Supplementary Material
StructureGS: Structure-aware Gaussian Splatting for Articulated Object Reconstruction

Gahye Lee, Gyoonseo Kim, Wonjong Jang, Jooeun Son, and Seungyong Lee

POSTECH, South Korea
{gahye0509, nevermore, wonjong, jeson, leesy}@postech.ac.kr

In this supplementary material, we provide additional details on implementation, robustness analysis for OBB initialization, and further experimental results.

1 Implementation Details

1.1 Training configuration

We optimize our model using the Adam optimizer [5] for 10,000 iterations. For optimizing 3D Gaussian Splatting (3DGS) parameters, we use the same learning rates as in the original 3DGS framework [3], except for the Gaussian means, which use a learning rate of 1.6e-2. The canonical-to-world transformation T_0 and the articulation transformation T_1 are jointly optimized with a learning rate of 1e-2. We initialize 10,000 Gaussian primitives for each OBB by uniformly sampling their positions within the canonical space $[-1, 1]^3$.

We use the following loss weights by default. The rendering loss weight is set to $\lambda_{\text{image}} = 10.0$. The part fitting loss consists of a coverage loss and a margin loss, with $\lambda_{\text{coverage}} = 100.0$ and $\lambda_{\text{margin}} = 50.0$, respectively. For the contact loss, $\lambda_{\text{contact}} = 0.1$. The extent regularizer is set with $\lambda_{\text{ext}} = 0.01$.

We optimize StructureGS using a loss schedule. The structure-aware losses and the extent regularizer are activated from the beginning to guide the initial construction of part structure. Starting from iteration 400, we enable the rendering loss to jointly optimize part appearance and geometry. Once the part structure stabilizes, the contact loss and extent regularizer are disabled at iteration 1,500, and subsequent refinement is performed using the rendering and part fitting losses.

All experiments were conducted on a workstation with an NVIDIA GeForce RTX 4090 GPU and an Intel Core i9-12900K CPU, using PyTorch 2.0.1 [10] with CUDA 11.8. We use the differentiable 3D Gaussian Splatting renderer provided by the `gsplat` [14] library. The average training time per scene is approximately 13 minutes, varying with scene complexity.

1.2 Densification strategy

We adopt the densification strategy of 3D Gaussian Splatting [3] during training. All Gaussians from all OBBs are rendered jointly, and Gaussians are split or

duplicated based on the rendering error and opacity, following the original 3DGS framework [3]. To preserve Gaussian-to-OBB membership, new Gaussians are assigned to the same OBB as their parent Gaussian.

1.3 OBB Initialization

We initialize OBB parameters using coarse point clouds estimated from the two articulation states. Given point clouds $\mathcal{P}_0$ and $\mathcal{P}_1$ reconstructed by VGGT [12], we compute a nearest-neighbor motion cue for each point $\mathbf{p}_i \in \mathcal{P}_0$:

$$d_i = \min_{\mathbf{q}_j \in \mathcal{P}_1} |\mathbf{p}_i - \mathbf{q}_j|_2. \tag{1}$$

We classify points with small motion cues as static and use the remaining points as moving candidates. Specifically, we threshold the distances d_i by τ_{motion} to separate static and moving candidate points.

The moving candidate points are clustered using DBSCAN [1]. The resulting clusters are treated as moving-part hypotheses. If the number of clusters is larger than the expected number of moving parts, we keep the dominant clusters by size and remove small noisy clusters. We then fit an oriented bounding box to each part hypothesis, including the static region. These fitted boxes initialize the OBB transforms T_0^k. The articulation transforms T_1^k are initialized to identity for all parts.

The initialization is intentionally coarse. Its role is to place the oriented boxes near plausible object parts, rather than to provide accurate segmentation. During optimization, the Gaussian primitives, OBB parameters, and articulation parameters are jointly refined by photometric and structure-aware losses.

1.4 Joint type and parameter estimation

The motion of each part is obtained from its optimized part-wise OBB transformation $\mathcal{T}_1^k$. For transformation $\mathcal{T}_1^k$, we first determine the joint type and then recover the corresponding joint parameters.

