MotionSplicer: Part-Based Motion Editing for 4D Volumetric Videos

Chaerin Min, Praccho Muna-McQuay, Tao Lu,
James Tompkin, and Srinath Sridhar
ivl.cs.brown.edu/research/motionsplicer

Brown University, Providence, RI, USA

Abstract. 4D volumetric video via Gaussian Splatting (GS) enables free viewpoint control of real-world scenes but does not enable motion editing. One challenge is achieving *part-based* control in multi-object scenes; prior works focus on single objects, assume rigid parts, or require manual annotations. MotionSplicer is a template-free and annotation-free method that automatically discovers 3D parts in GS. To bridge the semantic-to-motion domain gap, it extracts initial parts from temporal difference images and resolves multi-view inconsistencies via an adaptive 3D unify-and-split process. For robust tracking, it optimizes a time-invariant feature grid with temporal refinement to learn spatially and temporally coherent skinning weights. Experiments show that MotionSplicer can be applied more generally to objects and scenes than state-of-the-art methods, and enables creative motion edits like isolation, ghosting, and echo.

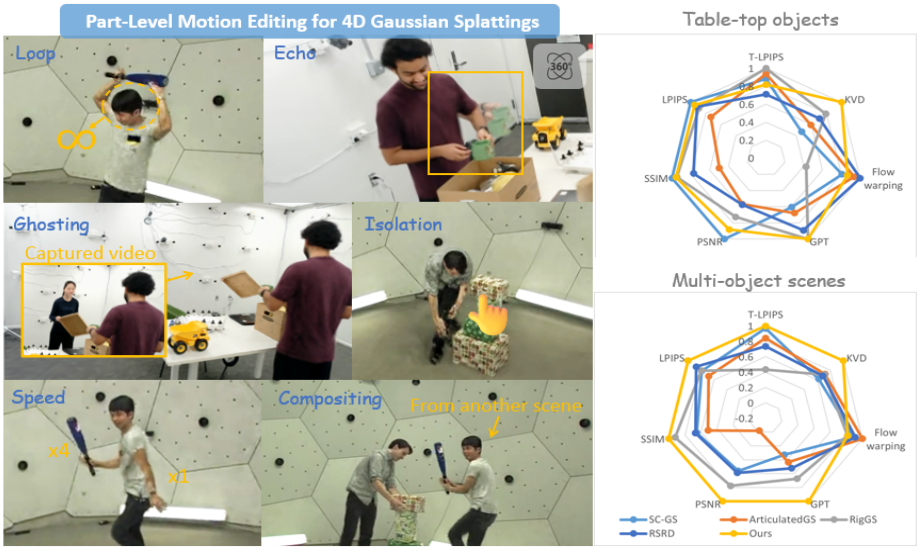


Fig. 1: (Left) **MotionSplicer** enables annotation-free editing of complex part-based motions in real 360° scenes. It uses sparse handle transformations, time-invariant Gaussian skinning weights, and 2D segmentation priors with per-scene optimization, to provide motion editing capabilities without manual labels. (Right) Comparing existing systems, complex multi-object scene edits benefit more from MotionSplicer.

1 Introduction

Creating 3D media that is interactive and supports free-viewpoint navigation from multiple cameras is a long-standing problem in computer vision and graphics [10, 51, 62, 64, 73, 99]. Since the real world is spatial and dynamic, 4D volumetric videos [31, 52, 83] have gained popularity [11, 74, 79, 87]. While 4D volumetric videos are emerging as essential components for applications like world models, XR, robot simulation, and game engines [2, 3, 15, 26, 68, 81, 96], their utility is limited because we cannot yet interactively edit and manipulate captured motion of objects and parts in 4D.

Previous work on editing 3D or 4D content focuses on entire objects [55, 94, 97, 98], or articulated objects [5, 14, 20, 37, 44, 57, 70, 77, 82, 85]. However, these methods are limited in their motion representation, largely focusing on simple articulation and struggling with non-rigid objects [32,80]. Other works [29,93] rely on hard-coding the movable parts, which requires tedious manual annotation and lacks generalization. Similarly, relying on known 3D meshes or URDFs [25, 37, 82] prevents scalability and generalization. While some recent works [43, 48, 71, 72, 80, 88, 91] have pursued *template-free* and annotation-free part-level 4D control of non-rigid objects, they are often restricted to a single object and do not scale to complex, multi-object real-world scenes.

To overcome this, we introduce **MotionSplicer**, a method to reconstruct multiple moving objects within a scene at the part level, such that we can enable controllable motion editing. The scene is modeled using 3D Gaussians [31] with part weights, where scene parts move relative to each other following linear blend skinning. We *infer* these part and motion relationships over time to enable previously impractical edits to be performed on a 4D volumetric video simply by selecting a part handle and adjusting its timing or motion. This enables object and part edits like *motion speed control*, *motion looping*, *motion isolation*, *compositing*, *ghosting*, and *echo*. Optionally, this also allows spatial editing, including novel 6-DoF transformation of parts and part scaling in the scene. We include a supplementary video that shows interactive editing and free-viewpoint video of edited motion in MotionSplicer.

For part-level 4D decomposition, naive application of 2D foundational segmentation models [13, 60] is insufficient due to a domain gap: they target *semantics* in 2D images rather than *motion boundaries* in multi-view video, causing severe under/over-segmentation and multi-view inconsistencies. To overcome this, we propose a novel 4D part discovery framework. This initializes motion handles by applying segmentation priors exclusively to temporal difference images, then lifts these to 3D, then applies a 3D unification and splitting algorithm that dynamically corrects motion segmentation.

For motion tracking, independently optimizing Gaussians over time can be unstable. Instead, we introduce a continuous spatio-temporal optimization strategy. To encourage spatial smoothness across non-rigid parts, we map skinning weights to motion handles using a time-invariant coordinate-based feature grid. Then, to ensure long-term temporal coherence, we couple this with a streaming

Methods	(a)	(b)	(c)	(d)	(e)
SC-GS [18]	Mono video	✓	△	No	MLP
RigGS [91]	Mono video	✓	✗	Heuristics	MLP
ArtiGS [14]	Mv start/end	✗	✗	Movable prediction on static	Optimize
RSRD, POD [32, 80]	Static mv +Mono video	✗	△	Semantic features on static	Optimize and distill
Ours	Mv video	✓	✓	Semantic features, motion cues (I_d+U&S)	Optimize w/ TC refine and learnable grid

Table 1: Differences in methodology and capabilities. (a) Input requirements (b) Capability to address soft boundaries (c) Capability to address complex multi-object scenes (d) Part discovery prior (e) Temporal tracking. △ means theoretically feasible, but demonstrated only to a limited extent. Mv indicates multi-view. By leveraging both semantic and motion cues and using more complex techniques, MotionSplicer provides the most comprehensive solution for continuous 4D part editing in complex, multi-object scenes.

temporal consistency refinement loss that re-evaluates previously optimized frames, encouraging coherent part decomposition throughout the 4D sequence.

Our experiments on public datasets [28, 40, 87] and real-world captures show that our method represents the 4D editable parts more flexibly in complex multi-object scenes than other approaches that rely on dense control point seeding and pruning [18], skeletal structures [91], axis-finding [14], or segmentation of table-top objects [32, 80]. We summarize methodological differences in Table 1. In summary, we present **MotionSplicer**, a method for template-free, part-level motion editing in 4D volumetric videos of scenes. Our contributions are:

- **A motion-aware 4D part discovery framework** that extracts motion handles from raw video without annotations. By lifting 2D segmentations of temporal difference images and applying a novel 3D unification and splitting process, we automatically resolve segmentation inconsistencies to build reliable part priors.
- **A continuous spatio-temporal optimization strategy.** By learning skinning weights via a time-invariant feature grid coupled with a temporal consistency refinement, we effectively enforce spatio-temporally coherent motion tracking for complex, non-rigid parts.

2 Related Works

Editable 4D Volumetric Videos 3D reconstruction with NeRF [52] and Gaussian Splatting [31] has been an active research area due to its many applications [51, 62, 73, 99], including AR/VR [6, 7, 26, 61, 96], interactive scene editing [9, 33, 34, 63], and entertainment [41]. If editable 3D can be extended to editable 4D for videos with motion controllability, then new scenes including articulation and deformation are possible [27, 37, 45, 50, 85], with applications in interactive robotics [1, 12, 22, 24, 69, 77, 82] and autonomous driving [4, 55, 89, 98].

Early controllable 4D works [94] were built upon layers and limited to bounding boxes only. Afterwards, more approaches attempted to provide scenes with small novel motions [26, 58, 76, 78, 96], but these did not consider part-based motion editing. Meanwhile, in self-driving, dynamic 3D motion graphs have been studied [55, 89] to edit cars on the road. Our goal is to address a more challenging problem of rigid and deformable part-level motion editing of 4D scenes with multiple objects.

Part-Level Editable 4D Volumetric Videos Earlier works in part-level motion editable 3D representations were centered on template-based approaches, including for humans [17, 38, 56, 59] and animals [47, 90], and approaches using manual annotations [29,57,93]. In robotics, work has proposed to predict joint axes and rotations for articulatable meshes at a category level [23,50,70]. Category-free articulation methods have tried to find part-level motion in meshes [20, 27, 44, 50], radiance fields [45,57,66,77], or Gaussians [5,14,82], but they focused only on rigid parts. Some works have assumed known meshes or URDFs [25, 70, 85], some mesh-based works were template free [37,43], and some relied on mesh retrieval [37,82] or were trained for use on specific domains [43]. Additional approaches have suggested to jointly learn part-level control points and/or skinning weights [54, 67, 88], but these lack real-time rendering of edited images as they were based on NeRFs. Later, 3DGS [31]-based methods [36, 48, 71, 72, 91] tried to optimize skeletons, joints, rigs, or low-dimensional abstractions of a single object. Specifically, methods like RSRD [32] and POD [80] are restricted to part-level rigid motions on single tabletop objects; furthermore, their lack of multi-view video input makes them very sensitive to clustering, segmentation, and view optimization, as shown in our experiments. Our work explores the challenging problem of editing motion for 4D volumetric videos in complex scenes with multiple objects and non-rigid object parts, and without any provided annotations.

3 Method

First, we provide an overview of MotionSplicer, as shown in Figure 2, then we describe how we identify parts in the scene. Then, our spatio-temporal optimization operates in a continuous frame-to-frame manner, optimizing 6-DoF part transformations at each timestep while jointly refining a temporally shared skinning weight grid. Finally, we demonstrate performing the motion edits.

