ESAM++: Efficient Online 3D Perception on the Edge Supplementary Materials

Qin Liu[1] Lavisha Aggarwal[2] Saptarashmi Bandyopadhyay[2] Marc Niethammer[3]
Ehsan Adeli[1] Andrea Colaco[2]
[1]Stanford University [2]Google [3]UC San Diego

https://github.com/qinliuliuqin/esamplusplus

1. Comparisons against other lightweight 3D backbones.

Most existing popular 3D backbones (e.g., PointTransformer [3] and Sonata [4]) are primarily optimized for GPU inference or offline processing, while our approach focuses on the specific challenges of **online**, **CPU-only** 3D perception. For a rigorous evaluation, we compare our method against a lightweight MinkowskiNet [1] variant and SPVNAS [2], a model specifically optimized for mobile-friendly and hardware-efficient performance. Results on ScanNet200 demonstrate that our method consistently outperforms all baselines.

Method	Backbone	Model Params	CPU Latency	mAP
ESAM-E	Sparse UNet	44.6M	934ms	43.4
MinkNet18	Sparse ResNet	11.7M	487ms	27.1
SPVNAS	SPVCNN	12.5M	269ms	22.8
Ours	SFPN-Small	**14.1M**	**211ms**	30.3
Ours	SFPN-Base	23.5M	252ms	39.7
Ours	SFPN-Large	41.2M	326ms	**43.7**

Table 1. Comparisons with lightweight 3D backbones specifically optimized for CPU inference.

2. Data-efficient learning.

Following the experimental setup of ESAM [8], we evaluate the class-agnostic performance of ESAM++ on ScanNet200 using reduced training sets (10% and 50%) as reported in Table 2. The results indicate a relatively minor performance degradation even with only half the training data, a trend consistent with the findings of ESAM.

3. Loss Function

To effectively supervise the online 3D segmentation task, we use two complementary types of loss functions: per-frame loss and cross-frame loss. The **per-frame loss** supervises predictions independently at each time step by leveraging 2D SAM-based annotations lifted to 3D. In contrast, the **cross-frame loss** enforces temporal consistency across adjacent frames by introducing a contrastive objective that aligns instance-level features over time. We detail each component below.

Table 2. Performance under varying training data proportions.

Data Proportion	AP	AP_{50}	AP_{25}
100%	43.7	66.1	81.2
50%	40.6	64.4	80.1
10%	32.4	53.0	71.4

Per-frame loss. Each RGB-D frame is annotated with consistent semantic and instance labels across time. Given these annotations, we compute per-frame losses based on the predictions from each query. Since queries Q_t are lifted one-to-one from 2D SAM masks, we bypass complex label assignment and directly supervise each query using the corresponding 2D mask annotation. Assuming each 2D SAM mask corresponds to a single instance, we obtain the ground-truth semantic label and 2D instance mask for each query. These masks are projected to 3D using depth-based pixel correspondence, from which we derive the 3D instance mask and its axis-aligned bounding box. With the above annotations, we compute the binary classification loss $\mathcal{L}^t_{cls}$ with cross-entropy to distinguish foreground and background instances. The predicted 3D masks are supervised using binary cross-entropy $\mathcal{L}^t_{\text{bce}}$ and Dice loss $\mathcal{L}^t_{\text{dice}}$. Bounding box and semantic predictions are trained with IoU loss $\mathcal{L}^t_{\text{iou}}$ and binary cross-entropy $\mathcal{L}^t_{\text{sem}}$, respectively. The per-frame loss $\mathcal{L}_1$ is defined as follow:

$$\mathcal{L}_1 = \frac{1}{T}\sum_{t=1}^{T}(\alpha\mathcal{L}^t_{cls} + \mathcal{L}^t_{bce} + \mathcal{L}^t_{dice} + \beta\mathcal{L}^t_{iou} + \mathcal{L}^t_{sem}), \quad (1)$$

where we set both α and β to 0.5 without tuning the parameters.

Cross-frame loss. We formulate a contrastive loss across adjacent frames. This loss encourages feature consistency for the same instance across adjacent frames f_t and f_{t+1}. The cross-frame loss $\mathcal{L}_2$ is defined as follow:

$$\mathcal{L}_2 = \frac{1}{T}\sum_{t=1}^{T}(\mathcal{L}_{cont}{}^{t\to t+1} + \mathcal{L}_{cont}{}^{t\to t-1}). \tag{2}$$

$$\mathcal{L}_{cont}{}^{t\to t+1} = -\frac{1}{Z}\sum_{i=1}^{Z}\log\frac{e^{\left(\langle f_t^i, f_{t+1}^i\rangle/\tau\right)}}{\sum_{j\neq i} e^{\left(\langle f_t^i, f_{t+1}^j\rangle/\tau\right)} + e^{\left(\langle f_t^i, f_{t+1}^i\rangle/\tau\right)}} \tag{3}$$

where $\langle\cdot,\cdot\rangle$ is cosine similarity; the corner cases $\mathcal{L}_{cont}{}^{T\to T+1}$ and $\mathcal{L}_{cont}{}^{1\to 0}$ are set to zero.