To determine the joint type, we compute the rotation angle from the rotational component of $\mathcal{T}_1^k$. A part is classified as *prismatic* if the rotation angle is below a threshold of 5° (i.e., $\theta < 0.087$ radians), and *revolute* otherwise.

For prismatic joints, we estimate the motion axis by normalizing the translation vector. The joint displacement is measured as the ℓ_2 norm of the translation. For revolute joints, we extract the rotation axis by converting the rotational component to axis-angle form. The pivot point corresponds to the fixed point of the rigid transformation, i.e., a point that remains unchanged under the transformation. We compute the pivot point as the minimum-norm solution to a linear system derived from $(R - I)\, p = -t$, where R and t are the rotation and translation components of $\mathcal{T}_1^k$.

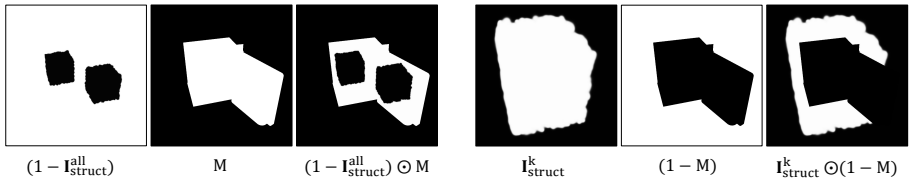

Fig. 1: Visualization of the error maps and intermediate terms of the part fitting loss. M denotes the binary ground-truth object mask, $\mathbf{I}^k_{\text{struct}}$ the occupancy map of the k-th part OBB, and $\mathbf{I}^{\text{all}}_{\text{struct}}$ their union. (a) and (b) illustrate two example cases where the coverage loss and the margin loss are non-zero, respectively. In (a), the error map is formed by multiplying the complement of the aggregated structure image $(1-\mathbf{I}^{\text{all}}_{\text{struct}})$ with the object mask M, yielding $(1-\mathbf{I}^{\text{all}}_{\text{struct}})\odot M$, which indicates object regions not covered by the union of OBBs. In (b), the error map is formed by multiplying the per-part structure image $\mathbf{I}^k_{\text{struct}}$ with the region outside the object mask $(1-M)$, yielding $\mathbf{I}^k_{\text{struct}}\odot(1-M)$, which indicates OBB regions outside the object.

1.5 Error map for the part fitting loss

To provide an intuitive understanding of the part fitting loss $\mathcal{L}_{\text{fit}}$ that consists of the coverage loss $\mathcal{L}_{\text{coverage}}$ and the margin loss $\mathcal{L}_{\text{margin}}$, we visualize the error maps of the individual loss terms, together with the structure image and object mask used to define the loss terms.

The coverage loss becomes active when regions inside the object mask are not covered by the union of OBBs. In Fig. 1a, we show the two intermediate terms, the complement of the all-part structure image $(1-I^{\text{all}}_{\text{struct}})$ and the object mask M, together with the resulting error map obtained by their element-wise product. The error map highlights uncovered regions inside the object mask, encouraging the OBBs to fully cover the object.

The margin loss encourages each OBB to tightly fit the object. In Fig. 1b, we visualize the part structure image I^k_{struct} and the complement of the object mask $1-M$, together with the resulting error map obtained by their element-wise product. This error map captures the empty space inside the OBB that is not occupied by the object.

2 Robustness to OBB initialization

To initialize the OBB parameters, we first obtain point clouds of the target object in the initial and articulated states using VGGT [12]. We then cluster the points into static and dynamic parts based on the distances between corresponding points in the two states, and fit an OBB to each cluster. This initialization provides only coarse and noisy structural information, as it depends on imperfect

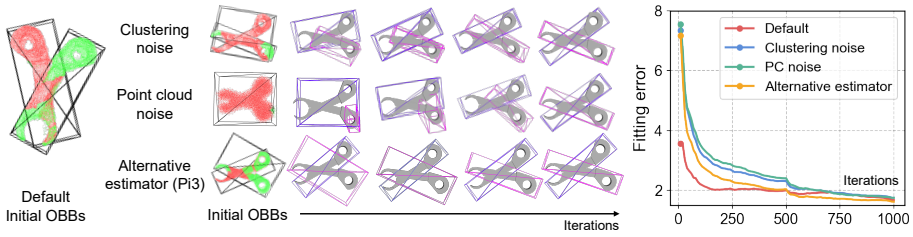

Fig. 2: Left: Default initial OBBs obtained from motion-based point clustering, where point colors indicate clustered part points. Middle: OBB optimization results over iterations under clustering noise, point cloud noise, and an alternative point cloud estimator. Right: Fitting error curves over optimization iterations for the default and perturbed settings.