3.1 Formulation

To represent the part-level controllable motion in 4D, we adapt the widely-used Linear Blend Skinning (LBS) model [49] to the 3D Gaussian Splatting framework. This approach is highly effective as it decomposes part-level dynamic motion, which is key for intuitive editing. Our representation thus consists of three components: (1) a single set of Gaussian primitives $g_i : \{\mu_i, q_i, \sigma_i, s_i, c_i\}, i \in \{1, ..., N\}$ at the canonical frame $t = 0, t \in \{0, .., T\}$, (2) sparse 6-DoF handles $p \in \{1, .., P\}$, and (3) time-invariant skinning weights $w_i \in \mathbb{R}^P$, such that $\sum_{p=1}^{P} w_{ip} = 1$, that are associated with each primitive g_i. N is the number of Gaussian primitives and P is the number of parts identified in the scene (as

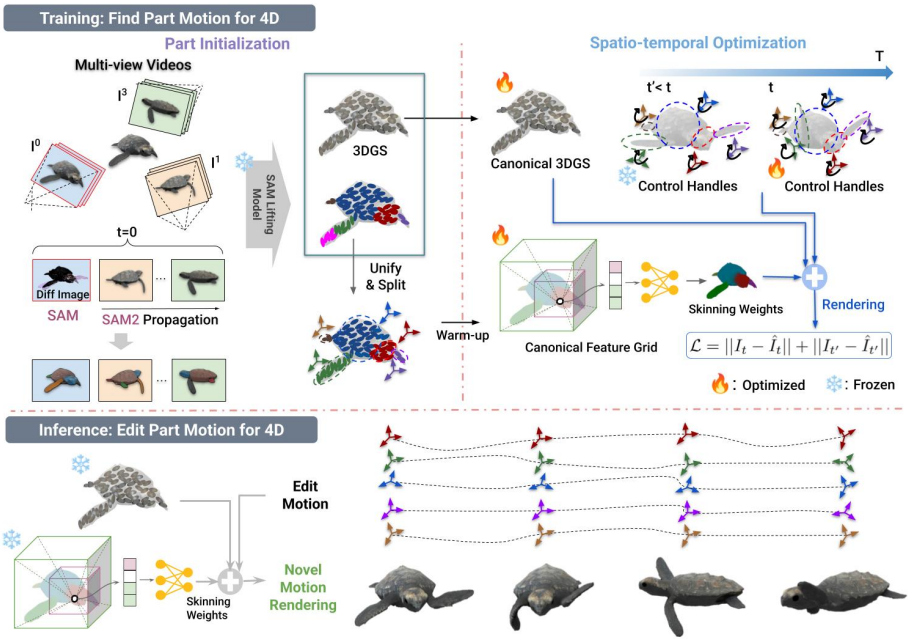


Fig. 2: (Top) Our pipeline initializes sparse 6-DoF handles and a time-invariant skinning weight grid by distilling 2D foundational model segmentations to our additional logit vector m_i in 3DGS primitive g_i. We then perform our spatial-temporal optimization process of per-timestep handle transformations and the shared skinning weight grid using render losses. Temporal consistency is further enforced by also applying losses to randomly sampled past frames using the current skinning weights. (Bottom) At inference, users can perform intuitive **part-level 4D motion editing** by adjusting the handles within their learned trajectory limits.

described in the following section) and T is the number of video frames. $\mu_i \in \mathbb{R}^3$, $q_i \in \mathrm{SO}(3)$, $\sigma_i \in \mathbb{R}$, $s_i \in \mathbb{R}^3$, c_i denote the 3D mean of the primitive, local orientation of the primitive, opacity, scales, and Spherical Harmonics coefficients, respectively. In the canonical frame, the handles are located at 3D position $c_p \in \mathbb{R}^3$. At $t > 0$, to edit the part-level motion, we transform the handles at each timestep t, both in rotations $R_p^t \in \mathrm{SO}(3)$ and translations $T_p^t \in \mathbb{R}^3$. The handle transformation at time t relative to the canonical frame is denoted as $\mathcal{H}^t = \{(h_p^t)\}, h_p^t \in \mathrm{SE}(3)$.

3.2 Identifying Handles and Skinning Weights

To achieve our goal of part-level motion editing in 4D volumetric video, we first need to extract the scene's constituent parts from the captured sequence. However, finding motion parts only from videos is a challenging problem. Despite the remarkable progress in dynamic GS-based models [42,46,65,86], building a motion-editable representation is still limited to either object-level [72, 80], synthetic [71,

91], mainly rigid parts [5, 14, 82], or label-based [57, 93]. We hypothesize that reliable part segmentation is the most challenging task for 4D scenes.

To overcome this challenge, we apply 2D segmentation foundation models [13, 60] to the canonical frame's multi-view images and use the segmentation as supervision to optimize each 3D primitive's logit parameter $m_i \in \mathbb{R}^{P'}$, defined in addition to other parameters in g_i, similar to a SAM Lifting model [92]. Here, P' denotes the maximum number of handles assumed to exist in a typical scene. We use the logit to initialize the skinning weights w_i. To be more specific, because real scenes often contain many static object parts, we provide a difference image $I_d = \|I_{t=0} - I_{t=\bar{t}}\|$ at the reference view, rather than the raw frame $I_{t=0}$, as input to SAM [35]. This encourages SAM to generate masks only for regions that are in motion. We then apply SAM2 to the spatially sorted multi-view images (treated as a sequence), using the masks from I_d as the initial prompt for propagation. Pixels outside these moving-part masks are assigned to a background handle. By default, we run SAM2 [60] in its *segment everything* mode, but optionally allow user click prompts when desired.

However, relying solely on the 2D-lifted initialization from SAM can be problematic, due to occlusion and over-/under-segmentation. To refine the initial P' groups, we perform a group unification and splitting (U & S). This is performed during the optimization of $t = 1$ (details in the following section) and results in the final set of P handles. Optionally, this U & S can be performed periodically to accommodate evolving structures if applicable. We unify groups with small relative translation and rotation, correcting over-segmentation. On the other hand, we split under-segmented groups by detecting regions with high accumulated gradients in the skinning weight feature grid and dividing them along their principal geometric axis, estimated via PCA. A more detailed description of U & S is provided in the supplementary PDF. The initialized skinning weights $m_i \in \mathbb{R}^{P'}$ are then merged or split along the P' dimension according to these unification or splitting decisions, yielding final weights $w_i \in \mathbb{R}^{P}$. Finally, we initialize the canonical position c_p for each part handle p as the geometric centroid of the 3D mean positions μ_i of all Gaussian primitives assigned to that part (i.e., treating the highest skinning weights w_i as the assignment probability for part membership).

3.3 Optimizing Handles and Skinning Weights

Despite providing a useful initial segmentation, 2D foundation models are insufficient on their own. They are not trained to identify articulated motion parts (e.g., a fridge door versus its body), and sparse multi-view inputs often produce inconsistent segmentations across views. To correct these issues, we introduce a scene-specific optimization of the skinning weights and handle transformations.

First, at time t, the optimization process proposes a handle transformation with respect to the canonical frame. With this handle transformation and skinning weights, the Gaussian is transformed as follows:

$$\mu_i = \sum_{p \in \{1,..,P\}} w_{ip}(R_p^t(\mu_i - c_p) + c_p + T_p^t), \quad \text{where} \quad q_i = \text{quat}(\sum_{p \in \{1,..,P\}} w_{ip} {R_p^t}^T \text{R}(q_i)),$$

where R(·) denotes the transformation from quaternion to rotation matrix. Optimizing the skinning weights w_i independently for each Gaussian primitive can lead to noisy or spatially inconsistent results. To prevent this and enforce spatial smoothness, we instead predict w_i using an NGP [53]-style learnable feature grid. This feature grid takes the canonical position μ_i of a Gaussian as input and outputs its corresponding skinning weights w_i. Before starting optimization, the feature grid is warmed up by learning $w_i(\mu_i)$, supervised by the skinning weights obtained from the previous section. We then optimize the parameters of this feature grid instead of weights w_i directly. The transformed Gaussians at t are then rasterized as,

$$C = \sum_{j \in J} c_j \alpha_j \prod_{k}^{j-1} (1 - \alpha_k)$$

where $\alpha_j = (1 - \exp(-\sigma_j \delta_j))$. δ_j is the ray interval, and J is the number of Gaussians intersecting with the ray [31]. These rasterized images I are compared with multi-view input images $\hat{I}_t^v, v \in \{1, .., V\}$ by,

$$\mathcal{L} = \lambda ||I - \hat{I}||_1 + (1 - \lambda)(1 - SSIM(I, \hat{I}))$$

to back-propagate the loss into $\mathcal{H}^t$ and the temporally shared feature grid of w_i. We repeat this process from $t = 1$ through $t = T$. This self-supervised refinement of w_i using the multi-view video allows understanding of inherent scene-specific part motion.

3.4 Temporal Consistency Refinement

As the streaming optimization proceeds from $t = 1$ to T, the handle transformation $\mathcal{H}^t$ for each frame is sequentially optimized and then frozen. When optimizing the current frame t, we also randomly sample a previously optimized frame $t' < t$ to enforce that the temporally-shared w_i remain consistent across all motions. We then compute the reconstruction loss for the sampled frame t' using its already frozen $\mathcal{H}^{t'}$, and back-propagate this loss only to the time-invariant feature grid of w_i. This temporal consistency refinement helps w_i learn by incorporating information from past motions.

