization methods to optimize joint parameters, camera poses, and the moving map.

However, this implementation has two problems: 1) how to deal with the newly observed part of the object in the video. For example, the interior of a microwave cannot be observed when the door is closed at the beginning. However, it will be observed during the interaction with the door. This interior part does not correspond with the object surface point cloud. Consequently, chamfer distance will provide an inaccurate estimation of point cloud alignment, resulting in biased gradients. 2) how

to optimize \mathcal{M}_t efficiently. In the previous formulation, we optimize the probability of each pixel belonging to the moving part of each frame, which contains $T \times H \times W$ parameters to optimize.

The above two problems share a similar property: both of them care about part-level information. The first one cares about whether a **part** is newly observed, while the second one cares about which **parts** are moving. If we can automatically segment the whole video into different parts and track them, we can focus on which parts are newly observed and which are moving. In this case, the number of parts is much smaller than the number of pixels, reducing the optimization complexity.

In practice, we first find the parts that appear in the initial frame and then filter out parts in the following frames that do not appear in the initial frame. This resolves the false correspondence issue when computing the chamfer distance. For the second problem, instead of optimizing \mathcal{M}_t on each pixel, we optimize \mathcal{M}_t of each segmented part. We define a vector $v \in (0, 1)^{D \times 1}$ to represent the probability of each part being a moving part, where D is the number of segmented parts. Then, for each part i , we can compute the probability of being a moving part by multiplying v_i and the mask of this part $E_t^i \in \{0, 1\}^{H \times W}$. We can construct \mathcal{M}_t simply by aggregating the mask of each part $\mathcal{M}_t = \sum_i^D v_i \cdot E_t^i$. We adopt an existing implementation of automatic part segmentation on videos. Finally, we find movable parts on \mathcal{P}^O via finding points in the dynamic regions in \mathcal{M}_0 . We adopt NKSR [6] to reconstruct meshes.

4 Experiments

We aim to answer the following questions with our experiments: 1) How well can we estimate joint parameters? 2) How well can we reconstruct the geometry of the articulated object? 3) How do different modules and strategies of our pipeline contribute to the final results? and 4) Can our method work in real-world scenarios?

4.1 Experiment setup

Dataset At present, there is no benchmark for evaluating joint parameter estimation from dynamic RGBD videos. Therefore, we construct our synthetic dataset for evaluation. We choose objects from 11 categories in the PartNet-Mobility dataset [25]. We load them into the SAPIEN simulator [49] to simulate interaction between humans and objects. For each interaction, we generate two videos shot from two different viewpoints. To generate camera motion, every 7 - 10 timesteps in the simulator, the camera will move to a random target pose. The target pose is sampled from a Gaussian distribution, where the mean value is the current camera pose and the standard deviation is set to ensure the camera can consistently look at the object. For the object surface point cloud, we simply fuse 24 views' point clouds of the object. For Robot-See-Robot-Do, we generate 100 views because it requires to train NeRF and 3D gaussian splatting [10]. Please refer to Appendix B for more details about our dataset. Since Robot-See-Robot-Do requires manually cropping and clustering 3D gaussian splatting [10], which is labor-intensive, we split the dataset into two parts. We randomly sampled 10% of test videos from the whole dataset to construct a smaller dataset named S-Dataset, containing 73 test videos, while L-Dataset contains the remaining videos in the original dataset.

Baselines We choose two very recent works on modeling and reconstructing articulated objects that take videos as input:

- Robot-See-Robot-Do (RSRD): The authors introduce a differentiable rendering pipeline to recover 3D part motion from human demonstration videos [10]. We follow their pipeline first to reconstruct and segment the 3D gaussian splatting of the object. Then, input a video to estimate the SE3 transformation of each part of 3D gaussians at each frame via differentiable rendering. Since their pipeline does not include geometry reconstruction module, we export 3D gaussians from their pipeline for geometry reconstruction evaluation for this method.
- Articulate Anything: The authors introduce a system to retrieve object mesh from a library and then estimate link and joint parameters from text, image, or video input [13]. We follow their setting to use the Partnet-Mobility dataset as the object mesh library, which contains all the ground truth object meshes for our test cases. For VLM, we also follow their setting to use Gemini Flash-1.5 model [37].

Table 1: Quantitative evaluation on S-Dataset. We report the mean and standard deviation across all the test cases within the dataset. Our method performs best on most metrics.

Joint	Methods	Axis(rad)↓	Position(m)↓	Type(%)↓	State(rad/m)↓	Failure(%)↓
Revolute	Ours	0.32±0.56	0.13±0.25	15.90	0.25±0.46	6.81
	Articulate Anything	0.82±0.79	0.81±0.40	40.90	N.A.	38.63
	RSRD	1.17±0.51	2.03±7.48	88.63	1.03±0.60	47.72
	Ours w/o Refine	1.03±0.67	0.24±0.25	68.18	0.59±0.47	4.54
	Ours w/o Coarse	1.26±0.29	0.34±0.26	77.27	0.69±0.45	2.27
	Ours w/o Segment	0.36±0.55	0.17±0.24	18.18	0.33±0.44	6.81
Prismatic	Ours	0.24±0.33	N.A.	10.34	0.08±0.22	0
	Articulate Anything	0.92±0.78	N.A.	50.00	N.A.	46.66
	RSRD	1.22±0.42	N.A.	41.37	0.70±0.48	37.93
	Ours w/o Refine	0.28±0.40	N.A.	6.89	0.16±0.16	0
	Ours w/o Coarse	0.42±0.11	N.A.	6.89	0.08±0.09	0
	Ours w/o Segment	0.30±0.41	N.A.	13.79	0.07±0.07	0

Metrics For articulated object reconstruction, we evaluate the following metrics: 1) Joint parameters, including joint type classification error in percentage, joint axis error in randian, joint position error for revolute joint in meter, and joint state error in randian for revolute joint and meter for prismatic joint. 2) Chamfer distance on the whole object, movable part of the object, and static part, denoted as CD-w, CD-m, CD-s, respectively. If the method crashes or retrieves the wrong objects, we classify these cases as prediction failures and assign $\frac{\pi}{2}$ for joint axis and state of revolute joint error, and 1 for the other metrics. Due to the page limit, we put more results and discussions in Appendix C

