Revisting EgoLifter: Additional Experiments with 3DGS Segmentation

Jack Daus and Ankit Raut
George Mason University
jdaus2@gmu.edu, araut6@gmu.edu

ChatGPT
Model o3
chatgpt.com

Abstract

Recent advances in 3-D Gaussian Splatting (3DGS) enable rapid, photorealistic scene capture, yet instance segmentation for 3DGS remains noisy. EgoLifter lifts 2-D SAM masks into 3-D by learning a 16-D per-Gaussian embedding with a supervised contrastive loss, but its clusters often bleed across object boundaries. We investigate whether injecting native 3DGS attributes—position, color, scale, and opacity—through a lightweight multilayer perceptron (MLP) can provide stronger cues for the contrastive objective. Experiments on the Aria Digital Twin sequence 131 explore MLP depth (2–4 layers), embedding size (3–16 D), and loss variants (Euclidean vs. NT-Xent). After 10k training steps our best model reaches $\approx$ 25 dB PSNR, on par with the EgoLifter baseline, but visual inspection of PCA projections of the learned embedding space shows no clearer separation of object clusters. These findings suggest that directly feeding geometric attributes does not provide additional benefit for this lifting task.

1. Introduction

Three-dimensional Gaussian Splatting (3DGS) has recently emerged as an efficient, photorealistic alternative to mesh- or NeRF-based scene capture [2]. A single 3DGS model represents an environment as millions of oriented 3-D Gaussians whose positions, colors, scales, and opacities are optimized from multi-view images. Once reconstructed, these point-like primitives enable fast rasterization and compact storage, making 3DGS attractive for digital-twin applications in architecture, engineering, and construction (AEC).

For AEC workflows, engineers often need to manipulate individual objects—e.g., remove scaffolding, reposition equipment, or measure clearance—but a raw 3DGS is delivered as one monolithic cloud. Today an operator must draw 3-D bounding boxes or lasso individual Gaussians by hand—an error-prone and time-consuming task. Automatic instance segmentation would unlock nondestructive editing as well as downstream analytics such as inventory counting and progress monitoring.

Prior work attempts to "lift" 2-D segmentation masks into 3-D by learning an additional embedding per Gaussian. A recent example, EgoLifter (Gu et al., 2024), projects Segment-Anything (SAM) masks from egocentric videos onto the splat cloud and trains a supervised contrastive loss in a 16-D feature space [1]. Although promising, EgoLifter frequently produces clusters that leak across object boundaries, limiting its practical use.

In this project we revisit the EgoLifter pipeline and ask a simple question: *Can the native geometric and photometric attributes already present in 3DGS provide stronger guidance than purely learned features?* We experiment with feeding these attributes through a lightweight MLP before applying the same contrastive objective. The remainder of the report describes the dataset (Section 2.3), our analysis of the embedded feature distribution of the baseline method (Section 2.6), our modifications (Section 2.7), followed by qualitative results and discussion (Section 3).

2. Approach

2.1. Code

This project is fork of Meta's EgoLifter project. The repository for this project can be found on GitHub at https://github.com/jackdaus/egolifter.

2.2. Background

Our approach builds on recent work by Gu et al. (2024) that segments 3DGS reconstructed from egocentric video. EgoLifter uses contrastive learning to lift 2-D segmentation masks into a 3DGS model; the 2-D masks are generated with the off-the-shelf Segment Anything Model (SAM).

2.3. Dataset

The data come from the Aria Digital Twin (ADT) dataset. Project Aria is an effort by Meta to accelerate machine-learning research by providing a platform that collects multi-modal egocentric data from the wearable *Project Aria Glasses*. ADT contains RGB videos, IMU

Figure 1. Example of automatic segmentation masks generated by the Segment Anything Model (SAM).

streams, eye-tracking, 3-D object poses, 2-D instance segmentation, and photo-realistic mesh models, totaling more than 400 minutes of egocentric capture.

For simplicity we focus on a single video sequence: `Apartment_release_work_skeleton_seq131_M1292` (126 seconds). See the public data explorer for details on this dataset. [1]

2.4. Preprocessing

The sequence is preprocessed with EgoLifter's ADT data processing scripts. Processing extracts RGB frames and applies SAM to produce dense per-frame masks. Masks are generated independently per image; object segmentations are not linked across frames.