point cloud estimation and point clustering. Nevertheless, our optimization remains robust to such noisy initialization and can still recover a reliable solution. We examine this robustness in the following experiment.

We evaluate robustness under three settings with 10 sparse views: clustering noise, point cloud noise, and an alternative point cloud estimator. Clustering noise is introduced by varying the threshold used to separate static and dynamic points across the two states, which reduces the accuracy of part segmentation and thus degrades the OBB initialization. Point cloud noise is introduced by adding Gaussian perturbations to the points estimated by VGGT [12] before performing part clustering for OBB initialization. Finally, we replace the default point estimator (VGGT) with a recent alternative method (Pi3 [13]) to evaluate the robustness of our method to different point estimation methods. These perturbations produce OBB initializations of varying quality, ranging from reasonable part-wise boxes to severely imbalanced cases. For example, a single OBB covers most of the object while the remaining boxes capture only small fragments, as shown in the "Point cloud noise" of Fig. 2.

Fig. 2 shows the initial OBBs and their evolution during optimization under these perturbation settings, together with the corresponding fitting error curves. Although the initial OBB quality varies substantially, the fitting error decreases during optimization and converges to similar values across all settings. Qualitatively, the OBBs progressively align with the object parts even when starting from different initializations. These results demonstrate that our method can recover reliable part-aligned structures from inaccurate OBB initializations. Moreover, the comparable convergence obtained with Pi3 [13] indicates that our method does not rely on a specific point cloud estimator.

3 Additional Experimental Results

3.1 Per-scene quantitative comparison on the PARIS dataset

In the main paper, we report the average performance on the PARIS synthetic benchmark. Here, for completeness, we provide the per-scene quantitative results under dense-view and sparse-view settings, respectively (Table 2).

Among the prior methods listed in Table 1 of the main paper, we compare our method only with those evaluated under comparable settings and metrics. Since our method takes multi-view RGB images of two states as input and does not assume known joint types, methods based on different input modalities or problem settings are excluded from the comparison. Specifically, A-SDF [9] is a data-driven learning-based method, REACTO [11] requires monocular RGB video sequences, ArtGS [8] additionally relies on depth input, and SPLART [6] assumes known joint types.

3.2 Additional qualitative comparison on the PARIS dataset

We also present additional qualitative comparisons on the PARIS dataset under both full-view and sparse-view settings. For each example, we reconstruct static and dynamic part meshes and visualize the estimated articulation axes.

As discussed in the main paper, existing methods generally exhibit degraded performance under sparse-view setting compared with the full-view setting. In contrast, our method remains robust in both sparse-view and full-view settings, achieving reliable part decomposition and articulation estimation. We provide a gallery of reconstructed part meshes in Fig. 4 to further illustrate the geometric fidelity of the recovered results.

3.3 Fitted OBBs and reconstructed 3DGS on the PARIS dataset

Fig. 6 shows additional results of our method on scenes from the PARIS benchmark. We visualize the fitted OBBs and reconstructed 3D Gaussians for each part. Across all scenes, the jointly optimized Gaussians capture geometric details, while the OBBs represent the underlying part structure. The results also demonstrate that our method generalizes well to parts with non-cuboidal geometries, as discussed in Section 5.4 of the main paper. For example, *Scissors*, *Stapler*, and *Fridge* scenes include highly concave regions.