4 Experiments

4.1 Implementation Details

We used both synthetic data (Sketchfab and OmniGibson [39]) and real 360° datasets (Panoptic [28], as well as our own captures) of RGB videos. We tested our method on 3 sequences from Panoptic, 1 from our own captures, 2 from OmniGibson, and 3 from Sketchfab. Our skinning weight network uses a multi-resolution hash grid [53] and a 2-layer MLP. During inference, our method renders in real time on an RTX 3090 GPU; per-sequence FPS and resolutions are reported

	Reconstruction			Motion Editing			
	PSNR ↑	SSIM ↑	LPIPS ↓	T-LPIPS ↓	KVD ↓	Flow-Warp ↓	GPT-EE ↑
Synthetic Objects							
SC-GS [18]	**40.04**	**0.993**	**0.0166**	0.0079	0.9493	0.9611	3.7
ArtiGS [14]	27.07	0.896	0.0921	0.0052	0.8716	0.3217	3.9
RigGS [91]	31.83	0.983	0.0473	**0.0023**	0.7444	2.8741	**4.8**
RSRD [32]	27.26	0.949	0.0398	0.0161	0.7973	**0.0000**	4.5
MotionSplicer	36.52	0.985	0.0311	0.0109	**0.6111**	0.6723	**4.8**
Synthetic Scenes							
SC-GS [18]	32.30	0.959	0.1386	0.0033	1.0945	0.7440	1.0
ArtiGS [14]	26.72	0.930	0.1646	0.0052	0.7836	**0.2647**	2.8
RigGS [91]	29.90	**0.971**	0.2629	0.0142	0.9728	2.2541	4.5
RSRD [32]	32.25	0.948	0.0891	0.0167	0.9274	0.4389	3.0
MotionSplicer	**34.49**	0.967	**0.0428**	**0.0022**	**0.7007**	0.4407	**4.7**
Real Scenes							
SC-GS [18]	22.05	0.730	0.3990	0.0125	1.0039	29.1976	2.0
ArtiGS [14]	16.36	0.679	0.5268	0.0494	1.0734	**17.4413**	1.7
RigGS [91]	26.40	0.853	0.3299	0.0810	0.9895	31.1659	1.7
RSRD [32]	19.21	0.677	0.3856	0.0436	1.0770	26.5958	1.8
MotionSplicer	**27.36**	**0.892**	**0.1683**	**0.0064**	**0.8086**	31.5803	**4.7**

Table 2: Quantitative evaluation compared to relevant state-of-the-art works [14, 18, 32, 91] that use motion-editable representations. From synthetic objects, through synthetic scenes, to real scenes, we see a trend that MotionSplicer performs better at both reconstruction and edited motion except for flow-warping error, presumably due to the ambiguity of occlusion masking for flow estimation. Note that flow-warp metrics can artificially reward zero-motion and are affected by real-scene occlusions.

in the supplemental material. *Further details, including all hyper-parameters, training times, and memory usage, are provided in supplemental material.*

Evaluation metrics For Table 2, we evaluate the reconstructed 4D rendering using PSNR, SSIM [75], and LPIPS [95]. For the edited motion 4D video, we used GPT-EE score, which employs GPT-4o as a judge to rate the alignment of edited results and the editing intention described in straightforward language on a [1, 5] scale. KVD measures perceptual fidelity [16] of edited video against the captured video via maximum mean discrepancy of DINOv2 feature distributions, to avoid using ImageNet as the reference [19]. T-LPIPS assesses temporal smoothness through average LPIPS distances between neighboring frames. Flow-Warp computes the mean L1 error of RAFT-estimated optical flow warping consistency, excluding occlusions via its forward-backward check.

The Radar chart shown in Figure 1 is the average of table-top object results and multi-object scene results, respectively. The numbers are normalized to 1, and are inverted if lower is better, for visualization. Thus, the larger the area under the curve, the better the method is across all metrics.

Method	PSNR ↑	SSIM ↑	LPIPS ↓	T-LPIPS ↓	KVD ↓	Flow-Warp ↓	GPT-EE ↑
SAM2 only	30.40	0.9265	0.1530	0.0088	0.6658	0.329	3.5
w/o I_d	32.81	0.9588	0.0681	0.0094	0.6726	0.329	**4.8**
w/o feature grid	34.00	0.9583	0.0681	0.0088	0.6462	0.325	4.5
w/o TC refine	35.55	0.9681	0.0479	0.0092	**0.6425**	0.313	**4.8**
Full	**36.54**	**0.9729**	**0.0345**	**0.0014**	0.6693	**0.303**	**4.8**

Table 3: Proposed components ablation. We validate the use of difference image I_d, feature grid, and TC refinement, as well as the necessity of our method over relying solely on a semantic segmentation model. The jump in T-LPIPS reflects both temporal flickering and overall artifacts (e.g., entangled parts, trailing floaters), and the latter dominated the metric in this case.

The user study shown in Figure 4 was conducted with 20 participants of diverse expertise (M=3.1/5, SD=1.33). In the relative evaluation, users select the better one from 6 randomized pairs of randomized left/right. All p-values were $p < 0.001$. Absolute evaluation assesses alignment with user intention on a 5-point scale (error bar indicates SD). Questionnaires and detailed scores are in the supplementary material.

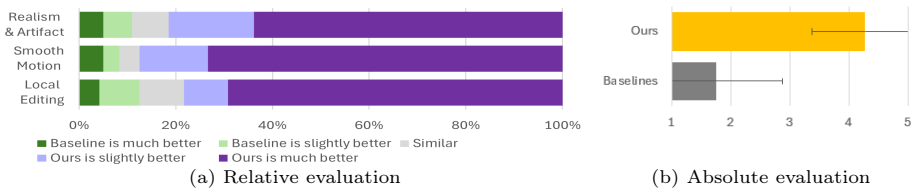
4.2 Experimental Results

Part-Level Motion Editing We demonstrate versatile part-level motion and spatial editing capabilities (Fig. 3). By manipulating the learned 6-DoF handles and their temporal trajectories, users can intuitively achieve part-based 4D motion edits such as Loop, Ghosting, Echo, Speed adjustment, and Freeze. Furthermore, we can Isolate simultaneous motions by controlling handles individually, or Composite parts across different scenes by transferring their associated Gaussian primitives via skinning weights. Beyond 4D motion editing, our method naturally extends to spatial editing (e.g., Scaling and Rotate), enabling novel part-level geometric transformations distinct from the captured transformation, while maintaining the motion.

Quantitative Comparisons Table 2 measures reconstruction and motion editing, and the Radar chart of Figure 1 effectively visualizes the trend. 'Synthetic Objects' consists of objects from Sketchfab, and 'Synthetic Scenes' comprises scenes from OmniGibson, and 'Real Scenes' includes scenes from Panoptic Studio and our own captured scenes. Temporal smoothness metrics including T-LPIPS and Flow-Warp are not perfect measures, because some baselines simply fail to move anything and thus have 'better' temporal smoothness. Additionally, the forward-backward check sometimes eliminates all unnatural motions and results in falsely good numbers, even 0.0000. However, MotionSplicer's consistent outperformance on perceptual scores such as GPT, KVD, and LPIPS, with significantly preferred results in the user study (this includes a motion smoothness question), demonstrates MotionSplicer's plausible motion editing compared to baselines. Additionally, although SC-GS has better reconstruction numbers on isolated synthetic objects, MotionSplicer outperforms it in overall motion editing metrics and user preference.

Fig. 3: Part-level 4D motion editing. (Loop) amplifying subtle motion, (Ghosting) leaving object motion only, (Compositing) removing and copying object from another scene, (Echo) overlaying part-motions, (Speed) part-level speed control, and (Freeze) freezing motion components. Additionally, our method naturally supports **spatial editing** on top of 4D motion. (Scaling) making a few objects smaller and putting them on a table. (Rotate) Rotating a part in a different direction from the captured rotation. *We show our editor and its free-viewpoint video in the supplementary video.*

To directly address potential fairness concern, we tried adapting SC-GS, RigGS, and RSRD on Car, Home, and Boxes to use multiple views. ArticulatedGS is already multi-view. While average PSNR changes only modestly (28.79 → 29.10, 27.65 → 26.74, 15.39 → 16.97), all remain below ours (32.40). The gap is from core representation issues, not view counts: RigGS assumes a single hierarchical



(a) Relative evaluation (b) Absolute evaluation

Fig. 4: User study. Because of the limited scope of motion editing metrics, we performed a user study to evaluate human preference for MotionSplicer over baselines [14, 18, 32, 91] on average, regarding the edited motion in 4D. A higher absolute evaluation score is better.

tree; SC-GS relies on dense joint optimization (prone to local minima); and RSRD depends on a semantic segmentation that leads to severe under/over-segmentation. Instead, our I_d and U&S adapts to motion boundaries, ensuring robust part discovery via our feature grid.

Qualitative Comparisons Figure 5 presents the necessity of using our method for 4D motion editing. SC-GS [18] can have smooth skinning weight distribution, but is not suitable for rigid parts like a car door, a scene with multiple doors, or complex real scenes, as shown with their skinning weight visualization. ArticulatedGS [14] tries to predict the axis to articulate objects and predict the moving mask (green color). When it comes to scenes, it struggles to figure out the moving part and the correct axis. The two doors of the car are also modeled with one axis, making it difficult to reanimate only one door. RigGS [91] estimates a hierarchical skeleton for an object; we visualized their skeleton with nodes and edges and their skinning weights with color. The rig on the car seems reasonable (though still not separating two doors), but the scene rigging struggles to place the rig on the moving parts, and fails to move the oven door as a part of a full scene. Robot-see-Robot-do (RSRD) [32] shows nice separation with DINO features, but the monocular part-level rigid tracking with heavy reliance on joint optimization makes the segmentation, clustering, or tracking limited. Note that we tried our best to fine-tune RSRD. In contrast, with adaptive U&S, TC refinement, and the most flexible representation with non-rigid skinning weight grid, MotionSplicer successfully handles objects and scenes, rigid and non-rigid parts, with a single unified method.

4.3 Ablation Study

In Figure 6, an appropriate number of parts leads to better motion optimization than too many or too few parts. Our model adaptively approximates the proper number of parts by U&S. Nevertheless, without I_d, initializing with all semantic parts creates a complex, inefficient start (due to $O(N \times P)$) that converges to 10 messy parts even with U&S. Our difference image (I_d) initialization empirically provides cleaner part finding. As shown in the turtle's shoulder, unlike hard 3D segmentations, MotionSplicer captures smooth deformation during motion by allowing each Gaussian primitive to have weighted assignments to multiple parts using skinning weights. Finally, our temporal consistency (TC) refinement

