Cross-Instance Gaussian Splatting Registration via Geometry-Aware Feature-Guided Alignment

---

Supplemental Material

Appendix Contents:

A. Additional Visual Results

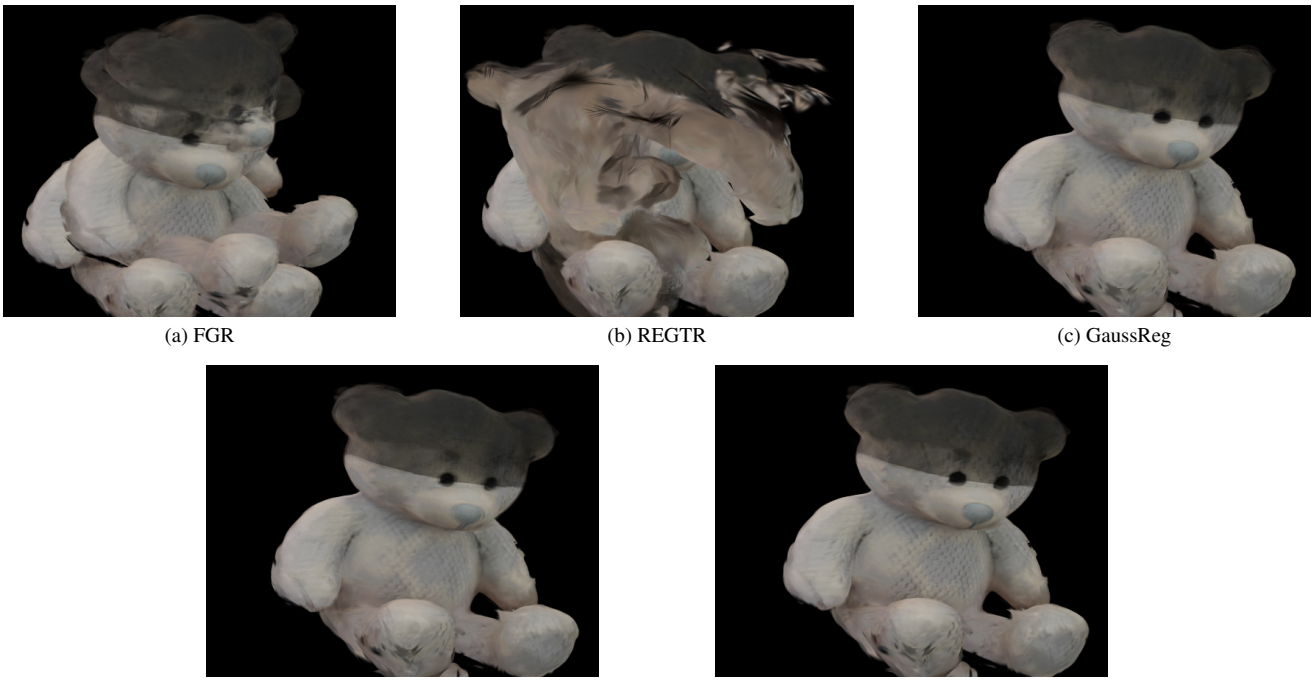
Figure A.1 depicts a case where the task is to align two models of the same object, as described in the first experiment in the paper. In this particular case, it is the same chair. As Figure A.1 shows, the competing methods (FGR, REGTR, and GaussReg) struggle with the geometric structure of the chair. In contrast, our GSA, even when using only its coarse step, achieves almost-perfect results. Adding the fine step here did improve the result a bit further, but as the coarse step was so successful, the visual difference is too subtle to note.

Figure A.2 depicts another same-object example, this time of a teddy bear. Again, GSA successfully aligned the models. The cloud-based methods (FGR, REGR) struggle here too. Here, GaussReg did better than it did with the chair. That said, quantitatively, GSA's result was still better, even if it is hard to tell this fact by visual inspection.

Figure A.3 provides cross-object examples. The other methods, which are not designed to cope with such a case, completely failed, so we omit their results. As the figure shows, our method successfully aligns (within each pair) the two different boats, the two different airplanes, and the two different chair,s even in hard settings, such as when the initialization is wrong by 180 degrees and/or when there is a significant scale difference.

(a) FGR (b) REGTR (c) GaussReg (d) **GSA: coarse (ours)** (e) **GSA: fine (ours)**

Figure A.1. A Comparison of alignment results on two 3DGS models of the same chair from the Objaverse dataset. Although the object has a clear and rigid structure, classical and learning-based methods struggle to achieve consistent alignment. Our GSA method, especially with coarse and fine refinement, achieves the most accurate and coherent results.