2.5. EgoLifter baseline

The original EgoLifter method lifts 2-D segmentation masks into a 3DGS scene through a supervised contrastive loss. Each Gaussian is augmented with a 16-D learnable embedding. During training these embeddings are rasterized by the differentiable 3DGS renderer and alpha-composited to form a per-pixel "segmentation feature" image. For every rendered frame, pixels are sampled and compared: if both pixels fall inside the same SAM mask they constitute a positive pair; if they lie in different masks they constitute a negative pair. The contrastive objective back-propagates to the 16-D embeddings, pulling Gaussians that belong to the same object together and pushing unrelated ones apart. Figure 2 illustrates the pipeline.

EgoLifter also trains a transient-object classifier that identifies dynamic elements before reconstruction. We disable this module and adopt the predefined "vanilla" settings so that our experiments isolate the segmentation component.

2.6. Analysis of the baseline embedding

We projected the 16-D embeddings with PCA and t-SNE. PCA revealed a near-normal distribution, while t-SNE showed a single dense blob with only several well separated islands, indicating room for improved cluster purity.

2.7. Proposed method

Our objective is to obtain cleaner instance separation in the learned embedding, as revealed by t-SNE and PCA projections. We hypothesize that the native 3DGS attributes already encode strong locality priors: Gaussians that belong to the same rigid object are spatially adjacent (*position*), similarly sized (*scale*), and often share optical density (*opacity*) and color. Instead of learning all 16 dimensions from scratch, we feed a subset of these "core" attributes directly into a network to learn the mapping to this embedding space.

Concretely, for each Gaussian we concatenate $\mathbf{g} = [x, y, z, s_x, s_y, s_z, \alpha]$. This vector $\mathbf{g} \in \mathbb{R}^{d_{\text{in}}}$ is passed through a lightweight multilayer perceptron $\phi_\theta : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{emb}}}$ with 2–4 layers and ReLU activations. The resulting embedding $\mathbf{e} = \phi_\theta(\mathbf{g})$ replaces EgoLifter's learnable 16-D extra-features. The remainder of the pipeline—differentiable rasterization and mask-based pixel sampling—is left unchanged.

Our experiments included exploring adjusting the following hyperparameters:

- Attribute subsets: position, scale, opacity vs. position, scale vs. position only
- Embedding dimension $d_{\text{emb}} \in 3, 16$
- MLP depth $\in 2, 3, 4$ (hidden width 128)
- Contrastive loss variant: Euclidean vs. NT-Xent

3. Results

Our best configuration achieves $\approx$ 25 dB PSNR at 10k steps, approximately matching the EgoLifter baseline. Additional quantitative segmentation metrics were not computed in this project due to time constraints; evaluation is therefore qualitative.

[1] https://explorer.projectaria.com/adt/Apartment_release_work_skeleton_seq131_M1292