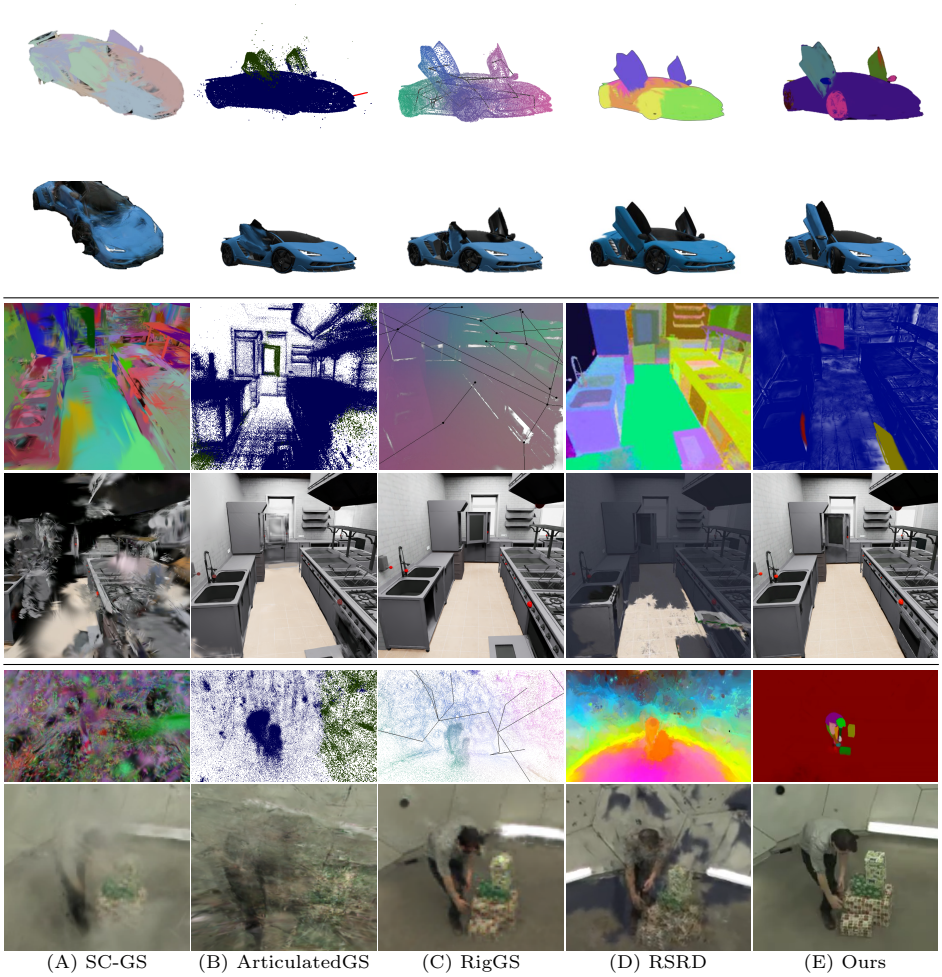


Fig. 5: Visualization of representations (every odd row) Predicted skinning weights (A, C, E), DINO feature (D), motion mask (B), axis (B), and skeleton (C). Our approach ensures reasonable part understanding in both objects and scenes. **4D motion editing results (every even row)** (D)'s segmented foreground is overlaid with background, and this was the best segmentation and clustering it could do. MotionSplicer, as a single method, better generalizes to rigid and non-rigid parts, and objects and scenes. *Best viewed in the supplementary video.*

ensures consistency during the streaming optimization. All ablation results are also quantitatively validated in Table 3.

As one of our key motivations, we additionally show Figure 7. Because of the varied viewing angles and lack of understanding of motion segmentation, SAM [35,60] can often under-segment an important moving part (e.g., fridge door and fridge body) or have multi-view inconsistency. Human part segmentation

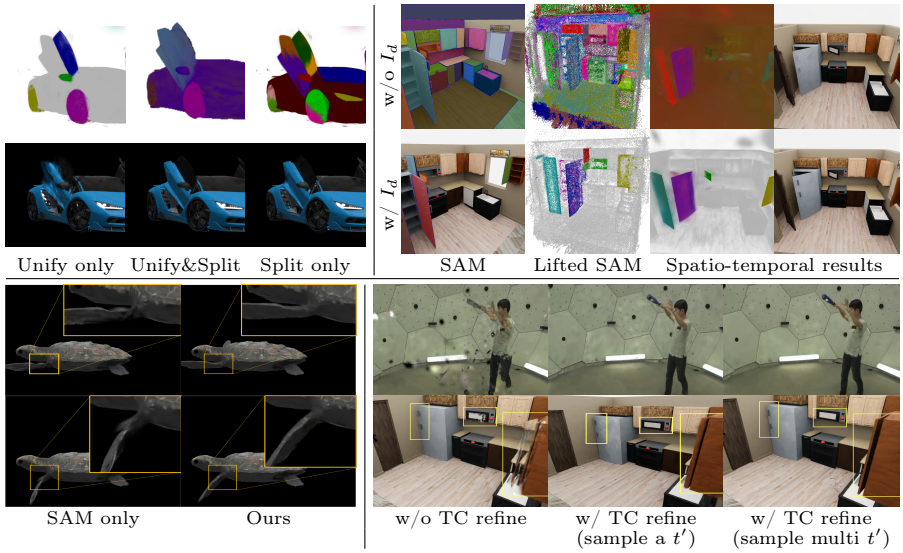


Fig. 6: Ablation study. (Top Left) Effect of Unify and Split on the part initialization and motion reconstruction quality. (Top Right) w/o I_d initializes the weights with all semantic parts, making undesirable complexity in practice. (Bottom Left) MotionSplicer captures smooth deformation at the turtle's shoulder. (Bottom Right) "sample multiple t'" means sampling the maximum number of previous frames, within the memory limit, which was either 4 or 3 depending on the data complexity.

models [13] might miss loose clothes or hair bangs. However, with MotionSplicer's ability to ensure consistency spatio-temporally, the initial 3D segmentation from under-segmented or inconsistent parts is fixed into a clean 4D segmentation that is aligned with the motion. It also shows (bottom) our method's robustness to potential dis-occlusion, e.g., the surface between the box and the floor.

4.4 Further Analysis

Alternative Approaches Since we use 2D foundational segmentation models only for the first frames, a question might arise: why not incorporate 2D foundational models for temporal understanding too? We were motivated empirically by the results shown in Figure 8. Although we provided the same first-frame 2D mask for initializing CoTracker3 [30] and GMFlow [84], those, if used as off-the-shelf, are fragile in tracking object parts in a spatio-temporally consistent way, resulting in quality degradation. On the contrary, MotionSplicer is able to maintain the parts throughout the video, while the person and the bat are moving fast.

Sensitivity To evaluate sensitivity of our model to experimental settings, we performed two analyses: *Reduced Number of Cameras:* We tested robustness by reducing the number of training views. On Car, reducing to 16, 8, and 4 yields PSNR 33.4, 30.1, and 19.8 respectively (reference: 33.63). On Home, they yield PSNR 34.39, 29.63 and 24.18 (ref: 32.45). So, reducing to 16 or even 8 cameras

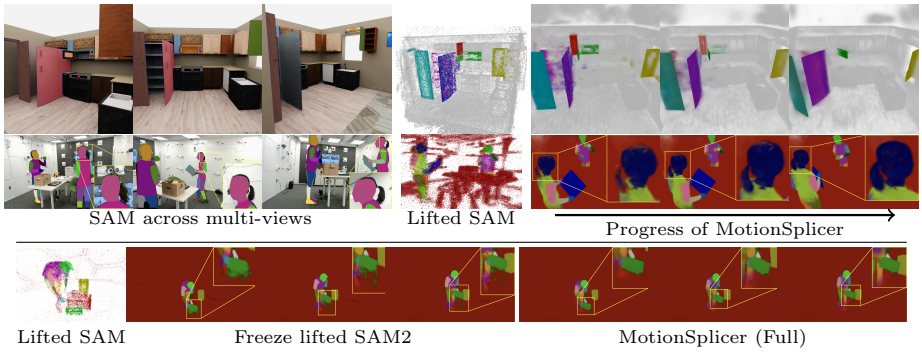


Fig. 7: Lifted SAM without MotionSplicer is insufficient. (Top) MotionSplicer corrects spatio-temporal inconsistencies and under-segmentation errors from lifted SAM to achieve a cleaner 4D motion segmentation. (Bottom) This shows that relying solely on the initial segmentation by freezing it fails during temporal changes, while our full spatio-temporal optimization remains robust to disocclusion.

Fig. 8: 4D integration helps feature lifting be more spatio-temporally robust. Alternative approaches combine a 2D segmentation model and 2D trackers, and lift them into 3D to build skinning weights.

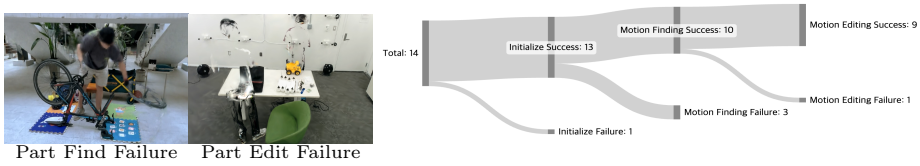


Fig. 9: Failure modes. Visual examples of our method's failure modes and a chart of their frequency.

still achieves an acceptable 30 dB quality. *Anchor-Frame Sensitivity (t):* We evaluated initialization sensitivity on Car. While t=0 yields 33.63 PSNR, t=10 achieves 36.14, and t=20 diverges (NaN). This shows that initialization matters, but performance does not strictly depend on the pose at frame zero.

5 Discussion

Failure Cases Figure 9 shows representative failure cases and their frequency. Part finding can fail when the initial masks are severely under-segmented, such as a wheel segmented together with the spaces between spokes, because U&S may not recover the missing separation without introducing artifacts. Part editing can fail when the observed motion is too small or too entangled with the background;

in such cases, some primitives receive incorrect background weights and removal or isolation leaves residual geometry.

Limitations Although MotionSplicer enables part-level motion editing for complex 4D volumetric videos, it still has several limitations. First, the method assumes synchronized multi-view video and is not designed for monocular or static-camera input. Second, its part discovery depends on the quality of the initial 2D segmentation and the anchor motion; the anchor frame must be reasonably informative. Severe under-segmentation, small observed motion, thin structures, or heavy occlusion can lead to the failure cases analyzed in Figure 9. Third, our LBS-based representation handles moderate non-rigid motion through smooth skinning weights, but highly deformable things such as fluids, loose cloth, hair, or topology-changing motion would require more specialized representations, additional semantic priors, or physical priors. Finally, our optimization is per-scene and can be computationally expensive for long videos or very dense scenes.

Future Work Possible directions are to extend the method toward monocular settings, improve initialization robustness, introduce an editable 4D feed forward model without per-scene optimization. To cope with highly deformable dynamics, future work could incorporate richer deformation representations such as spring bone parameters [21] into LBS parameters (e.g., loose clothes in [8]), consider material-aware dynamics, and use temporal foundation models to improve initialization in low-motion or heavily occluded regions.

6 Conclusion

We present MotionSplicer, a method that enables template-free, part-level motion editing for complex, multi-object 4D scenes. Unlike existing methods that are restricted to either single tabletop objects or part-level rigid motions that rely heavily on monocular video, MotionSplicer successfully handles both rigid and soft boundaries in complex, multi-object real scenes using a single method. Our core technical contribution is a hybrid framework: we first leverage foundation segmentation models on temporal difference images for robust, annotation-free part initialization, followed by our unification and splitting process. Furthermore, rather than relying on static features and simple optimization, we then optimize a time-invariant skinning weight grid using a temporal refinement loss to ensure spatio-temporal consistency across the entire scene.

7 Acknowledgments and Disclosure of Funding

This work was supported by NASA grant #80NSSC23M0075 and ONR DURIP grant N00014-23-1-2804.