4.2 Joint estimation results

We first comprehensively evaluate our method and two baselines on the S-Dataset. The quantitative results on S-Dataset are in Tab. 1, and qualitative results are in Fig. 4. Results on L-Dataset can be found in Appendix C.1. Our proposed method is comparable and superior in most metrics to the current state-of-the-art method on our dataset. In our dataset, Articulate Anything struggles with retrieving the correct

object meshes. Most failure cases in Tab. 1 are due to retrieval error, even though the library contains all the ground truth object meshes. For example, in Fig. 4 we find that Articulate Anything retrieves a lamp in the Stapler test case. Besides, the link placement and affordance detection modules are not robust either. In Fig. 4, Articulate Anything selects the wheel of the table to be the movable part in the Table test case, and places the lid of the USB below the main body in the USB test case. For RSRD, we spot 3D gaussian splatting in their pipeline cannot represent objects with less texture information accurately. Therefore, it often fails to crop and cluster 3D gaussians correctly, though human operators participate in this step. Besides, their pipeline cannot handle camera motion. Thus, its prediction deviates from the ground truth a lot as shown in Fig. 4.

4.3 Geometric reconstruction results

We evaluate the geometric reconstruction results with the Chamfer distance. Since mesh is the target geometry representation of the articulated objects, we sample 10000 points from the reconstructed mesh and the ground truth mesh, and compute the Chamfer distance between them. Tab. 2 shows the

Table 2: Geometry reconstruction evaluation on S-Dataset. We report the mean and standard deviation across test cases within the dataset. Our method achieves the lowest reconstruction error among all aspects.

Methods	CD-w↓	CD-m↓	CD-s↓
Ours	0.01±0.01	0.13±0.26	0.06±0.19
Articulate Anything	0.11±0.22	0.59±0.73	0.07±0.18
RSRD	3.39±21.47	0.82±1.17	0.14±0.41
Ours w/o Refine	0.01±0.01	0.40±0.44	0.35±0.46
Ours w/o Coarse	0.01±0.01	0.19±0.22	0.06±0.16
Ours w/o Segment	0.01±0.01	0.40±0.44	0.35±0.46

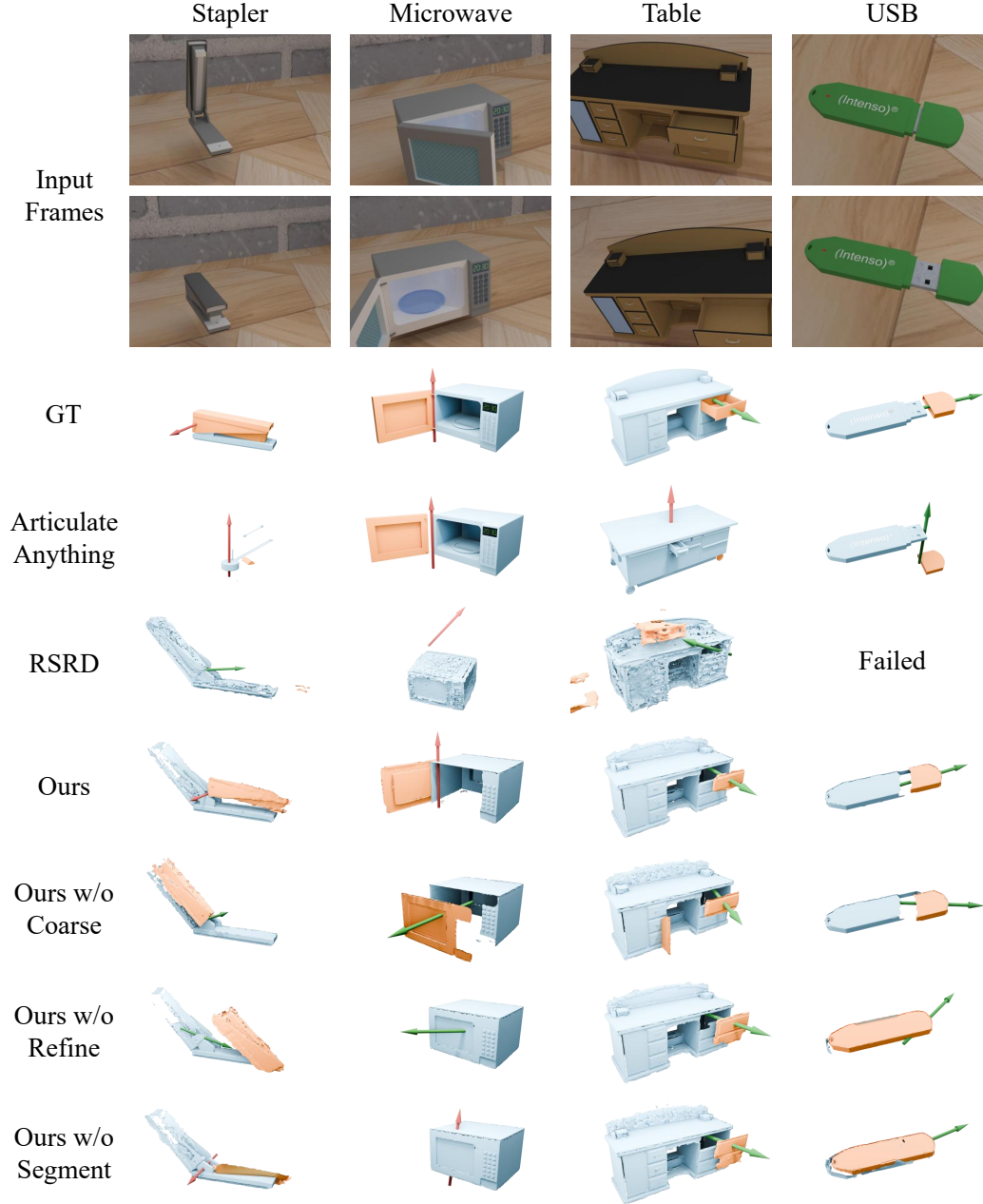


Figure 4: Qualitative results of joint prediction. For the object, we annotate the moving parts to be orange and static parts to be blue. For the joint axis, we annotate the prismatic joints to be green and revolute joints to be red. To illustrate the joint state prediction results, we render the articulated object at the end state.

results of geometry reconstruction on S-Dataset. Results on L-Dataset can be found in Appendix C.1. Due to the retrieval error, link placement error, and affordance detection error, Articulate Anything has a high reconstruction error, particularly on movable parts. RSRD performs even worse on geometry reconstruction since it struggles with clustering 3D gaussian splatting into different parts.