(a) FGR (b) REGTR (c) GaussReg

(d) **GSA: coarse (ours)** (e) **GSA: fine (ours)**

Figure A.2. Comparison of two 3DGS models of the same teddy bear, Objaverse [1] dataset. Although the object is somewhat simple and although there is a unique rigid transformation solution, existing point cloud-based registration methods still struggle while our coarse point-cloud registration achieves almost-perfect results (so in this particular case, the further improvement by our fine step is hard to notice visually).

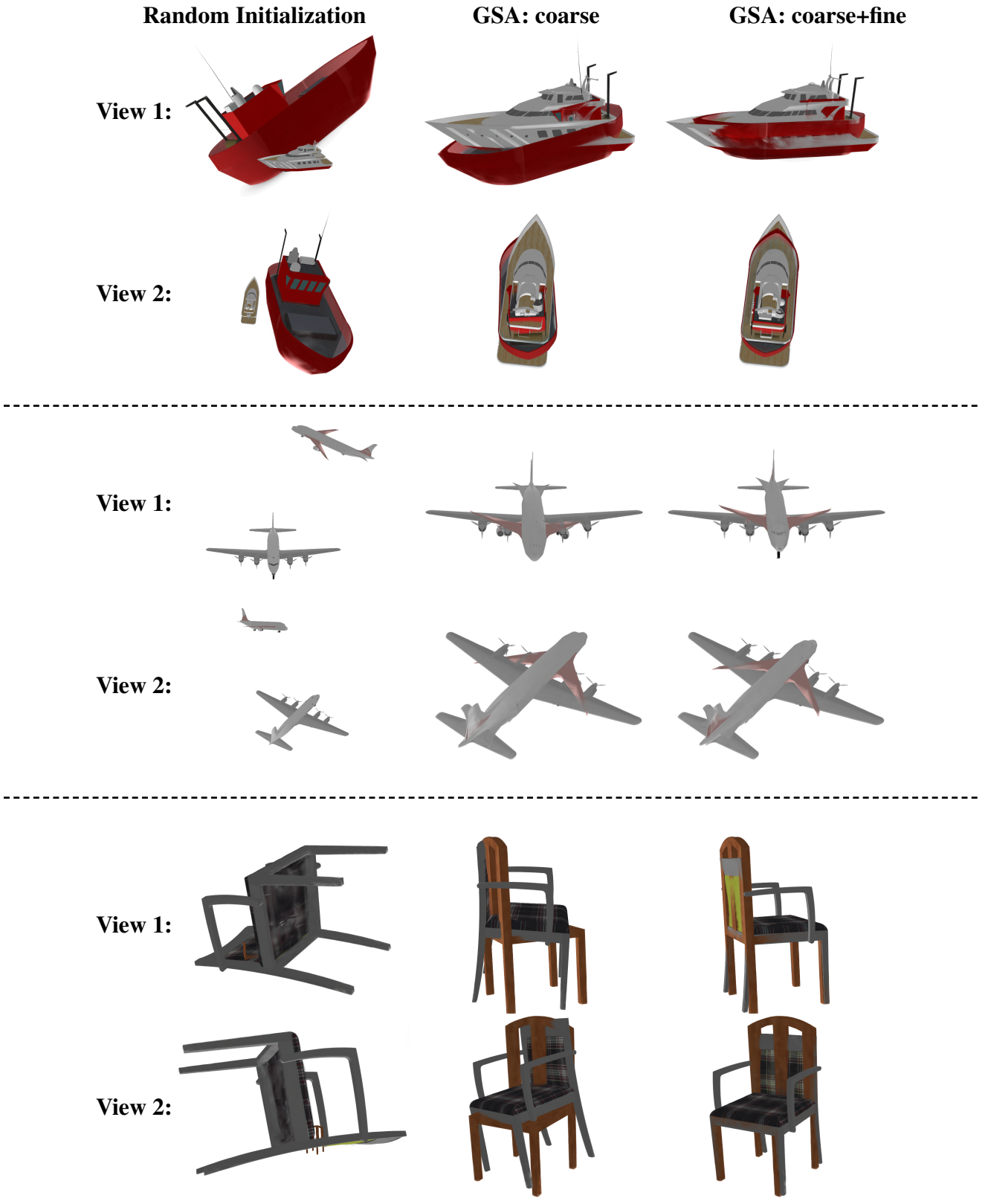


Figure A.3. Visual qualitative results for aligning different objects within the same cateogry. In each of the three 2-row blocks we show an example form a different category (boats, airplanes, and chairs). The two rows within each 2-row block show different views. In each 2-row block we show, from lest to right: 1) a random initialization; 2) the result of GSA's coarse step; 3) the result of GSA's fine step. The random initialization includes arbitrary translations, rotations up to 180 degrees, and significant scale differences.