References

1. Abou-Chakra, J., Rana, K., Dayoub, F., Suenderhauf, N.: Physically embodied gaussian splatting: A visually learnt and physically grounded 3d representation for robotics. In: 8th Annual Conference on Robot Learning

2. Bar, A., Zhou, G., Tran, D., Darrell, T., LeCun, Y.: Navigation world models. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 15791–15801 (2025)
3. Cen, J., Yu, C., Yuan, H., Jiang, Y., Huang, S., Guo, J., Li, X., Song, Y., Luo, H., Wang, F., et al.: Worldvla: Towards autoregressive action world model. arXiv preprint arXiv:2506.21539 (2025)
4. Chen, G., Liu, J., Du, S., Wu, C., Li, D., Huang, S.S., Zhang, G., Yang, S.: Gs-roadpatching: Inpainting gaussians via 3d searching and placing for driving scenes. arXiv preprint arXiv:2509.19937 (2025)
5. Chen, Q., Qu, D., Tang, Y., Song, H., Zhang, Y., Wang, D., Zhao, B., Li, X.: Freegaussian: Guidance-free controllable 3d gaussian splats with flow derivatives. arXiv preprint arXiv:2410.22070 (2024)
6. Chen, Y., Kwon, H., Inaltekin, H., Gorlatova, M.: Vr viewport pose model for quantifying and exploiting frame correlations. In: IEEE INFOCOM 2022-IEEE Conference on Computer Communications. pp. 1269–1278. IEEE (2022)
7. Chen, Y., Omoma, S., Kwon, H., Inaltekin, H., Gorlatova, M.: Quantifying and exploiting vr frame correlations: An application of a statistical model for viewport pose. IEEE Transactions on Mobile Computing (2024)
8. Cheng, W., Chen, R., Fan, S., Yin, W., Chen, K., Cai, Z., Wang, J., Gao, Y., Yu, Z., Lin, Z., et al.: Dna-rendering: A diverse neural actor repository for high-fidelity human-centric rendering. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 19982–19993 (2023)
9. Choi, S., Song, H., Kim, J., Kim, T., Do, H.: Click-gaussian: Interactive segmentation to any 3d gaussians. In: European Conference on Computer Vision. pp. 289–305. Springer (2024)
10. Cohen, M., Gortler, S.J., Szeliski, R., Grzeszczuk, R., Szeliski, R.: The lumigraph. Association for Computing Machinery, Inc. (August 1996), https://www.microsoft.com/en-us/research/publication/the-lumigraph/
11. Dai, P., Zhang, P., Dong, Z., Xu, K., Peng, Y., Ding, D., Shen, Y., Yang, Y., Liu, X., Lau, R.W., et al.: 4d gaussian videos with motion layering. ACM Transactions on Graphics (TOG) **44**(4), 1–14 (2025)
12. Dodeja, L., Schmeckpeper, K., Vats, S., Weng, T., Jia, M., Konidaris, G., Tellex, S.: Accelerating residual reinforcement learning with uncertainty estimation. IEEE Robotics and Automation Letters **11**(1), 970–977 (2025)
13. Güler, R.A., Neverova, N., Kokkinos, I.: Densepose: Dense human pose estimation in the wild. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7297–7306 (2018)
14. Guo, J., Xin, Y., Liu, G., Xu, K., Liu, L., Hu, R.: Articulatedgs: Self-supervised digital twin modeling of articulated objects using 3d gaussian splatting. arXiv preprint arXiv:2503.08135 (2025)
15. Ha, D., Schmidhuber, J.: Recurrent world models facilitate policy evolution. Advances in neural information processing systems **31** (2018)
16. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. Advances in neural information processing systems **30** (2017)
17. Hu, L., Zhang, H., Zhang, Y., Zhou, B., Liu, B., Zhang, S., Nie, L.: Gaussianavatar: Towards realistic human avatar modeling from a single video via animatable 3d gaussians. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2024)

18. Huang, Y.H., Sun, Y.T., Yang, Z., Lyu, X., Cao, Y.P., Qi, X.: Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4220–4230 (2024)
19. Hung, S.H., Dahlen, F., Nolet, G.: Fréchet kernels for finite-frequency traveltimeséii. examples. Geophysical Journal International **141**(1), 175–203 (2000)
20. Iliash, D., Jiang, H., Zhang, Y., Savva, M., Chang, A.X.: S2o: Static to openable enhancement for articulated 3d objects. arXiv preprint arXiv:2409.18896 (2024)
21. Isozaki, N., Ishima, S., Yamada, Y., Obuchi, Y., Sato, R., Shimizu, N.: Vroid studio: a tool for making anime-like 3d characters using your imagination. In: SIGGRAPH Asia 2021 Real-Time Live!, pp. 1–1 (2021)
22. Jia, M., Huang, H., Zhang, Z., Wang, C., Zhao, L., Wang, D., Liu, J.X., Walters, R., Platt, R., Tellex, S.: Open-vocabulary pick and place via patch-level semantic maps. arXiv preprint arXiv:2406.15677 (2024)
23. Jia, M., Huang, H., Zhang, Z., Wang, C., Zhao, L., Wang, D., Liu, J.X., Walters, R., Platt, R., Tellex, S.: Learning efficient and robust language-conditioned manipulation using textual-visual relevancy and equivariant language mapping. IEEE Robotics and Automation Letters (2025)
24. Jia, M., Wang, D., Su, G., Klee, D., Zhu, X., Walters, R., Platt, R.: Seil: Simulation-augmented equivariant imitation learning. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). pp. 1845–1851. IEEE (2023)
25. Jiang, G., Chang, H., Qiu, R.Z., Liang, Y., Ji, M., Zhu, J., Dong, Z., Zou, X., Wang, X.: Gsworld: Closed-loop photo-realistic simulation suite for robotic manipulation. arXiv preprint arXiv:2510.20813 (2025)
26. Jiang, Y., Yu, C., Xie, T., Li, X., Feng, Y., Wang, H., Li, M., Lau, H., Gao, F., Yang, Y., et al.: Vr-gs: A physical dynamics-aware interactive gaussian splatting system in virtual reality. In: ACM SIGGRAPH 2024 Conference Papers. pp. 1–1 (2024)
27. Jiang, Z., Hsu, C.C., Zhu, Y.: Ditto: Building digital twins of articulated objects from interaction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5616–5626 (2022)
28. Joo, H., Liu, H., Tan, L., Gui, L., Nabbe, B., Matthews, I., Kanade, T., Nobuhara, S., Sheikh, Y.: Panoptic studio: A massively multiview system for social motion capture. In: Proceedings of the IEEE international conference on computer vision. pp. 3334–3342 (2015)
29. Kania, K., Yi, K.M., Kowalski, M., Trzciński, T., Tagliasacchi, A.: Conerf: Controllable neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18623–18632 (2022)
30. Karaev, N., Makarov, I., Wang, J., Neverova, N., Vedaldi, A., Rupprecht, C.: Cotracker3: Simpler and better point tracking by pseudo-labelling real videos. arXiv preprint arXiv:2410.11831 (2024)
31. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Trans. Graph. **42**(4), 139–1 (2023)
32. Kerr, J., Kim, C.M., Wu, M., Yi, B., Wang, Q., Goldberg, K., Kanazawa, A.: Robot see robot do: Imitating articulated object manipulation with monocular 4d reconstruction. arXiv preprint arXiv:2409.18121 (2024)
33. Kim, C.M., Wu, M., Kerr, J., Goldberg, K., Tancik, M., Kanazawa, A.: Garfield: Group anything with radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 21530–21539 (2024)

34. Kim, H., Jang, J.H., Chun, S.Y.: Robust 3d-masked part-level editing in 3d gaussian splatting with regularized score distillation sampling. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5501–5510 (2025)
35. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., et al.: Segment anything. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 4015–4026 (2023)
36. Kratimenos, A., Lei, J., Daniilidis, K.: Dynmf: Neural motion factorization for real-time dynamic view synthesis with 3d gaussian splatting. In: European Conference on Computer Vision. pp. 252–269. Springer (2024)
37. Le, L., Xie, J., Liang, W., Wang, H.J., Yang, Y., Ma, Y.J., Vedder, K., Krishna, A., Jayaraman, D., Eaton, E.: Articulate-anything: Automatic modeling of articulated objects via a vision-language foundation model. arXiv preprint arXiv:2410.13882 (2024)
38. Lei, J., Wang, Y., Pavlakos, G., Liu, L., Daniilidis, K.: Gart: Gaussian articulated template models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 19876–19887 (2024)
39. Li, C., Zhang, R., Wong, J., Gokmen, C., Srivastava, S., Martín-Martín, R., Wang, C., Levine, G., Lingelbach, M., Sun, J., Anvari, M., Hwang, M., Sharma, M., Aydin, A., Bansal, D., Hunter, S., Kim, K.Y., Lou, A., Matthews, C.R., Villa-Renteria, I., Tang, J.H., Tang, C., Xia, F., Savarese, S., Gweon, H., Liu, K., Wu, J., Fei-Fei, L.: BEHAVIOR-1k: A benchmark for embodied AI with 1,000 everyday activities and realistic simulation. In: 6th Annual Conference on Robot Learning (2022), https://openreview.net/forum?id=_8DoIe8G3t
40. Li, T., Slavcheva, M., Zollhoefer, M., Green, S., Lassner, C., Kim, C., Schmidt, T., Lovegrove, S., Goesele, M., Newcombe, R., et al.: Neural 3d video synthesis from multi-view video. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 5521–5531 (2022)
41. Li, X., Cao, Z., Sun, H., Zhang, J., Xian, K., Lin, G.: 3d cinemagraphy from a single image. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4595–4605 (2023)
42. Li, Z., Chen, Z., Li, Z., Xu, Y.: Spacetime gaussian feature splatting for real-time dynamic view synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8508–8520 (2024)
43. Liu, I., Xu, Z., Yifan, W., Tan, H., Xu, Z., Wang, X., Su, H., Shi, Z.: Riganything: Template-free autoregressive rigging for diverse 3d assets. arXiv preprint arXiv:2502.09615 (2025)
44. Liu, J., Iliash, D., Chang, A.X., Savva, M., Mahdavi-Amiri, A.: Singapo: Single image controlled generation of articulated parts in objects. arXiv preprint arXiv:2410.16499 (2024)
45. Liu, J., Mahdavi-Amiri, A., Savva, M.: Paris: Part-level reconstruction and motion analysis for articulated objects. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 352–363 (2023)
46. Luiten, J., Kopanas, G., Leibe, B., Ramanan, D.: Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In: 2024 International Conference on 3D Vision (3DV). pp. 800–809. IEEE (2024)
47. Luo, H., Xu, T., Jiang, Y., Zhou, C., Qiu, Q., Zhang, Y., Yang, W., Xu, L., Yu, J.: Artemis: Articulated neural pets with appearance and motion synthesis. arXiv preprint arXiv:2202.05628 (2022)
48. Ma, S., Luo, Y., Yang, W., Yang, Y.: Mags: Reconstructing and simulating dynamic 3d objects with mesh-adsorbed gaussian splatting. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8745–8755 (2025)