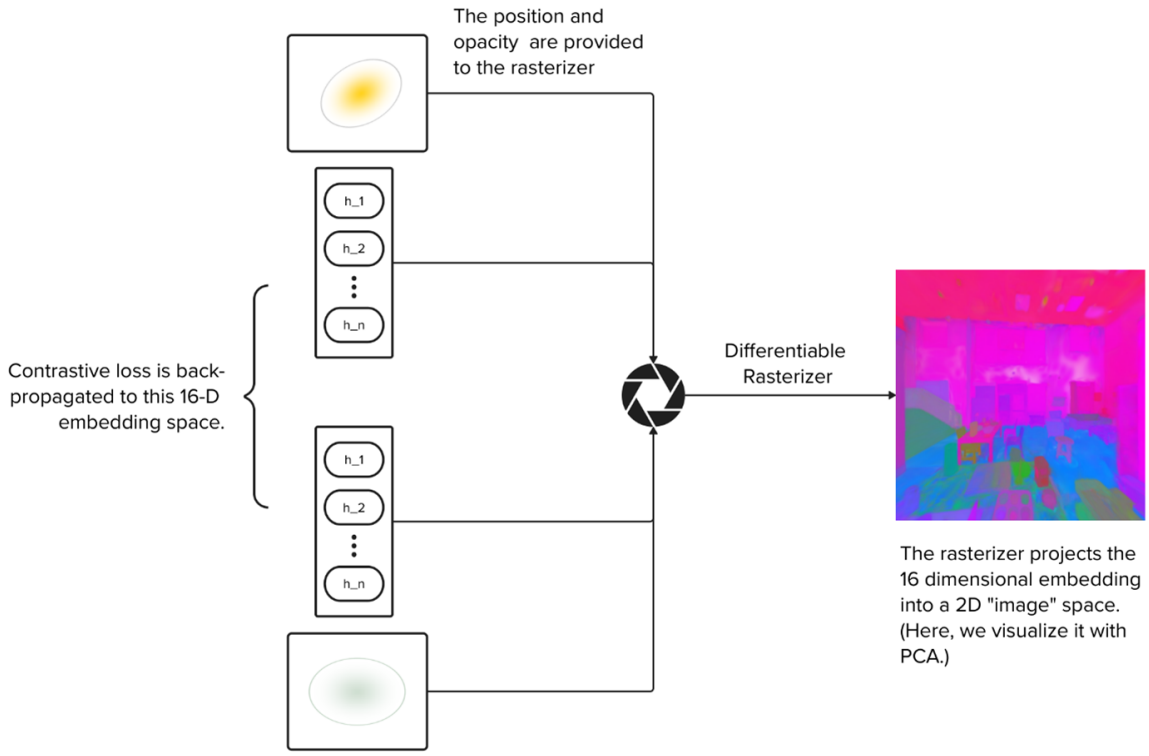


Figure 2. An overview of the contrastive loss from the original EgoLifter. The embedding space is projected through the rasterizer. Contrastive loss back-propagates to this 16-D embedding space with the goal of separating distinct objects in the scene.

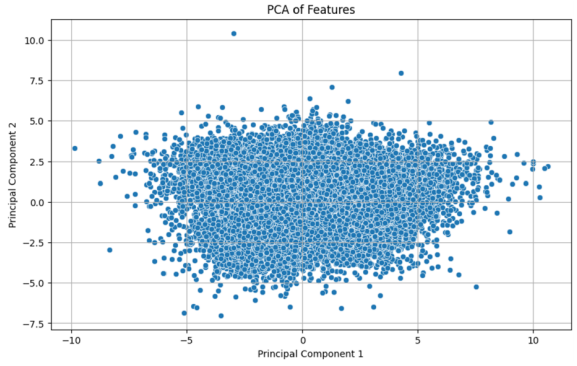


Figure 3. A PCA projection of the 16-D embedding space of the learned features from the original EgoLifter setup (vanilla model).

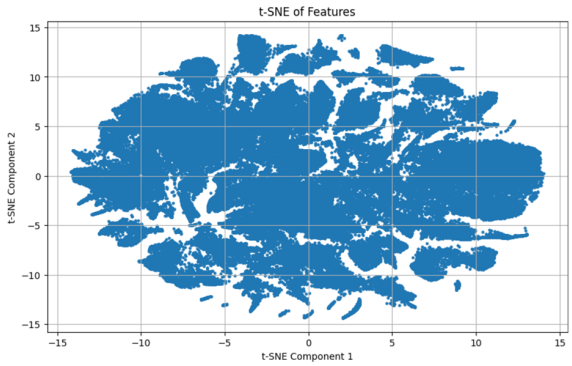


Figure 4. A plot of a t-SNE dimensionality reduction of the learned 16-D embedding from the original EgoLifter pipeline (vanilla model). We see some disconnected islands, but there are many large overlapping areas. This suggests 2D segmentations may not be lifted effectively.

3.1. Qualitative inspection

- **Baseline**: clusters bleed across object edges. The PCA projected rasterization of the 16-D embedding space looks somewhat uniform in color for nearby objects.
- **Ours**: the PCA projected rasterization still looks muddy. Results look worse than baseline. See Figure 8.

4. Related Work

This project extends EgoLifter and reuses its preprocessing pipeline. NT-Xent implementation followed guidelines by Matani [3]. Additional tooling: Weights&Biases for experiment tracking, `uv` for dependency management, and

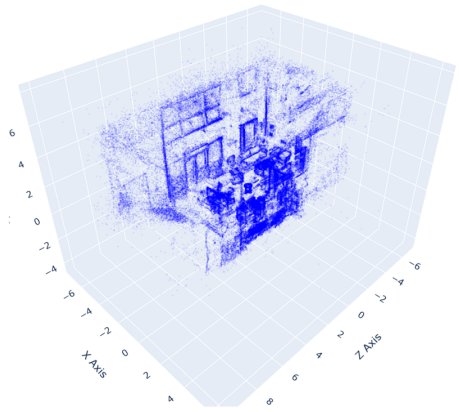


Figure 5. A plot of the position features from the learned 3DGS of ADT Sequence 131. We hypothesize that position features alone already provide a good starting place to learn the clustering of distinct objects.