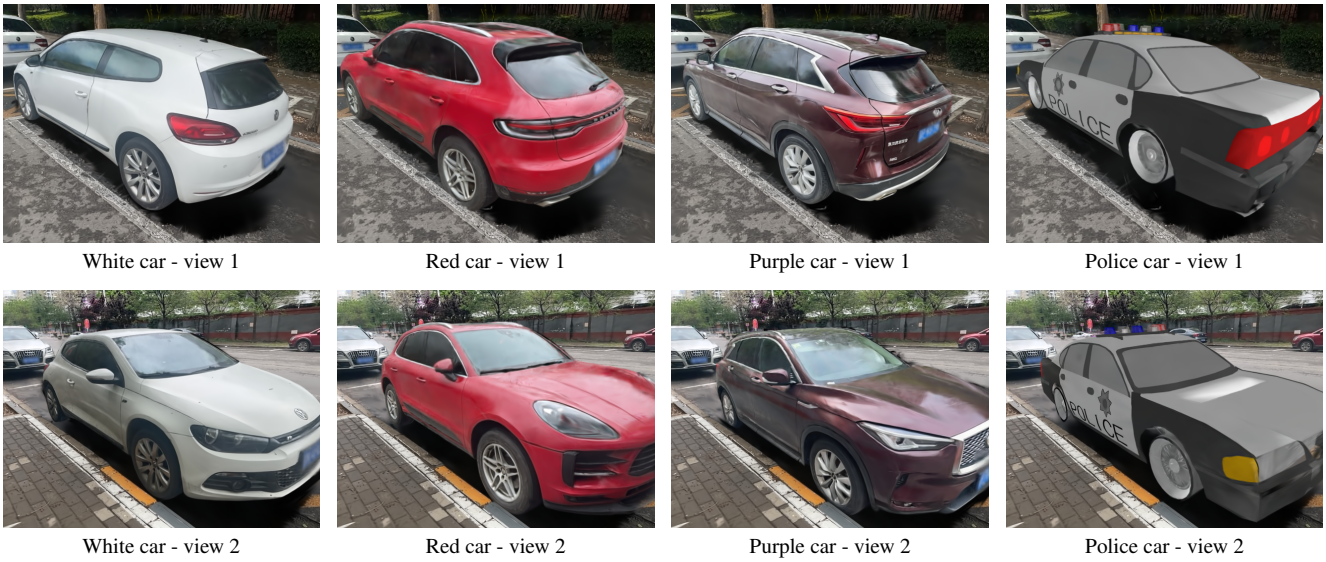
B. Applications

We demonstrate the practical utility of our method through two downstream applications: *geometrically consistent object replacement* and *synchronized novel view synthesis*. These examples showcase how our method enables accurate, scale-aware alignment for real-world 3D vision tasks.

B.1. Geometrically Consistent Object Replacement

In this application, depicted in Figure B.1, we show the replacement of a reference object in a scene with geometrically-aligned models from the same category. Specifically, we aligned three car models (red and purple cars from 3D Real Car Dataset [2], and a synthetic police car from ShapeNet [3]) to a reference white car from [2] using GSA. After alignment, the white car was removed from the scene and substituted with the aligned models.

Our approach ensures alignment in rotation, translation, and *symmetric scale*, where the scaling factor is applied uniformly across spatial dimensions. Symmetric scaling maintains the intrinsic geometry of the source model, preserving its characteristic shape and aspect ratio while adapting its overall size to best match the target object. For example, a long and low-profile car will remain elongated after scaling, and a taller car will retain its vertical prominence. This yields visually-coherent substitutions that are structurally faithful.



White car - view 1 | Red car - view 1 | Purple car - view 1 | Police car - view 1

White car - view 2 | Red car - view 2 | Purple car - view 2 | Police car - view 2

Figure B.1. Geometrically consistent object replacement using GSA. Each row shows a different camera view. The first column displays the original white car. The second to fourth columns show replacements with red, purple, and toy police cars, respectively. GSA aligns the models in rotation, translation, and symmetric scale, preserving their intrinsic geometry while adapting them to best match the target object's position, orientation, and scale in a structurally faithful way.

B.2. Synchronized Novel View Synthesis

We demonstrate synchronized novel view synthesis across semantically similar objects by aligning all models to a reference using GSA (e.g., aligning all the objects to object 1). This alignment enables consistent view generation from shared camera poses using 3DGS.

By resolving differences in orientation and scale, our method produces spatially coherent renderings beneficial for applications such as data visualization, robotic perception, and more.

Figure B.2 shows synchronized views across instances within the same category. Figure B.3 visualizes the jointly-aligned models from two additional novel views.