49. Magnenat-Thalmann, N., Laperrière, R., Thalmann, D.: Joint-dependent local deformations for hand animation and object grasping. In: Proceedings on Graphics interface'88. pp. 26–33 (1989)
50. Mandi, Z., Weng, Y., Bauer, D., Song, S.: Real2code: Reconstruct articulated objects via code generation. arXiv preprint arXiv:2406.08474 (2024)
51. Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoorthi, R., Ng, R., Kar, A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM Transactions on Graphics (ToG) **38**(4), 1–14 (2019)
52. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
53. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM transactions on graphics (TOG) **41**(4), 1–15 (2022)
54. Noguchi, A., Iqbal, U., Tremblay, J., Harada, T., Gallo, O.: Watch it move: Unsupervised discovery of 3d joints for re-posing of articulated objects. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3677–3687 (2022)
55. Ost, J., Mannan, F., Thuerey, N., Knodt, J., Heide, F.: Neural scene graphs for dynamic scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2856–2865 (2021)
56. Pavlakos, G., Choutas, V., Ghorbani, N., Bolkart, T., Osman, A.A.A., Tzionas, D., Black, M.J.: Expressive body capture: 3D hands, face, and body from a single image. In: Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 10975–10985 (2019)
57. Qu, D., Chen, Q., Zhang, P., Gao, X., Zhao, B., Wang, Z., Wang, D., Li, X.: Livescene: Language embedding interactive radiance fields for physical scene control and rendering. Advances in Neural Information Processing Systems **37**, 12271–12292 (2024)
58. Qu, Y., Chen, D., Li, X., Li, X., Zhang, S., Cao, L., Ji, R.: Drag your gaussian: Effective drag-based editing with score distillation for 3d gaussian splatting. arXiv preprint arXiv:2501.18672 (2025)
59. Rai, A., Gupta, H., Pandey, A., Carrasco, F.V., Takagi, S.J., Aubel, A., Kim, D., Prakash, A., De la Torre, F.: Towards realistic generative 3d face models. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 3738–3748 (2024)
60. Ravi, N., Gabeur, V., Hu, Y.T., Hu, R., Ryali, C., Ma, T., Khedr, H., Rädle, R., Rolland, C., Gustafson, L., Mintun, E., Pan, J., Alwala, K.V., Carion, N., Wu, C.Y., Girshick, R., Dollár, P., Feichtenhofer, C.: Sam 2: Segment anything in images and videos. arXiv preprint arXiv:2408.00714 (2024), `https://arxiv.org/abs/2408.00714`
61. Shao, Y., Huang, M., Loy, C.C., Dai, B.: Gaussim: Foreseeing reality by gaussian simulator for elastic objects. arXiv preprint arXiv:2412.17804 (2024)
62. Shum, H., Kang, S.B.: Review of image-based rendering techniques. Visual Communications and Image Processing 2000 **4067**, 2–13 (2000)
63. Simonelli, A., Müller, N., Kontschieder, P.: Easy3d: A simple yet effective method for 3d interactive segmentation. arXiv preprint arXiv:2504.11024 (2025)
64. Sohn, B.S., Bajaj, C., Siddavanahalli, V.: Volumetric video compression for interactive playback. Comput. Vis. Image Underst. **96**(3), 435–452 (Dec 2004)

65. Sun, J., Jiao, H., Li, G., Zhang, Z., Zhao, L., Xing, W.: 3dgstream: On-the-fly training of 3d gaussians for efficient streaming of photo-realistic free-viewpoint videos. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20675–20685 (2024)
66. Swaminathan, A., Gupta, A., Gupta, K., Maiya, S.R., Agarwal, V., Shrivastava, A.: Leia: Latent view-invariant embeddings for implicit 3d articulation. In: European Conference on Computer Vision. pp. 210–227. Springer (2024)
67. Uzolas, L., Eisemann, E., Kellnhofer, P.: Template-free articulated neural point clouds for reposable view synthesis. Advances in Neural Information Processing Systems **36**, 31621–31637 (2023)
68. Valevski, D., Leviathan, Y., Arar, M., Fruchter, S.: Diffusion models are real-time game engines. arXiv preprint arXiv:2408.14837 (2024)
69. Vats, S., Zhao, M., Callaghan, P., Jia, M., Likhachev, M., Kroemer, O., Konidaris, G.: Optimal interactive learning on the job via facility location planning. arXiv preprint arXiv:2505.00490 (2025)
70. Vora, A., Nag, S., Zhang, H.: Articulate that object part (atop): 3d part articulation from text and motion personalization. arXiv preprint arXiv:2502.07278 (2025)
71. Wan, D., Lu, R., Zeng, G.: Superpoint gaussian splatting for real-time high-fidelity dynamic scene reconstruction. In: Proceedings of the 41st International Conference on Machine Learning. pp. 49957–49972 (2024)
72. Wan, D., Wang, Y., Lu, R., Zeng, G.: Template-free articulated gaussian splatting for real-time reposable dynamic view synthesis. In: NeurIPS (2024)
73. Wang, Q., Wang, Z., Genova, K., Srinivasan, P.P., Zhou, H., Barron, J.T., Martin-Brualla, R., Snavely, N., Funkhouser, T.: Ibrnet: Learning multi-view image-based rendering. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4690–4699 (2021)
74. Wang, Y., Yang, P., Xu, Z., Sun, J., Zhang, Z., Chen, Y., Bao, H., Peng, S., Zhou, X.: Freetimegs: Free gaussian primitives at anytime anywhere for dynamic scene reconstruction. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 21750–21760 (2025)
75. Wang, Z., Simoncelli, E.P., Bovik, A.C.: Multiscale structural similarity for image quality assessment. In: The thrity-seventh asilomar conference on signals, systems & computers, 2003. vol. 2, pp. 1398–1402. Ieee (2003)
76. Wen, M., Wu, S., Wang, K., Liang, D.: Intergsedit: Interactive 3d gaussian splatting editing with 3d geometry-consistent attention prior. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 26136–26145 (2025)
77. Weng, Y., Wen, B., Tremblay, J., Blukis, V., Fox, D., Guibas, L., Birchfield, S.: Neural implicit representation for building digital twins of unknown articulated objects. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3141–3150 (2024)
78. Wimmer, T., Oechsle, M., Niemeyer, M., Tombari, F.: Gaussians-to-life: Text-driven animation of 3d gaussian splatting scenes. arXiv preprint arXiv:2411.19233 (2024)
79. Wu, G., Yi, T., Fang, J., Xie, L., Zhang, X., Wei, W., Liu, W., Tian, Q., Wang, X.: 4d gaussian splatting for real-time dynamic scene rendering. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 20310–20320 (2024)
80. Wu, M., Huang, H., Kerr, J., Kim, C.M., Zhang, A., Yi, B., Kanazawa, A.: Predict-optimize-distill: A self-improving cycle for 4d object understanding. arXiv preprint arXiv:2504.17441 (2025)

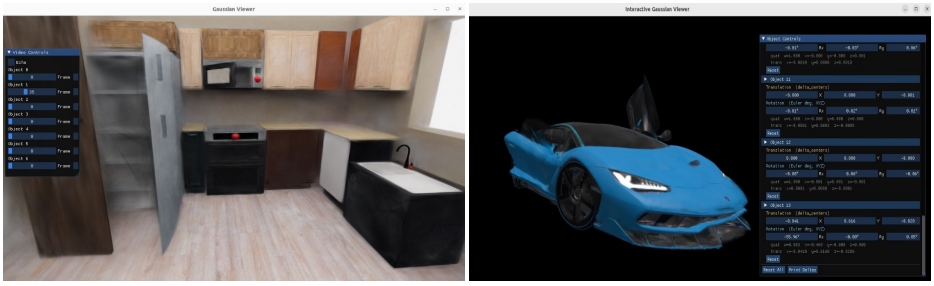
81. Wu, P., Escontrela, A., Hafner, D., Abbeel, P., Goldberg, K.: Daydreamer: World models for physical robot learning. In: Conference on robot learning. pp. 2226–2240. PMLR (2023)
82. Xia, H., Su, E., Memmel, M., Jain, A., Yu, R., Mbiziwo-Tiapo, N., Farhadi, A., Gupta, A., Wang, S., Ma, W.C.: Drawer: Digital reconstruction and articulation with environment realism. arXiv preprint arXiv:2504.15278 (2025)
83. Xie, Y., Takikawa, T., Saito, S., Litany, O., Yan, S., Khan, N., Tombari, F., Tompkin, J., Sitzmann, V., Sridhar, S.: Neural fields in visual computing and beyond. In: Computer Graphics Forum. vol. 41, pp. 641–676. Wiley Online Library (2022)
84. Xu, H., Zhang, J., Cai, J., Rezatofighi, H., Tao, D.: Gmflow: Learning optical flow via global matching. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 8121–8130 (2022)
85. Xu, X., Ruan, Y., Sridhar, S., Ritchie, D.: Unsupervised kinematic motion detection for part-segmented 3d shape collections. In: ACM SIGGRAPH 2022 Conference Proceedings. pp. 1–9 (2022)
86. Xu, Z., Peng, S., Lin, H., He, G., Sun, J., Shen, Y., Bao, H., Zhou, X.: 4k4d: Real-time 4d view synthesis at 4k resolution. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 20029–20040 (2024)
87. Xu, Z., Xu, Y., Yu, Z., Peng, S., Sun, J., Bao, H., Zhou, X.: Representing long volumetric video with temporal gaussian hierarchy. ACM Transactions on Graphics (TOG) **43**(6), 1–18 (2024)
88. Yang, G., Vo, M., Neverova, N., Ramanan, D., Vedaldi, A., Joo, H.: Banmo: Building animatable 3d neural models from many casual videos. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2863–2873 (2022)
89. Yang, Z., Chen, Y., Wang, J., Manivasagam, S., Ma, W.C., Yang, A.J., Urtasun, R.: Unisim: A neural closed-loop sensor simulator. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1389–1399 (2023)
90. Yao, C.H., Raj, A., Hung, W.C., Rubinstein, M., Li, Y., Yang, M.H., Jampani, V.: Artic3d: Learning robust articulated 3d shapes from noisy web image collections. Advances in Neural Information Processing Systems **36**, 48173–48184 (2023)
91. Yao, Y., Deng, Z., Hou, J.: Riggs: Rigging of 3d gaussians for modeling articulated objects in videos. arXiv preprint arXiv:2503.16822 (2025)
92. Ye, M., Danelljan, M., Yu, F., Ke, L.: Gaussian grouping: Segment and edit anything in 3d scenes. In: European Conference on Computer Vision. pp. 162–179. Springer (2024)
93. Yu, H., Julin, J., Milacski, Z.Á., Niinuma, K., Jeni, L.A.: Cogs: Controllable gaussian splatting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 21624–21633 (2024)
94. Zhang, J., Liu, X., Ye, X., Zhao, F., Zhang, Y., Wu, M., Zhang, Y., Xu, L., Yu, J.: Editable free-viewpoint video using a layered neural representation. ACM Transactions on Graphics (TOG) **40**(4), 1–18 (2021)
95. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 586–595 (2018)
96. Zhang, T., Yu, H.X., Wu, R., Feng, B.Y., Zheng, C., Snavely, N., Wu, J., Freeman, W.T.: Physics-based interaction with 3d objects via video generation
97. Zhou, S., Chang, H., Jiang, S., Fan, Z., Zhu, Z., Xu, D., Chari, P., You, S., Wang, Z., Kadambi, A.: Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 21676–21685 (2024)