Total loss. The total loss is formulated as:

$$\mathcal{L}_{total} = \lambda_1\mathcal{L}_1 + \lambda_2\mathcal{L}_2 \tag{4}$$

where we set both λ_1 and λ_2 to 0.5 without tuning the parameters.

4. Training Strategies

Following prior work [8, 9], we adopt a two-stage training strategy for ESAM++. In the first stage, we train a single-view perception model using ScanNet(200)-25k, a curated subset of ScanNet(200) containing individual RGB-D frames. At this stage, we exclude memory-based adapters and auxiliary task losses to focus solely on learning robust single-frame representations. This enables the model to develop a strong foundational understanding of object semantics and geometry from isolated views without the added complexity of temporal or memory-based reasoning.

In the second stage, we fine-tune the pretrained model on full RGB-D sequences, incorporating the memory-based adapters and the complete set of loss functions associated with auxiliary tasks. This stage is designed to enhance the model's ability to integrate information across frames and maintain consistent object representations over time. To ensure computational efficiency and manage memory usage during training, we randomly sample 8 consecutive RGB-D frames from each scene at every training iteration. This sampling strategy allows the model to learn from temporal context while keeping the memory footprint tractable.

All experiments are implemented using PyTorch [6]. Model training is conducted on an NVIDIA A6000 GPU, while inference is performed on Intel® Xeon® Silver 4314 CPUs running at 2.40 GHz, demonstrating the efficiency and deployability of our approach in resource-constrained environments. We use the AdamW optimizer [5] with an initial learning rate of 1e-4 and a weight decay of 0.05 to ensure stable optimization and effective regularization throughout both training stages.

5. Qualitative Results

We present three qualitative demos to illustrate the online 3D segmentation process, as shown in Figure 1, Figure 2, and Figure 3. These examples are randomly selected from the ScanNet200 datasets, demonstrating that ESAM++ effectively merges partial segmentation results into complete objects and produces fine-grained 3D masks within the online reconstructed scenes. Please refer to the accompanying demo videos for further details.

6. Combining with Online 3D Reconstruction Models

While our method currently relies on depth cameras for 3D reconstruction, it is both interesting and promising to explore its integration with online 3D reconstruction models [7] that operate solely on RGB inputs. Such a combination would enable the development of an online 3D perception system that functions using only RGB video, removing the dependency on depth sensors. We consider this an exciting direction for future work.

References

[1] 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019. 1

[2] Searching efficient 3d architectures with sparse point-voxel convolution. In *ECCV*, 2020. 1

[3] Point transformer v3: Simpler faster stronger. In *CVPR*, 2024. 1

[4] Sonata: Self-supervised learning of reliable point representations. In *CVPR*, 2025. 1

[5] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 2

[6] A Paszke. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019. 2

[7] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. *arXiv preprint arXiv:2501.12387*, 2025. 2

[8] Xiuwei Xu, Huangxing Chen, Linqing Zhao, Ziwei Wang, Jie Zhou, and Jiwen Lu. Embodiedsam: Online segment any 3d thing in real time. *arXiv preprint arXiv:2408.11811*, 2024. 1, 2

[9] Xiuwei Xu, Chong Xia, Ziwei Wang, Linqing Zhao, Yueqi Duan, Jie Zhou, and Jiwen Lu. Memory-based adapters for online 3d scene perception. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21604–21613, 2024. 2