4.4 Ablation study

We design three ablations of our method. To demonstrate the importance of the coarse prediction module, we remove the coarse prediction module from our pipeline and randomly initialize the parameters of the refinement module, named **Ours w/o Coarse**. To illustrate the importance of the

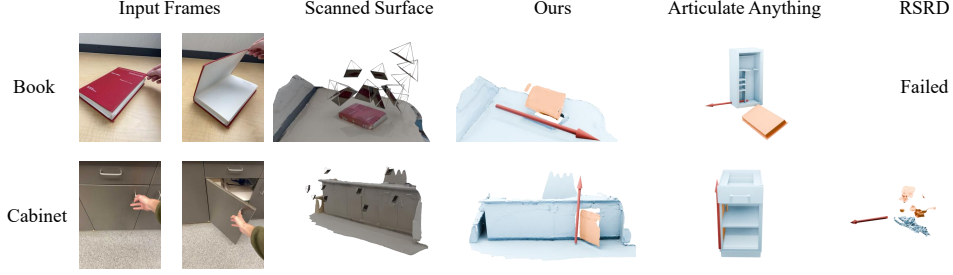


Figure 5: Qualitative results on real data.

refinement module, we remove the refinement module and evaluate coarse prediction results directly, named **Ours w/o Refine**. Lastly, to demonstrate the advantage of the automatic part segmentation strategy for the refinement process, we naively optimize the probability of being a moving part for each pixel at each video frame, named **Ours w/o Segment**.

The joint parameter prediction results are shown in Tab. 1. Without refinement, coarse prediction will fail on joint type prediction for more than half of the test cases, particularly on revolute joints. As we mentioned in Sec. 3, when two video frames are similar, it's hard to distinguish revolute joint and prismatic joint. In Fig. 4, we can also find that for Stapler and Microwave test cases, the coarse prediction module misclassifies these two test cases. Besides, without the refinement module, it's much harder to correctly segment the object's movable parts and static parts, as shown in Tab. 2. Without coarse prediction, refinement cannot optimize effectively with random initialization, as shown quantitatively in Tab. 2 and Tab. 1 and qualitatively in Fig. 4. Without automatic part segmentation, the joint prediction performance degrades as the refinement module suffers from the incorrect point correspondence issue mentioned in Sec. 3. For instance, it fails on the Microwave test case in Fig. 4. Besides, the segmentation also deviates from the correct segmentation, as shown in Tab. 2. In Fig. 4, we can also find that without automatic part segmentation, our method has difficulty segmenting the moving parts and static parts accurately. This illustrates the importance of automatic part segmentation in optimizing joint parameters and moving map of the video efficiently.

4.5 Real-world qualitative evaluation

We also evaluate our methods and baselines on real data. We use an iPhone 12 Pro with LiDAR camera to capture real data. We first use Polycam [28] to reconstruct the object's surface, and then use Record3D [31] to record an RGB-D video of human interaction with the object. To further improve the depth map quality, we adopt Prompt Depth Anything [15] to scale up the original depth map. Real videos are likely to include human hands, which are other moving objects in the scene. This will cause confusion to MonST3R when predicting the moving map. Thus, we use Grounded SAM 2 [32, 33] to mask hands with simple text prompts: "hands and arms".

From the qualitative results shown in Fig. 5, Articulate Anything fails on the Book test case due to the lack of similar meshes in the mesh library. In the Cabinet test case, Articulate Anything retrieves reasonable meshes but predicts the wrong joint rotation direction in the Cabinet test case, as the door rolls inside the cabinet. RSRD does not work in either case. This demonstrates our method can work on real data and outperform current methods. We put more qualitative results in Appendix C.7.

5 Conclusion

We introduced a new problem setting for reconstructing articulated objects from casually captured RGBD videos. Our problem setting is more practical than prior works, bringing new challenges to current methods. We then developed a coarse-to-fine framework for this problem. To evaluate, we proposed a new dataset that contains $20\times$ more objects than previous benchmarks. Our experiments show our method outperforms baselines, providing a practical and robust approach to reconstructing articulated objects. Nonetheless, there is still space for improvement. Better dynamic scene understanding approaches and video segmentation models can help improve our method. In addition, our pipeline does not reconstruct the interior of the object. Incrementally reconstructing the interior of the object is another future direction.

Acknowledgments and Disclosure of Funding

This work was funded in part by a Canada Research Chair, NSERC Discovery Grant, and enabled by support from the Digital Research Alliance of Canada. The authors would like to thank Jiayi Liu, Xingguang Yan, Austin T. Wang, Hou In Ivan Tam, Morteza Badali for valuable discussions, and Yi Shi for proofreading.