3.4 Scalability to multiple moving parts on ArtGS-Multi dataset

To evaluate scalability beyond objects with a single moving part, we conduct additional experiments on ArtGS-Multi [8], which contains articulated objects with 4 to 7 parts. Since ArtGS-Multi provides RGB-D observations, we use the provided depth only to construct the initial point clouds for OBB initialization, and do not use depth during optimization. We compare our method with ScrewSplat [4], which supports reconstruction of articulated objects with

Table 1: Quantitative results on the ArtGS-Multi dataset [8]. We compare with ScrewSplat [4], a baseline that supports articulated reconstruction with multiple moving parts. Ang Err and Pos Err evaluate joint-axis accuracy, GD-r and GD-p measure motion errors for revolute and prismatic motions, respectively, and CD-s/CD-m/CD-w report Chamfer-L1 distances for the static base, movable parts, and whole object. Lower is better for all metrics.

	Ang Err	Pos Err	GD-r	GD-p	CD-s	CD-m	CD-w
ScrewSplat	27.4	0.13	54.7	0.01	58.2	684.3	66.2
Ours	3.67	0.03	7.05	0.00	2.53	2.70	2.64

multiple moving parts. As shown in Table 1, our method successfully converges on multi-part objects and substantially improves both geometric and kinematic accuracy over the baseline, demonstrating that our part-level parameterization scales to more complex articulated structures.

3.5 Animation of real-world reconstructions

We provide animations for the real-world results in Sec. 5.3 of the main paper. For each example, we interpolate the motion of the dynamic parts using the estimated articulation parameters, transform the corresponding 3DGS set to intermediate states, and render the resulting images. These animations demonstrate that our method can robustly recover both part-wise geometry and articulation parameters.

References

1. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: Density-based spatial clustering of applications with noise. In: Int. Conf. knowledge discovery and data mining. vol. 240 (1996)
2. 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. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 27144–27153 (2025)
3. 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)
4. Kim, S., Ha, J., Kim, Y.H., Lee, Y., Park, F.C.: Screwsplat: An end-to-end method for articulated object recognition. arXiv preprint arXiv:2508.02146 (2025)
5. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2017), `https://arxiv.org/abs/1412.6980`
6. Lin, S., Fang, J., Irshad, M.Z., Guizilini, V.C., Ambrus, R.A., Shakhnarovich, G., Walter, M.R.: Splart: Articulation estimation and part-level reconstruction with 3d gaussian splatting. arXiv preprint arXiv:2506.03594 (2025)
7. 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)

8. Liu, Y., Jia, B., Lu, R., Ni, J., Zhu, S.C., Huang, S.: Artgs: Building interactable replicas of complex articulated objects via gaussian splatting. arXiv preprint arXiv:2502.19459 (2025)
9. Mu, J., Qiu, W., Kortylewski, A., Yuille, A., Vasconcelos, N., Wang, X.: A-sdf: Learning disentangled signed distance functions for articulated shape representation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 13001–13011 (2021)
10. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
11. Song, C., Wei, J., Foo, C.S., Lin, G., Liu, F.: Reacto: Reconstructing articulated objects from a single video. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5384–5395 (2024)
12. Wang, J., Chen, M., Karaev, N., Vedaldi, A., Rupprecht, C., Novotny, D.: Vggt: Visual geometry grounded transformer. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 5294–5306 (2025)
13. Wang, Y., Zhou, J., Zhu, H., Chang, W., Zhou, Y., Li, Z., Chen, J., Pang, J., Shen, C., He, T.: π^3: Permutation-equivariant visual geometry learning (2026), https://arxiv.org/abs/2507.13347
14. Ye, V., Li, R., Kerr, J., Turkulainen, M., Yi, B., Pan, Z., Seiskari, O., Ye, J., Hu, J., Tancik, M., Kanazawa, A.: gsplat: An open-source library for gaussian splatting. Journal of Machine Learning Research **26**(34), 1–17 (2025)

Fig. 3: Further qualitative comparison on the PARIS dataset [7] under full-view and sparse-view settings. The left column shows the articulated objects in two states. The remaining columns visualize reconstruction and articulation estimation results of different methods (PARIS [7], ScrewSplat [4], ArticulatedGS [2], and ours). Dynamic parts are shown in blue and static parts in gray, while red arrows indicate the estimated articulation axes.