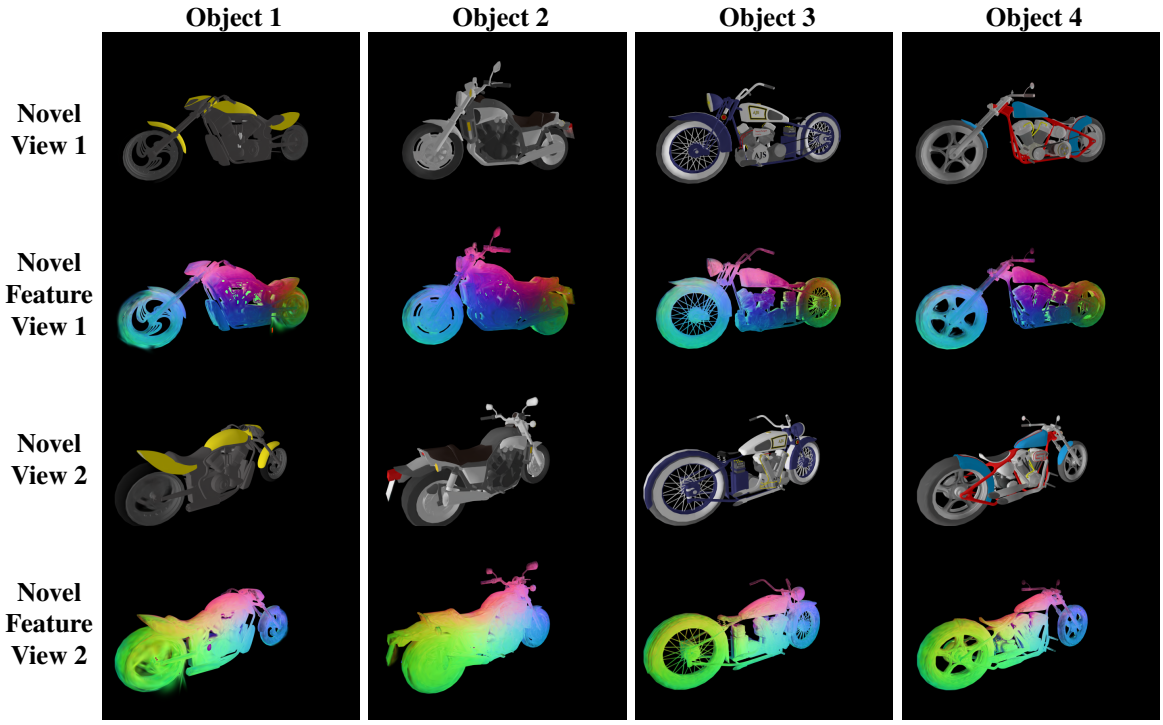


Figure B.2. Synchronized novel view and feature renderings across multiple semantically aligned motorcycle instances. Each row shows either a novel rendering or the corresponding feature projection for different objects and camera views.

Figure B.3. Renderings of the unified 3D Gaussian Splatting (3DGS) model combining all gaussians from the aligned motorcycle instances, shown from two additional novel synchronized views. These renderings provide a global perspective of the unified category-level geometry.

C. Scale Estimation

In Appendix A, we presented qualitative scale estimation results across objects within the same category. In such cases, scale and translation are inherently ambiguous due to intra-category variation in size and geometry, leading to multiple plausible alignment solutions and a one-to-many mapping in the transformation space.

To quantitatively assess our method under well-defined conditions, we trained 30 3D Gaussian Splatting (3DGS) [4] models on distinct ShapeNet [3] objects. Each object was duplicated and randomly transformed using translation, rotation (up to 180° per axis), and scaling (up to ×10). We then registered each transformed model to its original to evaluate scale estimation accuracy.

Scale Error Metric. We report the *relative scale error*, computed as the absolute difference between the estimated scale s_{est} and the ground-truth scale s_{gt}, normalized by the ground-truth value and expressed as a percentage:

$$\text{Scale Error} = \frac{|s_{\text{est}} - s_{\text{gt}}|}{s_{\text{gt}}} \times 100\%.$$

This metric penalizes both under- and over-estimation symmetrically and provides an interpretable measure of scale estimation quality.

Table C.1. Quantitative evaluation of rotation and scale estimation on transformed ShapeNet objects.

Method	**Mean RRE (deg)**	**Mean Scale Error (%)**
GSA: Coarse	0.362	0.21
GSA: Coarse+Fine	0.074	0.08

As shown in Table C.1, our method achieves highly accurate rotation and scale estimation. The coarse stage yields a scale error of 0.21% and a rotation error of 0.362°. With fine alignment, these improve to 0.08% and 0.074°, respectively.

To qualitatively support these results, we show in Figure C.1 an example with a large initial scale discrepancy and nearly 180° misalignment along two axes. While the coarse alignment already yields high accuracy, close-up views of the boat top reveal subtle yet consistent improvements achieved by the fine stage, further validating the quantitative gains in both scale and rotation.