References

- [1] Xingyu Chen, Yue Chen, Yuliang Xiu, Andreas Geiger, and Anpei Chen. Easi3r: Estimating disentangled motion from dust3r without training. *arXiv preprint arXiv:2503.24391*, 2025.
- [2] 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. *arXiv preprint arXiv:2405.11656*, 2024.
- [3] Yuanxing Duan, Fangyin Wei, Qiyu Dai, Yuhang He, Wenzheng Chen, and Baoquan Chen. 4d-rotor gaussian splatting: Towards efficient novel-view synthesis for dynamic scenes. In *Proc. SIGGRAPH*, July 2024.
- [4] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. ISSN 0001-0782. doi: 10.1145/358669.358692. URL <https://doi.org/10.1145/358669.358692>.
- [5] Lily Goli, Sara Sabour, Mark Matthews, Brubaker Marcus, Dmitry Lagun, Alec Jacobson, David J. Fleet, Saurabh Saxena, and Andrea Tagliasacchi. RoMo: Robust motion segmentation improves structure from motion. *arXiv:2411.18650*, 2024.
- [6] Jiahui Huang, Zan Gojcic, Matan Atzmon, Or Litany, Sanja Fidler, and Francis Williams. Neural kernel surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4369–4379, 2023.
- [7] Nan Huang, Wenzhao Zheng, Chenfeng Xu, Kurt Keutzer, Shanghang Zhang, Angjoo Kanazawa, and Qianqian Wang. Segment any motion in videos, 2025. URL <https://arxiv.org/abs/2503.22268>.
- [8] Zhenyu Jiang, Cheng-Chun Hsu, and Yuke Zhu. Ditto: Building digital twins of articulated objects from interaction. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [9] Laurynas Karazija, Iro Laina, Christian Rupprecht, and Andrea Vedaldi. Learning segmentation from point trajectories. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- [10] 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 *8th Annual Conference on Robot Learning*, 2024. URL <https://openreview.net/forum?id=2LLu3gavF1>.
- [11] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [12] Johannes Kopf, Xuejian Rong, and Jia-Bin Huang. Robust consistent video depth estimation. In *CVPR*, 2021.
- [13] 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. *arXiv preprint arXiv:2410.13882*, 2024.
- [14] Zhengqi Li, Richard Tucker, Forrester Cole, Qianqian Wang, Linyi Jin, Vickie Ye, Angjoo Kanazawa, Aleksander Holynski, and Noah Snavely. MegaSaM: Accurate, fast and robust structure and motion from casual dynamic videos. *arXiv preprint*, 2024.
- [15] Haotong Lin, Sida Peng, Jingxiao Chen, Songyou Peng, Jiaming Sun, Minghuan Liu, Hujun Bao, Jiashi Feng, Xiaowei Zhou, and Bingyi Kang. Prompting depth anything for 4k resolution accurate metric depth estimation. *arXiv*, 2024.
- [16] Jiayi Liu, Ali Mahdavi-Amiri, and Manolis Savva. PARIS: Part-level reconstruction and motion analysis for articulated objects. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2023.

- [17] Jiayi Liu, Denys Iliash, Angel X. Chang, Manolis Savva, and Ali Mahdavi-Amiri. SINGAPO: Single image controlled generation of articulated parts in object. *arXiv preprint arXiv:2410.16499*, 2024.
- [18] Minghua Liu, Mikaela Angelina Uy, Donglai Xiang, Hao Su, Sanja Fidler, Nicholas Sharp, and Jun Gao. Partfield: Learning 3d feature fields for part segmentation and beyond, 2025.
- [19] 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 *The Thirteenth International Conference on Learning Representations*, 2025.
- [20] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024.
- [21] Jun Lv, Qiaojun Yu, Lin Shao, Wenhai Liu, Wenqiang Xu, and Cewu Lu. Sagci-system: Towards sample-efficient, generalizable, and incremental robot learning. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 98–105. IEEE, 2022.
- [22] Zhao Mandi, Yijia Weng, Dominik Bauer, and Shuran Song. Real2code: Reconstruct articulated objects via code generation. *arXiv preprint arXiv:2406.08474*, 2024.
- [23] Etienne Meunier and Patrick Bouthemy. Unsupervised space-time network for temporally-consistent segmentation of multiple motions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22139–22148, 2023.
- [24] Etienne Meunier, Anaïs Badoual, and Patrick Bouthemy. Em-driven unsupervised learning for efficient motion segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4462–4473, 2023. doi: 10.1109/TPAMI.2022.3198480.
- [25] Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [26] Jiteng Mu, Weichao Qiu, Adam Kortylewski, Alan Yuille, Nuno Vasconcelos, and Xiaolong Wang. A-sdf: Learning disentangled signed distance functions for articulated shape representation. In *ICCV*, pages 12981–12991, 2021.
- [27] Neil Nie, Samir Yitzhak Gadre, Kiana Ehsani, and Shuran Song. Structure from action: Learning interactions for 3d articulated object structure discovery. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1222–1229. IEEE, 2023.
- [28] Polycam. Polycam, 2025. URL <https://poly.cam/>.
- [29] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.
- [30] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. URL <https://arxiv.org/abs/2408.00714>.
- [31] Record3D. Record3d, 2025. URL <https://record3d.app/>.
- [32] Tianhe Ren and Shuo Shen. Grounded SAM 2: Ground and Track Anything in Videos, 2025. URL <https://github.com/IDEA-Research/Grounded-SAM-2>.
- [33] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, Zhaoyang Zeng, Hao Zhang, Feng Li, Jie Yang, Hongyang Li, Qing Jiang, and Lei Zhang. Grounded sam: Assembling open-world models for diverse visual tasks, 2024.
- [34] Chaoyue Song, Jiacheng Wei, Chuan Sheng Foo, Guosheng Lin, and Fayao Liu. Reacto: Reconstructing articulated objects from a single video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5384–5395, 2024.
- [35] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with transformers. *CVPR*, 2021.

- [36] Aether Team, Haoyi Zhu, Yifan Wang, Jianjun Zhou, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Chunhua Shen, Jiangmiao Pang, and Tong He. Aether: Geometric-aware unified world modeling. *arXiv preprint arXiv:2503.18945*, 2025.
- [37] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.
- [38] Zachary Teed and Jia Deng. DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras. *Advances in neural information processing systems*, 2021.
- [39] Marcel Torne, Anthony Simeonov, Zechu Li, April Chan, Tao Chen, Abhishek Gupta, and Pulkit Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation. *Arxiv*, 2024.
- [40] Wei-Cheng Tseng, Hung-Ju Liao, Lin Yen-Chen, and Min Sun. Cla-nerf: Category-level articulated neural radiance field. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8454–8460. IEEE, 2022.
- [41] Qianqian Wang, Vickie Ye, Hang Gao, Weijia Zeng, Jake Austin, Zhengqi Li, and Angjoo Kanazawa. Shape of motion: 4d reconstruction from a single video. *arXiv preprint arXiv:2407.13764*, 2024.
- [42] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A. Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state, 2025.
- [43] Shizun Wang, Xingyi Yang, QiuHong Shen, Zhenxiang Jiang, and Xinchao Wang. Gflow: Recovering 4d world from monocular video. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 7862–7870, 2025.
- [44] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *CVPR*, 2024.
- [45] Xi Wang, Tianxing Chen, Qiaojun Yu, Tianling Xu, Zanzin Chen, Yiting Fu, Cewu Lu, Yao Mu, and Ping Luo. Articulated object manipulation using online axis estimation with sam2-based tracking. *arXiv preprint arXiv:2409.16287*, 2024.
- [46] Fangyin Wei, Rohan Chabra, Lingni Ma, Christoph Lassner, Michael Zollhöfer, Szymon Rusinkiewicz, Chris Sweeney, Richard Newcombe, and Mira Slavcheva. Self-supervised neural articulated shape and appearance models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15816–15826, 2022.
- [47] Yijia Weng, Bowen Wen, Jonathan Tremblay, Valts Blukis, Dieter Fox, Leonidas Guibas, and Stan Birchfield. Neural implicit representation for building digital twins of unknown articulated objects. In *CVPR*, 2024.
- [48] GuanJun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20310–20320, June 2024.
- [49] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [50] Kai Xu, Tze Ho Elden Tse, Jizong Peng, and Angela Yao. Das3r: Dynamics-aware gaussian splatting for static scene reconstruction. *arXiv preprint arxiv:2412.19584*, 2024.
- [51] Zihao Yan, Fubao Su, Mingyang Wang, Ruizhen Hu, Hao Zhang, and Hui Huang. Interaction-driven active 3d reconstruction with object interiors. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 42(6):249:1–249:12, 2023.
- [52] Charig Yang, Hala Lamdouar, Erika Lu, Andrew Zisserman, and Weidi Xie. Self-supervised video object segmentation by motion grouping. In *ICCV*, 2021.
- [53] Yunhan Yang, Yukun Huang, Yuan-Chen Guo, Liangjun Lu, Xiaoyang Wu, Edmund Y Lam, Yan-Pei Cao, and Xihui Liu. Sampart3d: Segment any part in 3d objects. *arXiv preprint arXiv:2411.07184*, 2024.