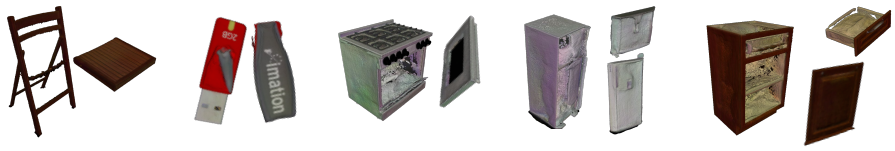

Fig. 4: Gallery of part-wise reconstructed meshes across different object categories.

Fig. 5: Animation results on real-world objects. For each example, the left two columns show the input image examples at the initial and articulated states, respectively. The remaining columns show rendered frames from the resulting animation, where the reconstructed part-level Gaussians are transformed according to the estimated articulation parameters.

Table 2: Quantitative per-scene results on the PARIS dataset under full-view and sparse-view settings with 100 and 10 input views, respectively. Geometry is measured by Chamfer Distance on the whole object (CD-w), static part (CD-s), and movable part (CD-m), with all values scaled by 10^3. Motion is evaluated by joint axis angular error, position error for revolute joints, and motion magnitude via geodesic and translation errors for revolute and prismatic motions, respectively. "F" denotes failure to predict a valid motion type, and "–" indicates non-applicable metrics. Per-scene motion metrics can be interpreted in the context of their scales: in most scenes under both full-view and sparse-view settings, angular errors are within 1°, position errors measured in meters are below 0.01 m, and geodesic errors are near 1°. These scene-wise variations are therefore numerically marginal. More importantly, our method achieves consistently low geometric error, whereas other methods may obtain competitive motion estimates while degrading geometric accuracy.

(a) Full-view setting (100 views)

			Synthetic										Real	
Group	Metric	Method	Foldchair	Fridge	Laptop	Oven	Scissor	Stapler	USB	Washer	Blade	Storage	Fridge	Storage
Motion	Ang Err	PARIS	0.837	0.128	0.867	0.204	2.451	1.059	0.179	2.323	48.118	0.179	2.595	19.53
		ScrewSplat	0.236	0.052	70.178	0.584	**0.020**	6.522	0.119	**0.088**	0.040	0.034	2.477	**4.244**
		ArticulatedGS	0.049	**0.028**	0.152	0.06	0.066	**0.074**	0.162	0.279	**0.020**	**0.021**	3.13	85.21
		Ours	**0.019**	0.055	**0.083**	**0.019**	0.092	0.128	**0.059**	0.096	0.292	0.034	**0.988**	13.01
	Pos Err	PARIS	0.218	0.005	0.009	0.001	0.361	1.020	0.832	0.636	-	-	0.016	-
		ScrewSplat	0.364	0.005	0.307	0.035	0.001	0.234	0.161	0.002	-	-	0.599	-
		ArticulatedGS	0.001	0.002	0.061	0.003	0.002	**0.001**	0.072	**0.000**	-	-	0.043	-
		Ours	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	0.003	**0.001**	**0.000**	-	-	**0.032**	-
	Geo Dist	PARIS	177.068	0.256	1.002	0.601	138.20	129.11	179.66	66.23	1.523	0.602	5.893	0.541
		ScrewSplat	14.602	0.555	47.378	8.930	79.942	20.378	85.726	0.069	0.400	**0.000**	91.809	0.499
		ArticulatedGS	0.063	0.383	4.970	0.163	**0.056**	**0.063**	0.173	0.220	0.400	**0.000**	6.195	F
		Ours	**0.027**	**0.068**	**0.055**	**0.027**	0.062	0.163	**0.068**	**0.055**	**0.001**	**0.000**	**0.017**	**0.024**
Geometry	CD-s	PARIS	11.676	3.064	**0.202**	9.485	2.636	**1.718**	2.189	16.291	1.806	6.206	41.897	67.321
		ScrewSplat	13.078	46.14	48.765	45.694	131.813	32.154	31.071	31.389	0.584	8.816	138.55	8.29
		ArticulatedGS	3.713	1.674	1.257	2.185	0.342	1.735	1.927	5.364	**0.304**	2.687	33.02	312.78
		Ours	**0.358**	**0.880**	0.374	**1.503**	**0.279**	5.360	**0.860**	**4.083**	0.311	**1.576**	**1.883**	**3.06**
	CD-m	PARIS	11.57	1.957	0.236	69.32	23.75	121.71	7.513	261.10	F	91.10	119.60	293.69
		ScrewSplat	34.295	1.433	17.723	126.36	0.257	61.70	2.160	43.82	2.636	31.617	12.39	50.76
		ArticulatedGS	**0.520**	0.699	6.05	0.818	0.407	1.283	1.061	**1.833**	1.526	1.664	71.61	1122.6
		Ours	1.083	**0.562**	**0.173**	**0.309**	**0.249**	**1.123**	**0.663**	7.639	**1.083**	**1.113**	**0.841**	**15.55**
	CD-w	PARIS	1.238	2.398	**0.228**	5.940	1.427	19.03	1.341	20.07	0.547	6.722	17.59	47.11
		ScrewSplat	1.986	4.124	6.004	5.444	45.28	25.66	15.45	22.204	0.425	12.78	20.04	8.77
		ArticulatedGS	0.512	1.395	5.010	2.024	0.331	1.508	1.265	4.918	0.231	1.890	23.09	231.41
		Ours	**0.227**	**0.847**	0.281	**1.368**	**0.248**	**0.881**	**0.745**	**2.985**	**0.175**	**1.441**	**1.242**	**3.226**