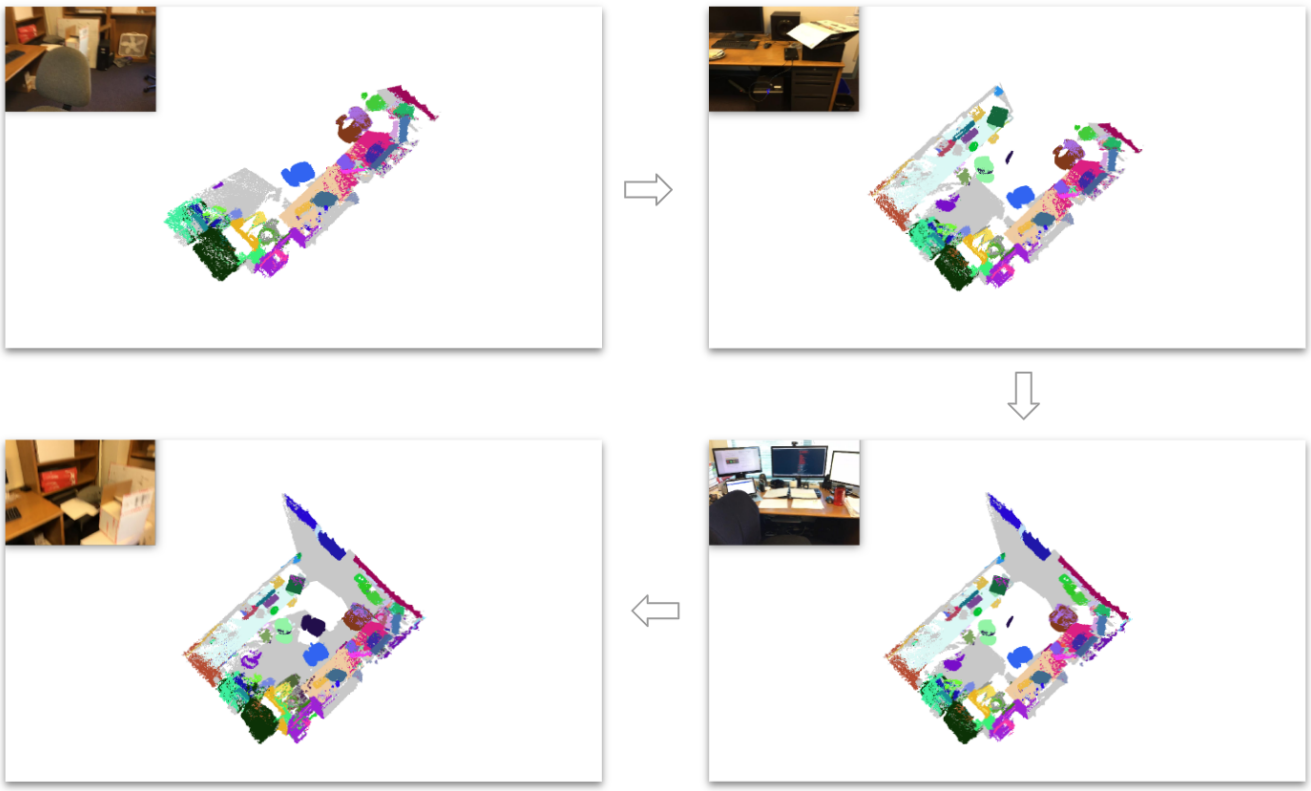

Figure 1. Online visualization—case study 1.