Rerun for spatial data visualization.

5. Resources

- **EgoLifter repository**: https://github.com/facebookresearch/egolifter
- **SAM and SAM2**: image/video segmentation tools — https://github.com/facebookresearch/sam2
- **ChatGPT o3**: assistance with data-visualization code, debugging, dependency resolution, implementing NT-Xent loss, and writing this report.
- **Weights&Biases**: experiment logging and graphing.
- **uv**: fast Python dependency management.
- **Rerun**: interactive visualization of spatial AI data.
- **Vizer**: interactive 3DGS viewer.

6. What I learned

By starting with an existing open-source project, I learned a great deal from the EgoLifter authors about structuring large-scale AI experiments. I adopted supporting libraries such as PyTorch Lightning to modularize training code and Hydra to manage configuration files. Weights&Biases enabled real-time logging and monitoring of training runs, while Rerun made it easy to visualize and explore spatial data.

Throughout the project I gained experience resolving dependency issues with the uv package manager and discovered the practical limitations of Google Colab. I also learned how to provision on-demand cloud GPU instances using Lambda Labs and RunPod. Finally, hands-on use of SAM and SAM2 for object and video segmentation broadened my proficiency using modern annotation pipelines.

7. Discussion and Conclusion

Although our experimental variants match the EgoLifter baseline in photometric quality ($\approx$ 25 dB PSNR at 10k steps), they do not produce visibly cleaner segmentation clusters. This negative finding highlights two methodological challenges that merit further study.

Gradient leakage into geometry. The supervised contrastive loss back-propagates through the differentiable rasterizer, thereby altering the XYZ positions of Gaussians as well as their embeddings. Freezing, or stopping the gradient for, the positional parameters when computing the segmentation loss could let the optimizer focus on feature separation without distorting scene geometry.

Frame-wise masks. Our experiment still relies on single-frame SAM masks that ignore temporal consistency. Video object segmentation (VOS) tools such as SAM2 can provide cross-frame masks, offering richer positive and negative pairs for contrastive learning. Incorporating such masks—and possibly an additional temporal loss—may help tighten clusters and reduce bleeding artifacts. Although our initial project plan included the use of SAM2 for VOS, compute limitations prevented us from processing our dataset as intended.

Conclusion Despite not achieving state-of-the-art segmentation, the study clarifies which design choices are ineffective and outlines concrete directions for future work.

References

[1] Q. Gu, Z. Lv, D. Frost, S. Green, J. Straub, and C. Sweeney. Egolifter: Open-world 3d segmentation for egocentric perception. *arXiv preprint arXiv:2403.18118*, 2024. 1

[2] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. 1

[3] D. Matani. Nt-xent (normalized temperature-scaled cross-entropy) loss explained and implemented in pytorch, 2023. 3

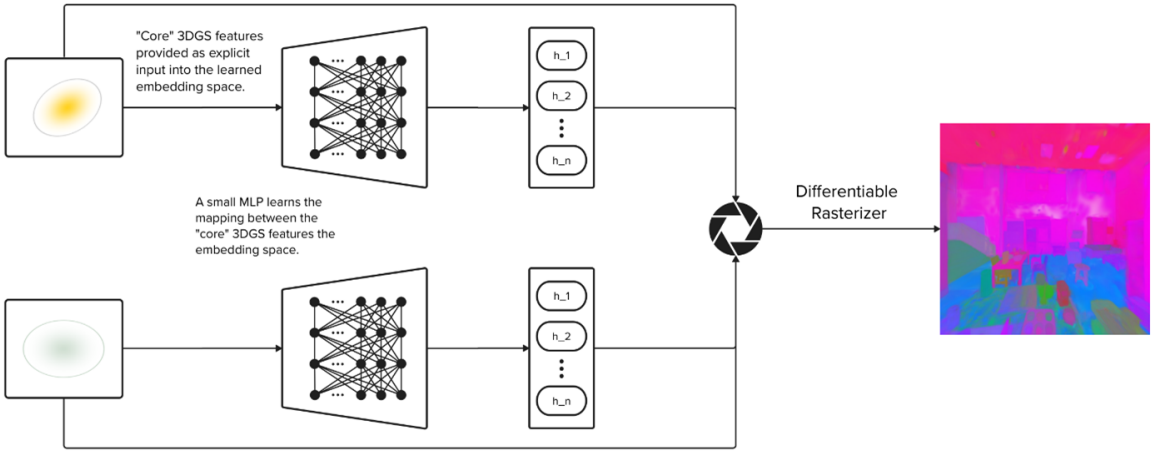


Figure 6. Our updated architecture with direct inputs from the core splat features into an MLP which then projects into the embedding space for learned lifted segments.