(b) Sparse-view setting (10 views)

			Synthetic										Real	
Group	Metric	Method	Foldchair	Fridge	Laptop	Oven	Scissor	Stapler	USB	Washer	Blade	Storage	Fridge	Storage
Motion	Ang Err	PARIS	22.03	27.92	27.95	75.54	4.15	30.15	1.08	18.19	66.97	54.39	34.06	46.33
		ScrewSplat	**0.138**	**0.034**	11.167	6.757	0.059	8.085	1.324	87.14	0.335	**0.32**	2.192	89.08
		ArticulatedGS	51.74	0.101	**0.138**	1.922	**0.04**	3.448	0.459	9.357	**0.034**	28.854	4.575	35.55
		Ours	0.935	1.061	0.280	**0.506**	0.117	**0.911**	**0.099**	**0.466**	0.852	1.786	**1.25**	**12.25**
	Pos Err	PARIS	0.137	0.286	0.131	0.050	0.180	0.403	0.004	0.110	-	-	0.211	-
		ScrewSplat	**0.001**	**0.001**	0.007	0.056	0.003	0.81	0.391	F	-	-	0.066	-
		ArticulatedGS	0.592	0.014	0.003	0.378	**0.000**	0.428	**0.000**	0.028	-	-	**0.002**	-
		Ours	**0.001**	0.024	**0.001**	**0.011**	**0.000**	**0.004**	0.001	**0.014**	-	-	0.034	-
	Geo Dist	PARIS	102.81	74.62	28.7	92.1	82.91	104.23	0.97	88.38	0.090	0.19	24.2	0.18
		ScrewSplat	**0.206**	119.93	80.711	5.081	0.069	52.30	21.43	F	0.401	F	1.632	F
		ArticulatedGS	79.66	4.271	0.384	11.96	**0.040**	69.50	0.388	17.03	0.400	0.141	4.968	0.642
		Ours	1.207	**4.121**	**0.267**	**3.531**	0.088	**0.626**	**0.088**	**1.046**	**0.080**	**0.004**	**0.927**	**0.050**
Geometry	CD-s	PARIS	309.16	180.69	92.14	34.21	907.8	3.16	2.46	16.34	1.12	27.43	63.28	55.08
		ScrewSplat	**1.72**	6.95	132.85	14.39	33.75	99.51	49.46	43.01	1.2	38.32	129.28	24.58
		ArticulatedGS	2.856	**0.993**	1.121	2.522	0.640	**1.491**	1.67	5.723	**0.215**	3.218	38.31	242.27
		Ours	3.483	1.256	**0.297**	**1.749**	**0.326**	3.708	**0.749**	**5.046**	0.348	**2.606**	**2.211**	**3.168**
	CD-m	PARIS	98.13	564.73	130.76	236.95	141.4	94.94	**2.47**	62.73	F	158.55	173.3	875.93
		ScrewSplat	88.27	117.64	3.51	436.79	37.14	70.42	123.1	123.1	105.52	229.52	34.39	36.55
		ArticulatedGS	104.30	1.434	0.693	22.69	0.568	445.56	3.810	**0.212**	3.423	44.303	66.40	931.62
		Ours	**5.244**	**0.890**	**0.151**	**0.508**	**0.288**	**1.071**	3.263	3.909	**0.967**	**3.522**	**1.086**	**22.127**
	CD-w	PARIS	175.73	198.72	82.82	32.47	312.9	13.45	2.11	18.37	0.56	9.2	32.24	65.58
		ScrewSplat	3.99	63.04	35.32	26.35	21.38	43.33	16.14	16.14	55.02	30.21	15.58	12.96
		ArticulatedGS	2.837	**1.024**	0.617	2.577	0.583	181.03	1.156	5.278	**0.195**	3.158	25.14	202.28
		Ours	**1.560**	1.166	**0.251**	**1.596**	**0.302**	**0.714**	**1.011**	**3.935**	0.204	**2.462**	**1.509**	**3.431**