98. Zhou, X., Lin, Z., Shan, X., Wang, Y., Sun, D., Yang, M.H.: Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 21634–21643 (2024)
99. Zitnick, C.L., Kang, S.B., Uyttendaele, M., Winder, S., Szeliski, R.: High-quality video view interpolation using a layered representation. ACM transactions on graphics (TOG) **23**(3), 600–608 (2004)

MotionSplicer: Part-Based Motion Editing for 4D Volumetric Videos

Supplementary Material

A Viewer for Interactive Motion Editing



Motion Editing Viewer Spatial Editing Viewer

Fig. 10: Interactive viewers. We provide a couple of real-time viewers for motion editing and spatial editing, respectively. Users can intuitively edit part-level motion and try novel transformation on their own. These are shown in the video and will be open-sourced upon acceptance.

As shown in Figure 10, we implement a couple of interactive viewers that enables real-time, part-level motion manipulation in 4D scenes. One is motion editing viewer and the other one is spatial editing viewer. The motion editing viewer exposes a set of sliders for each individual motion part. Adjusting a slider updates the frame index of the associated handle transformation, with additional options for freezing or hiding parts while playing the edited motion, providing intuitive control over part trajectories. The spatial editing viewer provides 6-DoF handles of each part, so that the user can freely manipulate each part. All renderings are performed via the differentiable rasterizer from 3DGS [31].

Our viewers are real-time because they only require a canonical set of 3DGS primitives, as well as our learned per-frame handle transforms and skinning weight grid. Note that only the handle transformations are time-varying, with both the canonical Gaussians and skinning weight grid being stored once and reused for the entire sequence. **Demo videos of the viewer are included in the supplementary video.** All source code (including the viewer) will be released upon acceptance.

B Additional Experimental Details

Datasets As shown in Table 4, we used multi-view streams, where up to 3 views were set aside for test views. Our captured dataset includes synchronized 360-degree real-world videos recorded with 85 cameras, featuring humans interacting

Source	Sequence	Res.	Views	Frames	Splats	Parts	Fwd.	Bwd.	FPS	Storage
Sketch -fab	Turtle	800	27+3	60	13,305	6	2.92	4.18	1603	17M+126K
	Car	800	27+3	40	90,780	14	5.39	8.51	812	22M+84K
	Skater	800	27+3	42	49,244	17	3.06	4.29	745	19M+88K
[39]	Home	1024	28+2	100	381,428	7	9.80	23.89	159	41M+210K
	Kitchen	1024	28+2	100	1,281,893	5	30.86	53.35	80	100M+210K
[28]	Softball	360p	27+4	100	360,054	18	15.22	25.85	420	40M+210K
	Boxes	360p	27+4	150	376,486	18	15.83	27.03	429	41M+315K
	Basketball	360p	27+4	150	400,646	31	23.09	45.65	413	42M+315K
Custom	Tasting	FHD	84+1	45	993,393	32	66.44	155.38	139	81M+95K

Table 4: Data Statistics and Computational Costs. # **Views** sums train and test views. # **Parts** is the count of parts after Unify and Split. **Fwd./Bwd.** are forward and backward times in milliseconds. **Storage** combines the sizes of canonical components (canonical GS and feature grid) and deformation components (handles) in bytes.

with multiple objects. For Panoptic dataset [28], we followed the convention of Dynamic3DGS [46] for test view split. For synthetic datasets, we used Blender and Isaac Sim to obtain multi-view video inputs from Sketchfab objects and OmniGibson [39] scenes, respectively. We only required RGB videos without using any additional labels or known geometry. For Figure 9, we additionally used forward-facing datasets, including two LongVolcap [87] sequences and a DyNeRF [40] sequence.

Implementational Details We used learning rate of 10^{-5}, 10^{-3}, and 10^{-6} for optimizing the part-level translation T_p, rotation R_p, and the parameters of feature grid for skinning weight w_i, respectively. All computational times in Table 4 are measured at a RTX 3090 GPU. We performed 6000 iterations for each frame during optimization. In practice, during optimization, we could use up to 16 GPUs, which allows optimization with approximately 2.5 minutes per frame without Temporal Consistency (TC) refinement. With TC refinement enabled, the training time increases roughly in proportion to the number of previous frames randomly selected for refinement at each iteration. We set this number to either 3 or 4, depending on the available VRAM size. TC refinement is not needed at inference at all. Note that the forward and backward times listed in Table 4 are measured without the TC refinement. When TC refinement was employed, we optionally used an RTX A6000 GPU for non-object scenes during optimization. We set the balance factor λ to 0.8. For the skinning weights, we employed a 16-level multi-resolution feature grid starting from a base resolution of 16, increasing with a factor of 2 at each level. The features have 8 dimensions and are processed via hash encoding [53] followed by a 2-layer MLPs with 64 hidden channels.

Computational Costs The rendering FPS is measured during inference and accounts for the entire pipeline, including feature grid querying, handle application, linear blend skinning, and rasterization. It does not include the file I/O time. Regarding storage, the canonical primitives occupy between 884 KB and 85 MB of memory, depending on the scene's complexity, while the weight

		PSNR ↑	SSIM ↑	LPIPS ↓	T-LPIPS ↓	KVD ↓	Flow-warp ↓	GPT-EE↑
(a)	SC-GS [18]	43.22	0.997	0.013	0.0027	0.9773	0.300	4.8
	ArtiGS [14]	28.06	0.929	0.091	Failed	1.2311	Failed	4.8
	RigGS [91]	34.30	0.988	0.034	0.0029	0.7893	0.696	5.0
	RSRD [32]	19.04	0.906	0.070	0.0109	0.8278	0.000	4.8
	MotionSplicer	39.25	0.987	0.034	0.0018	0.7594	0.505	5.0
(b)	SC-GS [18]	38.71	0.995	0.014	0.0090	0.7761	1.354	4.8
	ArtiGS [14]	15.86	0.775	0.166	0.0072	0.8330	0.474	2.5
	RigGS [91]	28.38	0.980	0.047	0.0004	0.7276	7.528	4.8
	RSRD [32]	Failed	Failed	Failed	Failed	Failed	Failed	Failed
	MotionSplicer	33.63	0.983	0.043	0.0296	0.4904	1.351	4.5
(c)	SC-GS [18]	38.19	0.987	0.023	0.0119	1.0945	1.229	1.5
	ArtiGS [14]	37.31	0.985	0.019	0.0031	0.5505	0.170	4.5
	RigGS [91]	32.82	0.980	0.061	0.0036	0.7164	0.398	4.5
	RSRD [32]	35.48	0.992	0.010	0.0212	0.7669	0.000	4.2
	MotionSplicer	36.68	0.984	0.017	0.0014	0.5836	0.162	4.8
(d)	SC-GS [18]	26.82	0.930	0.211	0.0033	1.0945	0.744	1.0
	ArtiGS [14]	28.52	0.927	0.183	0.0061	0.7226	0.184	1.0
	RigGS [91]	28.51	0.969	0.260	0.0192	0.9701	2.609	4.5
	RSRD [32]	28.72	0.935	0.093	0.0096	0.8925	0.353	4.5
	MotionSplicer	32.45	0.961	0.051	0.0030	0.7321	0.578	4.5
(e)	SC-GS [18]	37.77	0.988	0.066	0.0035	1.5102	0.345	1
	ArtiGS [14]	24.92	0.934	0.146	0.0042	0.8446	0.345	4.5
	RigGS [91]	31.28	0.974	0.266	0.0093	0.9754	1.899	4.5
	RSRD [32]	35.78	0.961	0.085	0.0239	0.9624	0.524	1.5
	MotionSplicer	36.54	0.973	0.035	0.0014	0.6693	0.303	4.8
(f)	SC-GS [18]	22.11	0.697	0.431	0.0125	1.0039	29.198	2.0
	ArtiGS [14]	18.57	0.728	0.509	Failed	1.0760	Failed	2.5
	RigGS [91]	26.16	0.840	0.323	0.0211	0.9792	20.474	2.0
	RSRD [32]	18.548	0.867	0.460	0.0488	0.8629	19.588	1.5
	MotionSplicer	27.98	0.903	0.129	0.0070	0.4602	19.634	4.8
(g)	SC-GS [18]	20.83	0.668	0.463	Failed	1.0288	Failed	2.0
	ArtiGS [14]	18.28	0.690	0.523	0.0494	0.9125	17.441	1.5
	RigGS [91]	26.07	0.836	0.318	0.1152	0.8922	20.142	1.5
	RSRD [32]	17.46	0.665	0.395	0.0246	1.1063	32.598	2.0
	MotionSplicer	27.03	0.881	0.146	0.0007	0.8682	19.835	4.5
(h)	SC-GS [18]	22.02	0.685	0.435	Failed	1.0790	Failed	1.5
	ArtiGS [14]	17.98	0.660	0.546	Failed	1.1159	Failed	1.0
	RigGS [91]	26.26	0.836	0.327	0.1691	0.8943	27.239	1.5
	RSRD [32]	20.95	0.689	0.376	0.0627	1.0477	20.594	1.5
	MotionSplicer	27.32	0.900	0.130	0.0074	0.8847	20.221	4.8
(i)	SC-GS [18]	23.25	0.871	0.267	Failed	0.9347	Failed	2.5
	ArtiGS [14]	10.60	0.636	0.529	Failed	1.1892	Failed	1.5
	RigGS [91]	27.12	0.899	0.352	0.0187	1.1922	56.810	4.8
	RSRD [32]	Failed	Failed	Failed	Failed	Failed	Failed	Failed
	MotionSplicer	27.10	0.884	0.269	0.0106	1.0213	66.631	4.2

Table 5: Detailed quantitative comparison. (a) Turtle, (b) Car, (c) Skater, (d) Home, (e) Kitchen, (f) Softball, (g) Boxes, (h) Basketball, (i) Tasting.

grid consistently occupies 16 MB, independent of scenes. Beyond the canonical frame, the additional memory cost per frame is lightweight ($\sim$ 2.1 KB), consisting of only 7 real parameters (3D translation and quaternion). Pre-processing steps, which occur only once, take up to 62 seconds for segmentation and approximately 40 minutes for 3D lifting.