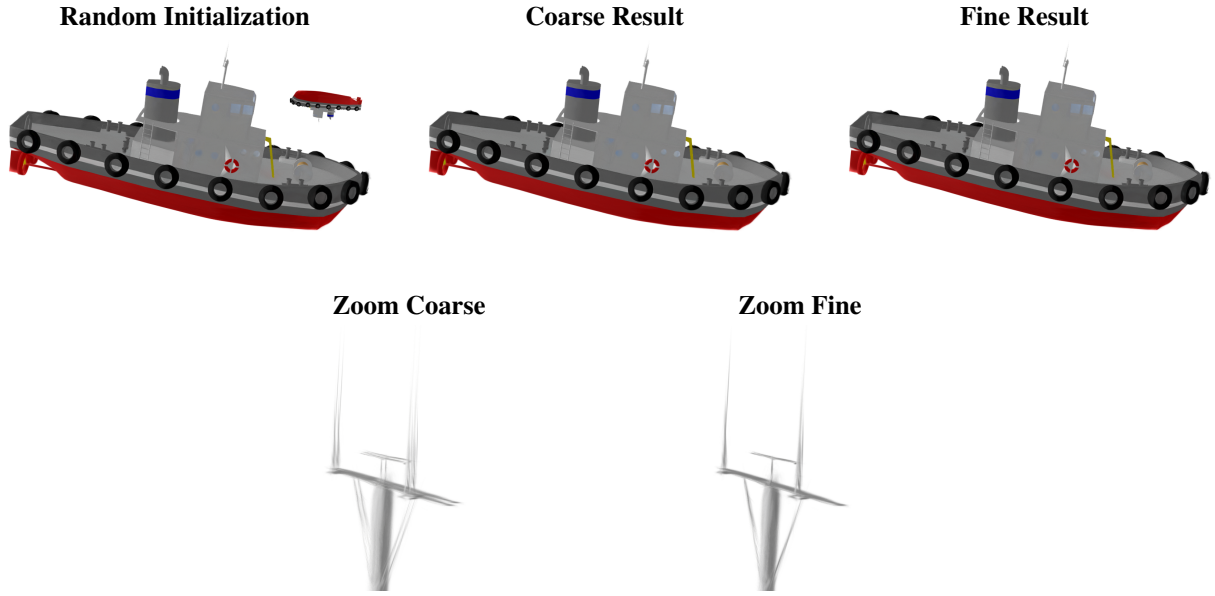


Figure C.1. Qualitative registration results. Top row: comparison of the random initialization, coarse, and fine alignment results. Bottom row: close-up views of the boat top, highlighting the improved accuracy of the fine stage over the coarse alignment.

D. The Impact the Choice of Features has on the Alignment

We now further explain our choice of using the features from [5] and not, say, other plausible choices such as DINOv2 [6] or TellingLeftfromRight [7].

We consider three feature types: DINOv2 [6], which is not geometry-aware, and the geometric-aware TellingLeftfromRight [7] and view-guided spherical map features [5]. Figure D.1 shows that DINOv2 struggles with distinguishing between semantically similar but spatially distinct parts (*e.g.*, all four wheels and the hood/trunk). This is issue for alignment as it leads to ambiguity and up to *180-degree misalignment*. Similarly, TellingLeftfromRight, while effective for some categories, for others degenerates to image-level left–right prediction instead of object-centric geometric reasoning, and therefore does not resolve *cross-image symmetry ambiguities*, which can lead to 180-degree errors. In other words, while DINOv2 and TellingLeftfromRight operate in higher-dimensional spaces (*e.g.*, 384 and 768 for ViTs and ViT-Base, resp.) and encode rich semantics, their utility for alignment is limited. In contrast, *view-guided spherical map features* provide effective geometry-aware representations, leading to significantly improved alignment. Moreover, these features are much more compact (3D feature vectors), thereby ensuring not only geometric awareness but also efficiency.