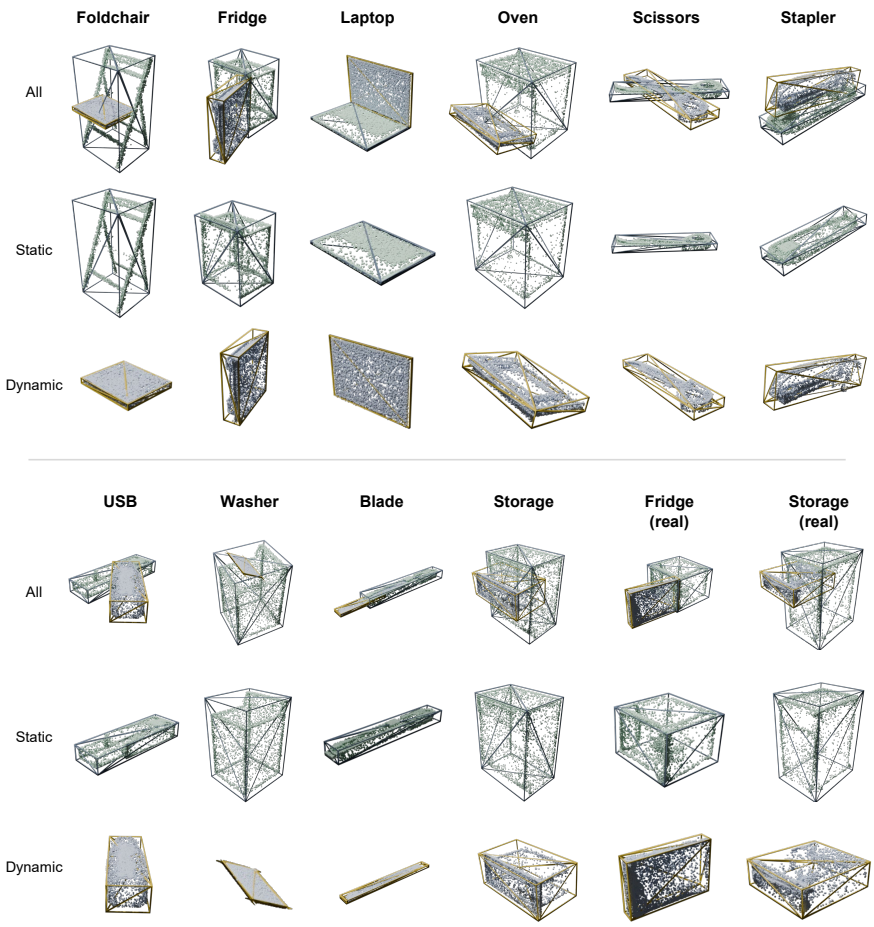

Fig. 6: OBB and 3DGS fitting results on the PARIS dataset. Columns correspond to different scenes from the PARIS benchmark. Rows show the full reconstruction ("All") and the static and dynamic parts. For each case, we visualize the fitted OBBs together with the reconstructed 3D Gaussians.