- [54] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. In *International Conference on Learning Representations (ICLR)*, 2024.
- [55] 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. *arXiv preprint arxiv:2410.03825*, 2024.
- [56] Zhoutong Zhang, Forrester Cole, Zhengqi Li, Michael Rubinstein, Noah Snavely, and William T. Freeman. Structure and motion from casual videos. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 20–37, Cham, 2022. Springer Nature Switzerland. ISBN 978-3-031-19827-4.

Appendix

A Implementation details

In this section we introduce more details about implementation of our method and baselines.

A.1 Method

Coarse prediction Since the input video usually contains more than 100 frames, we sample the input video to around 20 frames. We first send these sampled frames to MonST3R [55] to compute the moving map of each frame. Then, we select pairs of frames to compute the feature matching using LoFTr [35]. Not all feature matches are accurate, so we only consider feature matches that have a confidence value larger than 0.95 to be valid. We first use feature matching within the static regions to compute the relative camera poses $p_{t,t-1}^{cam}$ from frame t to $t - 1$ sequentially. We use these relative camera poses to align all the frames to the first camera coordinate. Then, we proceed to estimate joint parameters. We select frame pairs within three intervals to balance the accuracy of feature matching and joint prediction. Due to the inaccuracy of moving map and feature matching, we filter out frame pairs that contain fewer than 80 valid feature matches. For the rest of the frame pairs, we estimate joint parameters for the revolute joint and the prismatic joint separately. For each joint type, we run RANSAC [4] for 50 iterations to select the joint parameters estimation that best fits the observation. For each frame pair, we also compare the distances between corresponding feature points under two sets of joint parameters. If the distance for the revolute joint is lower, this frame pair votes for revolute, and vice versa. After estimating joint parameters for all the frame pairs, we compute the mean value for the joint axis and position for both joint types. We choose the joint type that gets the most votes as the predicted joint type for coarse prediction.

Refinement Though we predict the joint type in coarse prediction, we still retain the prediction results of both joint types from coarse prediction and initialize the refinement module with those values. For moving vector initialization, we compute the intersection area between each part and the moving map from MonST3R [55], and then normalize it by the area of each part. During the optimization, we compute the optimization objective according to Eqs. (1) and (2). But computing the chamfer distance will be very slow if the number of points is very high. Therefore, for each frame, we randomly sample 50% of points from the whole observation to compute the Chamfer distance. The gradient will not be affected since we are optimizing the moving vector instead of the probability of being a moving part of each pixel at each frame. This strategy significantly improves the optimization speed without losing prediction accuracy. We concurrently optimize the joint prediction for both joint types for 400 iterations, using Adam optimizer [11] and 5e-3 learning rate. After optimization, we need to choose the joint type. We first classify the test case as a prismatic joint if the distance between the object surface point cloud and the joint axis is larger than 0.1 meters. For the remaining cases, we compare the lowest chamfer distance of these two cases and select the joint type that has the lower chamfer distance. These are the final joint type prediction results for our method. Finally, we find pixels in the moving map whose value is larger than 0.7 to be the moving points. We project these points back to 3D and find points in the object surface point cloud whose distance to the moving points is less than 0.03 meters to be the movable parts of the object. The other points in the object surface point cloud are classified as static parts of the object. We reconstruct meshes using NKSR [6].

Our method requires a GPU with at least 24GB of CUDA memory. For one test video, our coarse prediction module needs around 2 minutes to run MonST3R to get moving map of the video and another 30 seconds to estimate joint parameters, using RTX4090 GPU. For the refinement, we first need to run automatic part segmentation on the test video, which takes 4-8 minutes. Then, we need to run optimization for the two joint types separately. Thus, we recommend using multiple GPUs for this stage. For one joint type, our optimization needs 3-5 minutes to run on an RTX4090 GPU, or 8-15 minutes on an RTX A5000 GPU. Finally, for mesh reconstruction, to get better reconstruction quality, we use an RTX A6000 GPU to reconstruct the mesh, since it has 48GB of CUDA memory to allow more points for reconstruction.

Table B.1: Dataset configuration.