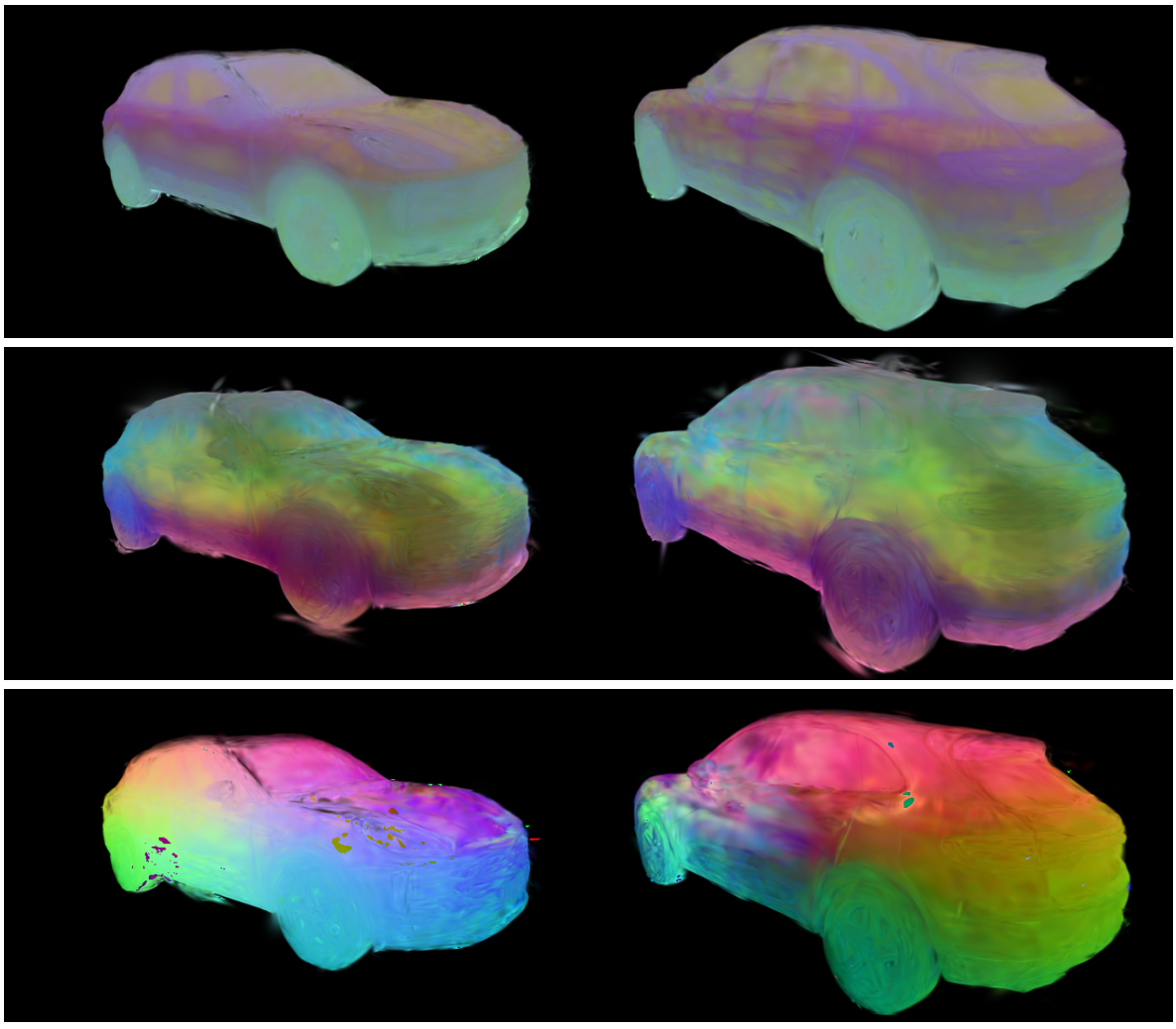
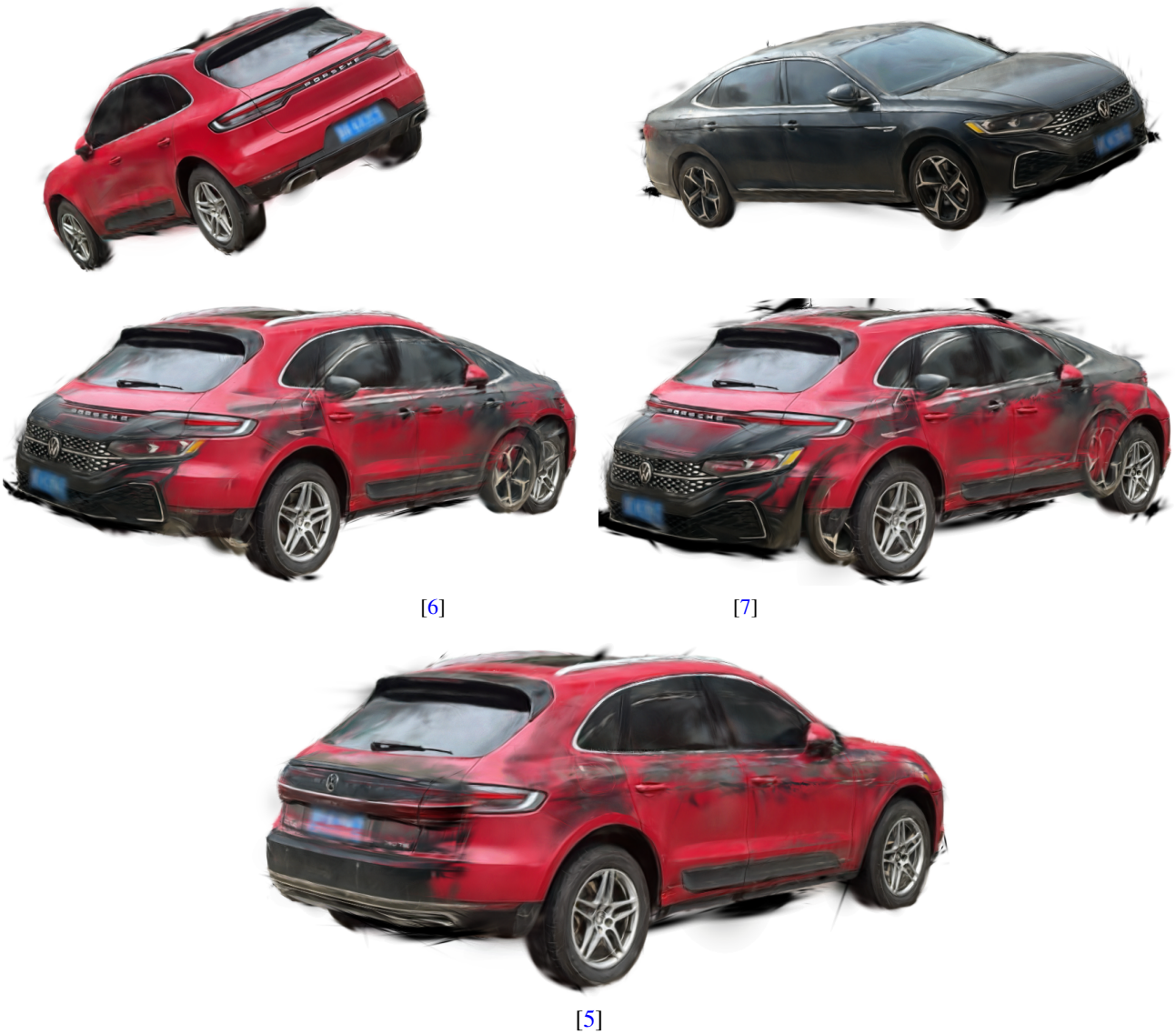