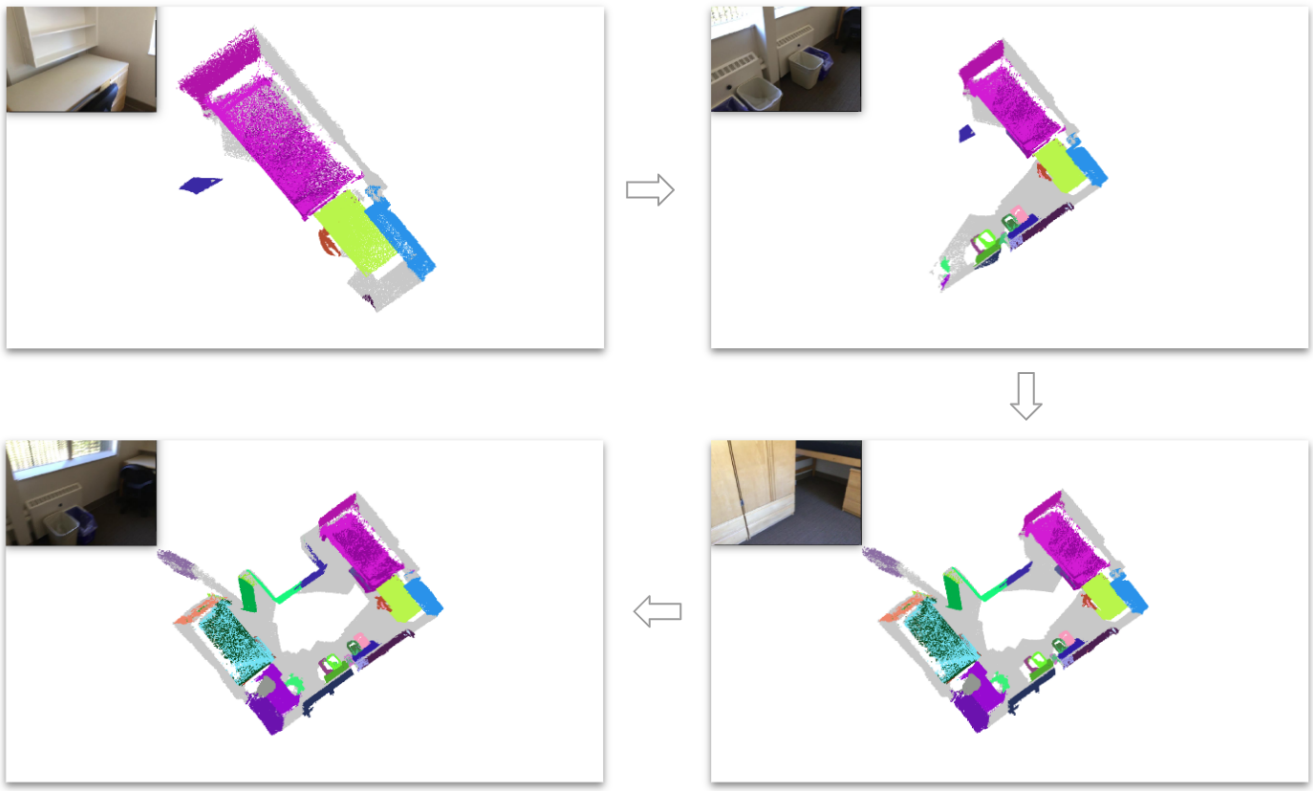

Figure 2. Online visualization—case study 2.

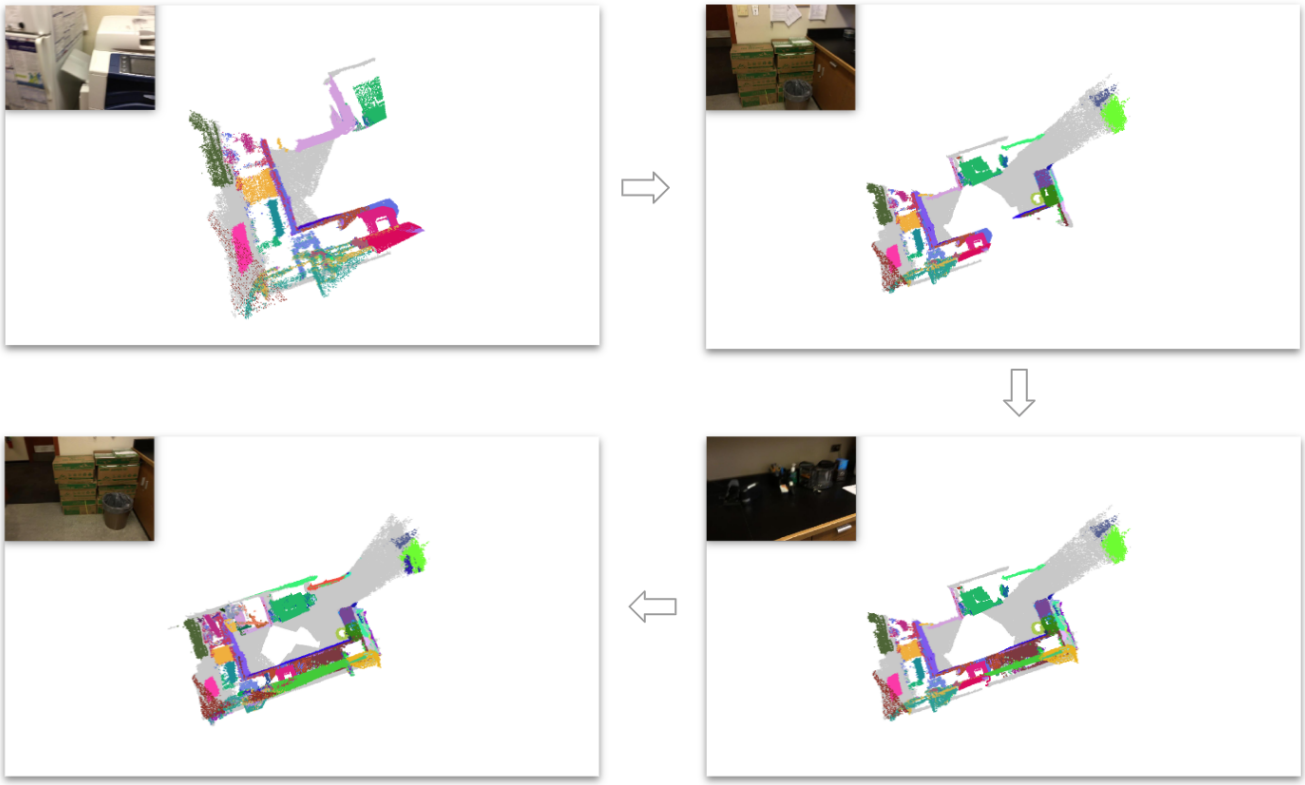

Figure 3. Online visualization—case study 3.