Category	#Object	Joint Type	#Video(revolute, prismatic)	# Average Frames
WashingMachine	17	revolute	34 (34, 0)	108.35
USB	30	revolute, prismatic	60 (24, 36)	149.5
StorageFurniture	30	revolute, prismatic	120 (64, 56)	122.8
Stapler	8	revolute	20 (20, 0)	118.9
Scissors	30	revolute	60 (60, 0)	91.56
Refrigerator	25	revolute	76 (76, 0)	97.18
Microwave	16	revolute	32(32, 0)	134.31
Laptop	30	revolute	60 (60, 0)	126.03
Dishwasher	29	revolute	58 (58, 0)	136.93
Box	19	revolute	50 (50, 0)	129.08
Table	50	revolute, prismatic	214 (32, 182)	123.81
Total	284	revolute, prismatic	784 (510, 274)	121.68

A.2 Baselines

Robot-See-Robot-Do We follow the original pipeline of RSRD to run the experiments on our dataset. RSRD will estimate the motion of each part at each frame. Based on the part motion, we select the motion of the part that has the largest translation distance throughout the video to compute the joint parameters. For geometry reconstruction, we export the gaussian splats from the dig model in RSRD pipeline [10] and build mesh using NKSr [6].

Articulate Anything We follow the original pipeline of Articulate Anything to run the experiments on our dataset. Articulate Anything will retrieve top-k objects from the Partnet-Mobility dataset based on the input video. Then, it selects the best one from top-k candidates [13]. When evaluating its results, we consider them to be a failure if the correct object does not lie in the retrieved top-k candidates.

B Dataset details

We build our dataset in the SAPIEN simulator [49]. We design a small cuboid space with a simple texture as the environment. Then, we place the object in that space and render videos while manipulating the object using the built-in controller. Fig. B.1 illustrates the data collection environment. We build a dataset for robust evaluation of our method in Tab. B.1.

C Additional evaluation and discussion

C.1 Joint estimation and geometry reconstruction results on L-Dataset

We conduct a more extensive evaluation of methods, excluding RSRD, on the L-Dataset. Quantitative results on joint estimation and geometry reconstruction are shown in Tab. C.2 and Tab. C.3. Similar to the results in S-Dataset, our method outperforms Articulate Anything and other ablated versions consistently on L-Dataset.

C.2 Variance of joint estimation results

To further evaluate the stability of our method, we conduct experiments on the S-Dataset with 10 different seeds. The mean and standard deviation of the final results are shown in Tab. C.4. We can find that our method consistently outperforms the ablated version of our method. Besides, by comparing our results and ours w/o Refinement results, we can find that the refinement module can reduce the variance of the prediction.

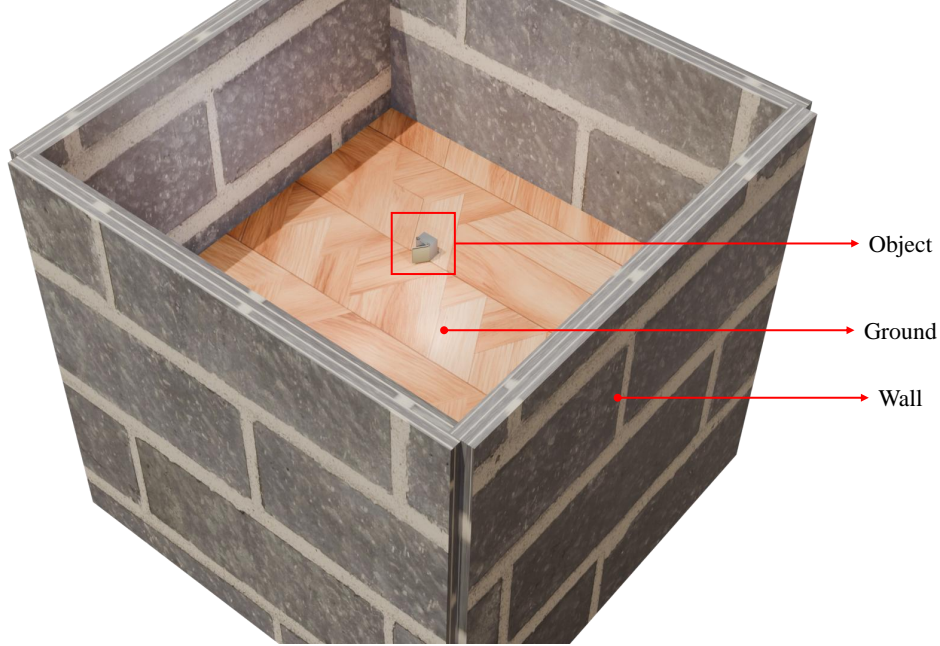


Figure B.1: This is an illustration of the scene we use to generate our dataset.

Table C.2: Quantitative evaluation on L-Dataset. We report the mean and standard deviation across all the test cases within the dataset. Our method achieves the lowest prediction error on most metrics.

Joint	Methods	Axis(rad)↓	Position(m)↓	Type(%)↓	State(rad/m)↓	Failure(%)↓
Revolute	Ours	0.29±0.51	0.12±0.22	14.80	0.27±0.46	3.43
	Articulate Anything	0.73±0.77	0.73±0.40	37.12	N.A.	35.19
	Ours w/o Refine	1.09±0.62	0.24±0.33	73.60	0.59±0.51	2.57
	Ours w/o Coarse	1.30±0.25	0.34±0.23	76.82	0.68±0.44	1.07
	Ours w/o Segment	0.49±0.62	0.20±0.22	26.60	0.38±0.45	3.21
Prismatic	Ours	0.27±0.40	N.A.	8.16	0.06±0.15	2.44
	Articulate Anything	0.83±0.78	N.A.	49.79	N.A.	44.89
	Ours w/o Refine	0.27±0.44	N.A.	8.57	0.10±0.08	0.40
	Ours w/o Coarse	0.51±0.20	N.A.	14.69	0.12±0.15	1.63
	Ours w/o Segment	0.29±0.42	N.A.	9.79	0.08±0.15	2.44

C.3 Moving map segmentation results

Moving map segmentation is an important intermediate results of our method. We evaluate moving map segmentation using mIOU metric. From the quantitative results in Tab. C.5, we find our method achieves comparable results to MonST3R [55], though our pipeline ignores the inner parts of the object, which could be moving part in the video. Without coarse prediction as initialization, our method is hard to optimize for very accurate moving map segmentation. Similarly, without automatic mask generation, our pipeline cannot optimize moving map of the video efficiently, resulting in worse moving map segmentation of the video.

Table C.5: Evaluation on moving map.

Methods	mIOU↑
Ours	0.59653
Ours w/o Refine (MonST3R)	0.61632
Ours w/o Coarse	0.42545
Ours w/o Segment	0.40310

Table C.3: Quantitative evaluation on geometry reconstruction on L-Dataset. We report the mean and standard deviation across all the test cases within the dataset. Our model is generally comparable to and superior to the baseline and ablated versions.