Figure D.1. Augmented 3DGS models using different choices of semantic features: [5], [6], and [7]. For [6] or [7], dimensionality was reduced to 3 by Principal Component Analysis.

A typical comparison between these three feature types, when used within the proposed GSA method, is shown in Figure D.2. As explained in the figure's caption, in this example, using the features from either [6] or [7] results in gross errors, with angular errors close to 180 degrees. This is because these features do not disambiguate or do not disambiguate enough to distinguish the car directions (e.g., left, right, front, back). When we leverage the features from [5], however, the problem disappears.



[6] [7]

[5]

Figure D.2. Qualitative comparison of the GSA coarse alignment step, when using different features. Top row: Unaligned models. Middle row, left: Alignment result when using DINOv2 [6]. Note the angular mistake is about 180 degrees. Middle row, right: Alignment results when using and [7]. Again the mistake is about 180 degrees, and the tilting angle is also evidently wrong. Bottom row: In contrast, when GSA uses [5], it successfully aligns the cars.

E. The Common problem in 3DGS Models of Background-related Gaussians and our Effective solution to it

Figure E.1 shows that this phenomenon can hurt results if left unaddressed. For example the black Gaussians can block novel views of the object. While this issue affects 3DGS models in general, it is particularly problematic in our case because such occlusions directly degrade the multi-view consistency loss during fine alignment.

As a solution, which is in fact useful not just for our case but in general when working with 3DGS models, we apply a simple effective strategy. When building the 3DGS model, in each densification step (a standard step in 3DGS construction), we remove the Gaussians that do not encode meaningful features. Specifically, if a Gaussian has a near-zero feature vector (as measured by its magnitude), we remove it. This approach eliminates unwanted artifacts, improves robustness, and enhances the overall 3D reconstruction quality.

(a) 3DGS with spurious background Gaussians

(b) 3DGS when our solution for eliminating background Gaussian is used.

Figure E.1. The effect of background pruning. (a) Extraneous background Gaussians create occlusions, and are inconsistent across the two models, complicating alignment. (b) Our solution removes these artifacts, improving alignment consistency.

F. Computational Complexity and Runtime Analysis

Our current implementation of coarse alignment is fully GPU-accelerated, achieving runtimes of less than one second. The subsequent fine registration step adds approximately 6–8 seconds. During the coarse stage, we compute a subset of potential correspondences for each point in model 1 based on semantic feature similarity, with a complexity of $O(NM)$, where N and M are the number of Gaussians in the first and second models, respectively. We then perform iterative nearest neighbor searches within these subsets, constrained to points with at least one valid match. This search runs in $O(SK_{\max})$, where S is the number of such points and $K_{\max}$ is the maximum size of any correspondence subset.