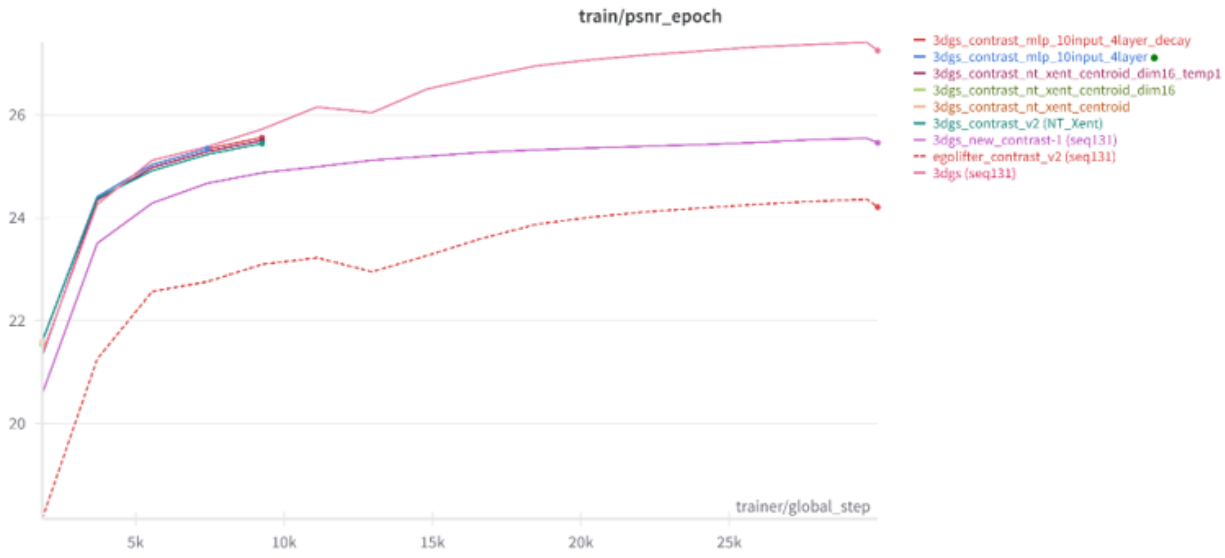
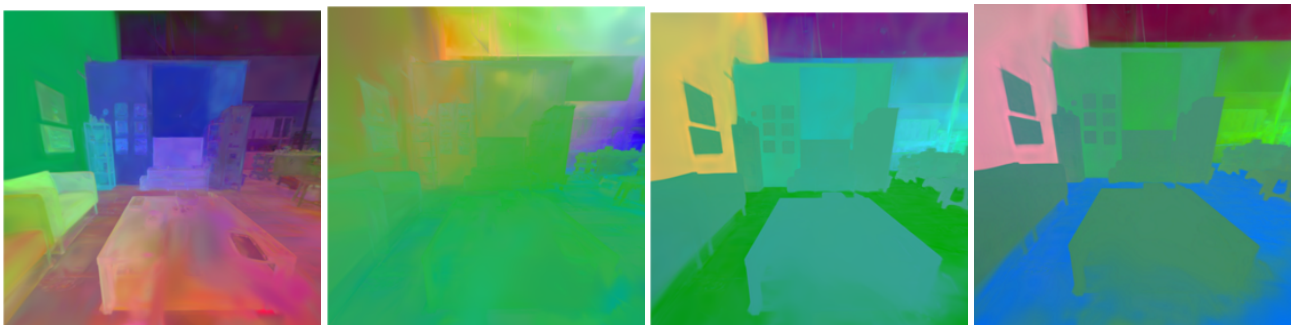


Figure 7. PSNR of several experimental runs. The best performing model is the original EgoLifter vanilla model (3dgs (seq131))



4 layer mlp	2 layer ml 3 dim embedding NT-Xent Loss	2 layer mlp 16 dim embedding NT-Xent Loss	2 layer mlp 16 dim embedding NT-Xent Loss Centroid Loss

Figure 8. Qualitative results of lifted segmentation features. These are rasterizations of the PCA projections of the learned embeddings.