Methods	CD-w↓	CD-m↓	CD-s↓
Ours	0.01±0.03	0.10±0.21	0.06±0.19
Articulate Anything	0.08±0.21	0.42±0.64	0.04±0.15
Ours w/o Refine	0.01±0.03	0.32±0.42	0.27±0.42
Ours w/o Coarse	0.01±0.03	0.15±0.25	0.05±0.14
Ours w/o Segment	0.01±0.03	0.35±0.43	0.30±0.44

Table C.4: Evaluating the variance of our method and ablated version. We run 10 experiments with different seeds on S-Dataset. We compute the mean value of each run and report the mean and standard deviation of these mean values. Our method consistently outperforms the ablated version of our method.

Joint	Methods	Axis↓	Position↓	Type↓	State↓	Failure↓
Revolute	Ours	0.30±0.03	0.13±0.01	0.15±0.02	0.25±0.00	0.06±0.00
	Ours w/o Refine	1.15±0.08	0.26±0.01	0.75±0.06	0.63±0.04	0.04±0.00
	Ours w/o Coarse	1.26±0.01	0.34±0.00	0.79±0.03	0.69±0.01	0.02±0.00
	Ours w/o Segment	0.38±0.03	0.17±0.00	0.19±0.02	0.33±0.00	0.06±0.00
Prismatic	Ours	0.23±0.03	N.A.	0.06±0.03	0.05±0.01	0.00±0.00
	Ours w/o Refine	0.29±0.06	N.A.	0.11±0.04	0.10±0.02	0.00±0.00
	Ours w/o Coarse	0.48±0.01	N.A.	0.08±0.02	0.12±0.00	0.03±0.00
	Ours w/o Segment	0.25±0.03	N.A.	0.08±0.02	0.07±0.00	0.00±0.00

C.4 Camera pose estimation results

There are already several dynamic scene reconstruction works that can infer camera poses of a video sequence. This comparison aims to demonstrate why not directly use camera estimation results from existing dynamic scene reconstruction models. We compare our camera pose prediction results with two very recent works MonST3R [55] and CUT3R [42]. We transform all the camera poses to the camera coordinate of the first video frame. We compare both camera rotation error and translation error on our dataset. From the results shown in Tab. C.6, we can find that our method can produce more accurate camera pose estimation, particularly camera orientation, against recent works on dynamic scene understanding.

Table C.6: Evaluation on camera pose estimation.

Methods	Rotation Error↓	Position Error↓
Ours	0.08066	0.09246
MonST3R	0.14482	0.08729
CUT3R	0.14663	0.10536
Ours w/o Refine	0.11814	0.10918
Ours w/o Coarse	0.11073	0.16544
Ours w/o Segment	0.11547	0.16892

C.5 Moving Map Prediction Choice

In our coarse prediction module, we use MonST3R [55] to generate a moving map of the input video. A natural question is, why not use an off-the-shelf motion segmentation method? Here we compare the output of moving map prediction results from MonST3R and Segment Any Motion [7], which is a very recent work on motion segmentation on videos. In Fig. C.2 we can find that Segment Any Motion produces much larger error prediction results compared to MonST3R. Segment Any Motion first computes point tracks of the video, and selects moving points based on point tracks. Then, it

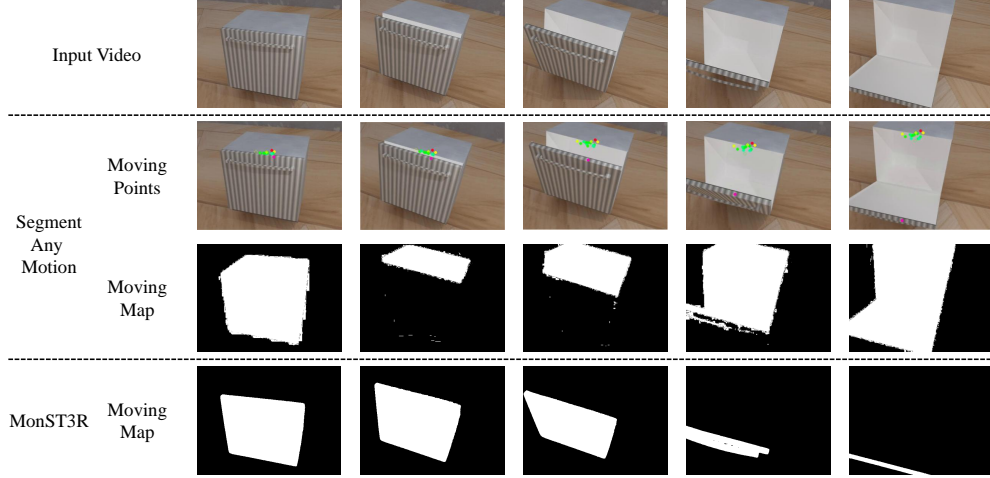


Figure C.2: Comparison of moving map prediction between MonST3R and Segment Any Motion. From this example, we can find that the point-track-based method is very sensitive to the location of the points. Moving points predicted by Segment Any Motion lie on the edge of the door of this dishwasher. Using these points as a prompt for Segment Anything 2 [30] will likely segment the wrong parts and objects.

uses these points to prompt Segment Anything 2 to generate the moving map of the video. But in this case, the moving points lie around the edge of the door of this dishwasher. Thus, some points prompt selecting the dishwasher’s top as the moving part instead of the front door. This small position error will lead to a large error in moving map segmentation. In contrast, MonST3R computes the moving map based on optical flow and estimated camera poses. It subtracts the optical flow caused by camera motion from the final optical flow to determine which part of the video is moving. This strategy turns out to be much robust to point-track based motion segmentation methods.

C.6 Video segmentation strategy

Automatic part segmentation on the video plays a vital role in our pipeline. We find that automatic part segmentation performs better on the reversed video. Most videos in our dataset are about opening the door or drawer. In this case, some movable parts are difficult to segment out due to the very high similarity of texture of every part of the object. But when the door is opened or the drawer is pulled out, it’s much easier to distinguish the movable part from the other parts of the object. Fig. C.3 illustrates the segmentation results on a normal and reversed video.