While this GPU implementation provides excellent performance, we plan to contribute an optimized multithreaded CPU version of the coarse alignment algorithm to the Open3D [8] framework. This will broaden accessibility and ensure high-performance semantic feature-based registration even in CPU-only environments.

G. Hybrid Kabsch-Umeyama and Horn Closed-Form Solution to the Absolute Orientation Problem

The following describes a closed-form solution combining the Kabsch-Umeyama method [9] for optimal rotation and translation with Horn's method for estimating a symmetric scale [10, 11]. Let $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N$ and $\boldsymbol{Y} = \{\boldsymbol{y}_i\}_{i=1}^N$ be two sets of corresponding points in $\mathbb{R}^3$. The goal is to find a similarity transformation (rotation $\boldsymbol{R}$, scale s, and translation $\boldsymbol{t}$) aligning $\boldsymbol{X}$ to $\boldsymbol{Y}$.

G.1. Problem Formulation

The optimal similarity transformation minimizes:

$$E(s, \boldsymbol{R}, \boldsymbol{t}) = \sum_{i=1}^{N} \|s\boldsymbol{R}\boldsymbol{x}_i + \boldsymbol{t} - \boldsymbol{y}_i\|^2. \tag{1}$$

G.2. Closed-Form Solution

Step 1: Compute Centroids Compute centroids of the point sets:

$$\bar{\boldsymbol{x}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i, \tag{2}$$

$$\bar{\boldsymbol{y}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{y}_i. \tag{3}$$

Step 2: Compute Covariance Matrix Define the covariance matrix:

$$\boldsymbol{H} = \sum_{i=1}^{N} (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{y}_i - \bar{\boldsymbol{y}})^T. \tag{4}$$

Step 3: Compute Optimal Rotation $\boldsymbol{R}$ via Kabsch-Umeyama Perform Singular Value Decomposition (SVD) on $\boldsymbol{H}$:

$$\boldsymbol{H} = \boldsymbol{U}\mathbf{S}\boldsymbol{V}^T. \tag{5}$$

The optimal rotation from Umeyama [9] is:

$$\boldsymbol{R} = \boldsymbol{V}\,\mathrm{diag}(1, 1, \det(\boldsymbol{V}\boldsymbol{U}^T))\,\boldsymbol{U}^T, \tag{6}$$

ensuring $\boldsymbol{R}$ is a proper rotation ($\det(\boldsymbol{R}) = +1$).

Step 4: Compute Symmetric Scale s via Horn's Method Horn's symmetric scale [11] is given by:

$$s = \frac{\sum_{i=1}^{N} \|\boldsymbol{y}_i - \bar{\boldsymbol{y}}\|^2}{\sum_{i=1}^{N} \|\boldsymbol{x}_i - \bar{\boldsymbol{x}}\|^2}. \tag{7}$$

This symmetric scale factor ensures good performance in forward and inverse alignment tasks.

Step 5: Compute Translation $\boldsymbol{t}$ Finally, compute the translation vector:

$$\boldsymbol{t} = \bar{\boldsymbol{y}} - s\boldsymbol{R}\bar{\boldsymbol{x}}. \tag{8}$$

This hybrid approach, combining the rotation and translation from the Kabsch-Umeyama method [9] with Horn's symmetric scale estimation [11], provides improved accuracy and robustness, using the feature-guided correspondences.

References

[1] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13142–13153, June 2023. 3

[2] Xiaobiao Du, Haiyang Sun, Shuyun Wang, Zhuojie Wu, Hongwei Sheng, Jiaying Ying, Ming Lu, Tianqing Zhu, Kun Zhan, and Xin Yu. 3drealcar: An in-the-wild rgb-d car dataset with 360-degree views. *arXiv preprint arXiv:2406.04875*, 2024. 5

[3] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 5, 7

[4] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 7

[5] Octave Mariotti, Oisin Mac Aodha, and Hakan Bilen. Improving semantic correspondence with viewpoint-guided spherical maps. In *CVPR*, 2024. 8, 9

[6] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. In *arXiv preprint*, 2023. 8, 9

[7] Junyi Zhang, Charles Herrmann, Junhwa Hur, Eric Chen, Varun Jampani, Deqing Sun, and Ming-Hsuan Yang. Telling left from right: Identifying geometry-aware semantic correspondence. In *CVPR*, 2024. 8, 9

[8] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018. 11

[9] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991. 12

[10] Berthold KP Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the optical society of America A*, 4(4):629–642, 1987. 12

[11] Berthold KP Horn, Hugh M Hilden, and Shahriar Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Society of America A*, 5(7):1127–1135, 1988. 12