Additional details on evaluation metrics For GPT-EE score, we use GPT-4o (temperature 0.0) as an automated judge. For each video, 10 frames are uniformly sampled from the first second, arranged into a 2×5 grid (max 256px per frame, JPEG quality 95), and the same is done for the original captured frames. Both grids are jointly submitted with the following system prompt: *"Evaluate the Editing Effect (EE): Did the motion follow the instruction? Is the magnitude sufficient? Score on a [1.0, 5.0] float scale, where 5.0 is perfect ground-truth equivalence and 1.0 is complete failure. Avoid whole-number scores."* For KVD, per-frame features are extracted using DINOv2 ViT-B/14, taking the CLS token (d=768) after resizing to 256 and center-cropping to 224 x 224. For Flow-Warp, RAFT-Large estimates forward and backward flow for each consecutive frame pair for occlusion masking. Pixels are masked as occluded when the forward-backward consistency error exceeds τ=3.0 px. The score is the mean L1 photometric error over non-occluded pixels. T-LPIPS used AlexNet backbone.

C Detailed Quantitative Results

As shown in Table 5, we compare our method against state-of-the-art motion-editable representations [14,18,32,91]. Existing methods often rely on constrained representation and optimization only: SC-GS [18] utilizes control points with skinning weights, optimized via ARAP, ArticulatedGS [14] predicts axis and motion around the axis, and RigGS [91] assumes hierarchical skeleton. Robot-see-robot-do (RSRD) [32] relies on static features and does not use soft boundary representation such as blend skinning. In contrast, we leverage foundational models to initialize the part-based non-rigid motion and adaptively optimize the handles, allowing independent flexibility for each handle without relying on fixed priors like rigs, axes, or table-top assumptions. Furthermore, our approach ensures spatial consistency via the feature grid and enhances temporal consistency through the TC refinement process. Some baselines fail to reanimate anything, and thus no valid temporal consistency metrics can be calculated for them (T-LPIPS, Flow-Warp). RSRD sometimes fails to cluster parts, which is an inevitable step for that method. In that case, we report it as 'fail'. ArticulatedGS and SC-GS produced misplaced axes and noisy control points in real scenes, making parts barely movable and T-LPIPS/Flow-warp unmeasurable. Blank metrics for RSRD indicate segmentation failures: when running their official code, their clustering eliminated all objects which made tracking impossible. In summary, while SC-GS yields higher metrics on single synthetic objects, our approach mostly outperforms the baselines, especially on scenes. Notably, in complex multi-object scenarios, especially in real-world scenes, our method outperforms all baselines by a significant margin.

D User Study Details

To complement the summary chart of the user study in the main paper, we elaborate detailed numbers and instructions in Table 6 and Table 7. The user study was conducted via an anonymous online questionnaire. Participants are asked to report their background by answering the following question: *"How familiar are you with Computer Vision, 3D Graphics, or Video Editing?"*, where we confirm the inclusive expertise level that is average of 3.1 and SD 1.33.

- **1 (No experience):** I have little to no knowledge in these fields.
- **2 (Beginner):** I understand basic concepts, or took an introductory undergraduate class.
- **3 (Intermediate):** I have taken advanced courses or completed course projects in these fields.
- **4 (Advanced):** I am/was a researcher or a professional in these fields in general.
- **5 (Expert):** I am currently specializing in these fields as a researcher or as a job.

During the evaluation, all rendered videos were presented in infinite looping without time limit, so that participants can take their time to compare the 4D edited motion details. Finally, the statistical significance ($p < 0.001$) was calculated using paired t-test.

Instructions			
Realism & Artifact	*Which one looks more realistic with fewer visual artifacts?*		
Smooth Motion	*Which one shows more natural and coherent movement?*		
Local Editing	*Which one better affect only the target part without distorting the background or other parts?*		
Voting Results			
	Realism & Artifact |	Smooth Motion |	Local Editing
Ours is much better	76	88	83
Ours is slightly better	21	17	11
Similar	9	5	11
Baseline is slightly better	7	4	10
Baseline is much better	6	6	5
Total	119	120	120

Table 6: User study's relative comparison details. Note that the participants are given anonymized comparisons, i.e., they did **not** see the words 'Ours' nor 'Baseline', but they are only given randomized left or right.

E Details of Unify and Split

We provide a detailed description of our unification and splitting strategy (adaptive part refinement), that is briefly stated in the main paper. This process

	Baselines	Ours
Average	1.75	4.28
SD	1.12	0.90

Instruction: *Does each video successfully match the task stated above the videos?*

Table 7: User study's absolute comparison details. The participants were asked to give score [1,5] to both ours and randomized baseline, both anonymized in randomized order.

Algorithm 1 Unify and Split (Adaptive Part Refinement).

Require: $\mathcal{P} = \{\mathbf{p}_p = (\mathbf{t}_p, \mathbf{r}_p, \mathbf{c}_p)\}_{p=1}^{P}, W, iter$
Require: $\tau_{pos}, \tau_{rot}, \tau_{grad}$
Ensure: $\mathbf{t}_p \in \mathbb{R}^3, \mathbf{r}_p \in \mathbb{R}^4, \mathbf{c}_p \in \mathbb{R}^3, w_p \in \mathbb{R}^N, W \in \mathbb{R}^{N \times P}, \mu_n \in \mathbb{R}^3$. N is number of Gaussians.

if $(iter \mod iter_{refine}) = iter_{refine} - 1$ **then**
  // Step 1: Unifying redundant handles
  for $i = 1 : P$ **do**
    $\mathcal{R}_i = \{\mathbf{p}_j \mid (||\mathbf{t}_j - \mathbf{t}_i||_2 < \tau_{pos}) \cap (\angle(\mathbf{r}_j - \mathbf{r}_i) < \tau_{rot})\}$ ▷ Identify redundant parts
    if $|\mathcal{R}_i| > 0$ **then**
      $\mathcal{R} \leftarrow \mathcal{R}_i \cup \{\mathbf{p}_i\}$
      $\mathbf{p}_{new}$: ▷ Create a new handle
        $\mathbf{t}_{new} \leftarrow \frac{1}{|\mathcal{R}|} \sum_k^{|\mathcal{R}|} \mathbf{t}_k,\ \mathbf{r}_{new} \leftarrow \text{normalize}(\frac{1}{|\mathcal{R}|} \sum_k^{|\mathcal{R}|} \mathbf{r}_k)$
        $\mathbf{c}_{new} \leftarrow \text{median}(\mu \text{ of } \mathcal{R}\text{'s assigned Gaussians})$
      $w_{new} \leftarrow \max_k(w_k \text{ of } \mathcal{R}\text{'s assigned Gaussians})$ ▷ Update weights
      $\mathcal{P} \leftarrow (\mathcal{P} \backslash \mathcal{R}) \cup \{\mathbf{p}_{new}\}$
    end if
  end for
  // Step 2: Splitting under-segmented handles
  $\mathcal{S} = \{\mathbf{p}_k \mid \sum \nabla_{w_k} > \tau_{grad}, \mathbf{p}_k \in \mathcal{P}\}$ ▷ Identify split candidates
  for $k = 1 : |\mathcal{S}|$ **do**
    Principal axis $\mathbf{a}$ and scale l. $\mathbf{a}, l \leftarrow \text{PCA}(\mu \text{ of } \mathbf{p}_k\text{'s assigned Gaussians})$
    $\mathbf{p}_{left},\ \mathbf{p}_{right}$: ▷ Create two new handles
      $\mathbf{c}_{left} \leftarrow \mathbf{c}_k - l \cdot \mathbf{a},\ \mathbf{c}_{right} \leftarrow \mathbf{c}_k + l \cdot \mathbf{a}$
      $\mathbf{t}_{left} \leftarrow \mathbf{t}_k,\ \mathbf{r}_{left} \leftarrow \mathbf{r}_k,\ \mathbf{t}_{right} \leftarrow \mathbf{t}_k,\ \mathbf{r}_{right} \leftarrow \mathbf{r}_k$
    if $(\mu - \mathbf{c}_k) \cdot \mathbf{a} < 0$ **then** ▷ Update weights
      $w_{left} \leftarrow w_k$
    else
      $w_{right} \leftarrow w_k$
    end if
    $\mathcal{P} \leftarrow (\mathcal{P} \backslash \{\mathbf{p}_k\}) \cup \{\mathbf{p}_{left}, \mathbf{p}_{right}\}$
  end for
end if
return $\mathcal{P}, W$

ensures that the motion parts are neither over-segmented nor under-segmented, adapting to the complexity of the scene motion.

We denote the set of part handles as $\mathcal{P} = \{\mathbf{p}_p\}_{p=1}^{P}$, where each handle $\mathbf{p}_p$ is associated with a translation $\mathbf{t}_p \in \mathbb{R}^3$, a rotation $\mathbf{r}_p \in \mathbb{R}^4$ in quaternion, and a canonical center $\mathbf{c}_p \in \mathbb{R}^3$. The skinning weights for the Gaussian primitives are denoted by w, where the weight matrix for the entire scene is defined as $W \in \mathbb{R}^{N \times P}$. N is the number of Gaussian primitives. The unification and splitting process involves a few hyper-parameters: τ_{pos} for distance threshold of merging handles, τ_{rot} for angular threshold of merging handles, ∇_W for accumulated gradients on the skinning weights, used as a probe for under-segmentation, and τ_{grad} for gradient threshold of identifying under-segmented parts by the magnitude of the gradient ∇_W.

The complete procedure is presented in Algorithm 1. **Step 1 (Unifying redundant handles)**: We iteratively check pairs of handles. If two handles have similar spatial positions and rotation trajectories after optimization iterations of $iter_{refine}$, as probed below thresholds τ_{pos} and τ_{rot}, they are considered redundant. These handles are merged into a single handle with averaged transformation parameters to prevent over-segmentation. **Step 2 (Splitting under-segmented handles)**: Conversely, we monitor the gradients of the skinning weights. A high gradient accumulation ($\sum \nabla_{W_k} > \tau_{grad}$) indicates that a single handle is trying to model complex, non-rigid motion that requires more degrees of freedom. In such cases, we split the handle into two along its principal axis, computed via PCA on the assigned Gaussians' means. This densifies the number of parts in the scene, avoiding under-segmentation.