C.7 Additional real-world results

In the main paper we demonstrate the results of reconstructing real articulated objects with revolute joint. Here we demonstration reconstructing articulated objects with prismatic joints in Fig. C.4. Articulate Anything still struggles with retrieving correct object in Storage case, while RSRD fails to crop and cluster the object from the environment for the Drawer case.

C.8 Failure case analysis

Our pipeline relies heavily on automatic part segmentation, since the refinement module is built upon part representation. We summarize three common failure modes in Fig. C.5. The first row shows that auto part segmentation may not identify newly observed parts in the video. It combines the interior of the drawer with its front face. When computing the Chamfer distance for optimization, the interior parts cannot find correspondence on the object surface points cloud, resulting in inaccurate estimation. The second row shows that auto part segmentation may not segment parts accurately. In this case, a part of the storage body is segmented out with the door. This part is static while the door is moving. This will also introduce noise to the Chamfer distance. The third row illustrates that auto part segmentation may not segment parts consistently. The lid of the trash can belongs to different

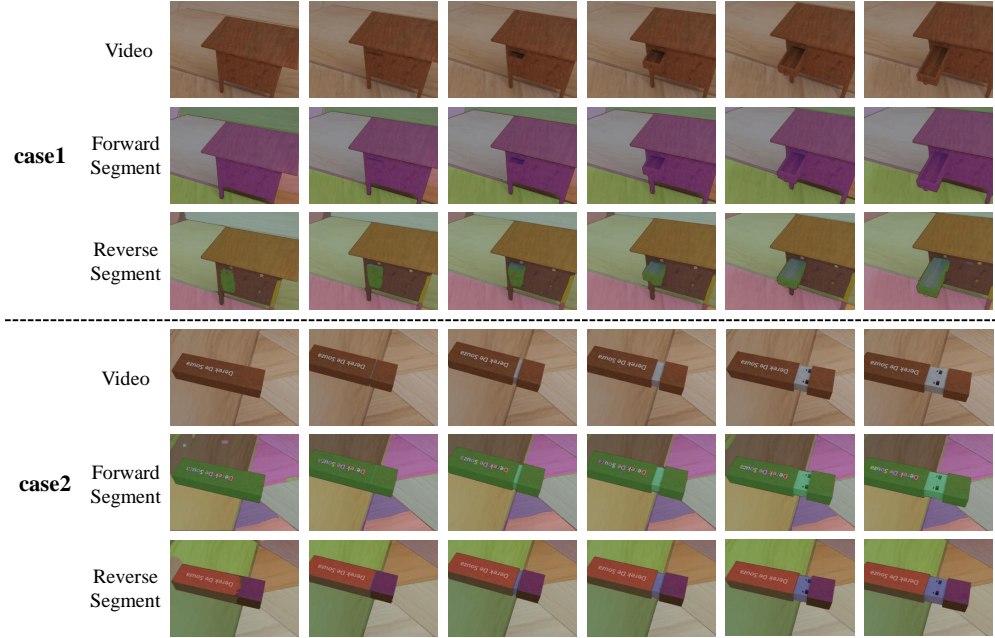


Figure C.3: Examples of automatic part segmentation on videos with different orders. From these two cases, we can find that segmenting the video in a reversed order is easier to identify different parts of the object, particularly when the texture of different parts is very similar.

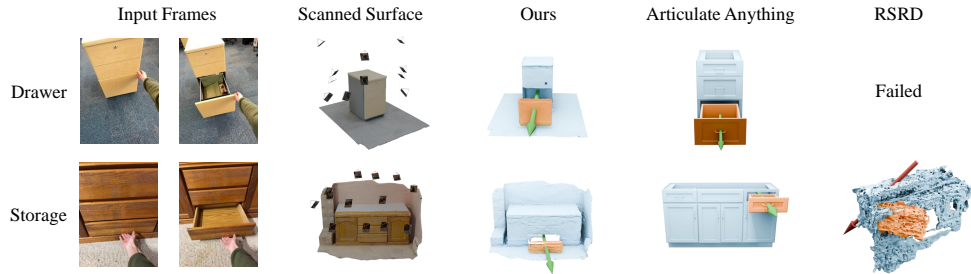


Figure C.4: Qualitative results on real data.

segments in the video, shown in two different colors. Our pipeline will filter out parts that do not appear in the first frame. In this case, the trash can lid will be considered as newly observed parts and filtered out in the later sequence of the video, losing important information for joint estimation.

From the failure modes discussed above, we believe geometry information will help to refine the segmentation results. For example, we can use recent 3D segmentation approaches like SAMPart3D [53] and PartField [18] to produce segmentation on the observed point cloud. This geometry-based segmentation can correct the errors in automatic part segmentation, which is purely image-based.

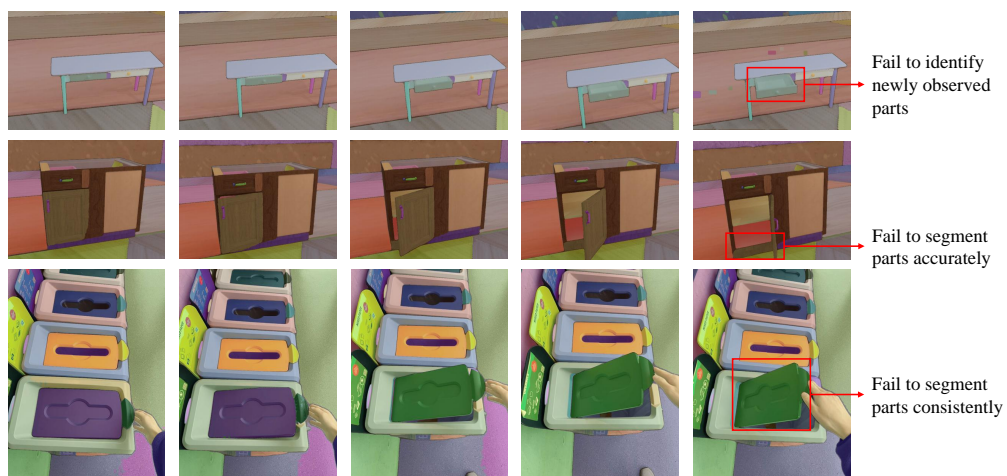


Figure C.5: Three common failure modes on automatic part segmentation.