Feature4X: Bridging Any Monocular Video to 4D Agentic AI with Versatile Gaussian Feature Fields

Supplementary Material

This supplement is organized as follows:

- Section A contains more details of 4D reconstruction;
- Section B contains implementation details;
- Section C contains the details of foundation models' features extraction and downstream task inference;
- Section D contains the algorithmic details of language-guided 4D editing with LLM;
- Section E contains more baseline comparisons;
- Section F contains ablation studies of our method.

A. Details of 4D Reconstruction

Our video-to-4D feature field reconstruction pipeline is based upon MoSca [2], a state-of-the-art monocular video-based dynamic 3D scene reconstruction method. We augment their representation with our proposed unified feature field and follow their training procedure with added feature rendering loss to the original optimization objective. Here we give a brief overview of the training pipeline. Please refer to MoSca [2] for more details.

A.1. Dynamic Scene Representation

Given a monocular input video, the underlying dynamic 3D scene is modeled as the composition of a static 3D background, represented with a set of static 3D Gaussians $\{\mathcal{G}_{\text{static}}\}$, and a dynamic 3D foreground, represented by a set of 3D Gaussians $\{\mathcal{G}\}$ that deform over time.

The deformation of Gaussians $\{\mathcal{G}\}$ is modeled by a structured representation named 4D Motion Scaffold. It is a graph $(\mathcal{V}, \mathcal{E})$ where the nodes $\mathcal{V}$ are 3D motion trajectories $\mathbf{v}^{(i)} = [\mathbf{Q}_1^{(i)}, \dots, \mathbf{Q}_t^{(i)}], \mathbf{Q} = [\mathbf{R}, \mathbf{t}] \in SE(3)$. Intuitively, they describe the rigid transformations or 6DoF poses of points through time. They also each have a control radius attribute $r^{(i)}$, describing the range of influence.

Given two nodes, $\mathbf{v}^{(i)}, \mathbf{v}^{(j)}$, we define their distance as the maximum distance of their translation component $\mathbf{t}$ over all timesteps. Specifically, $D(i,j) = \max_\tau ||\mathbf{t}_\tau^{(i)} - \mathbf{t}_\tau^{(j)}||$. Intuitively, two nodes are close to each other only if they are close at all timesteps. Based on distance metric D, we construct a K-Nearest Neighbor (KNN) Graph $(\mathcal{V}, \mathcal{E})$, describing the mutual influence of nodes on each other.

The set of dynamic 3D Gaussians $\mathcal{G}$ can be thought of as the union of 3D Gaussians from all timesteps. Each Gaussian $G \in \mathcal{G}$ is originally spawned at a certain source timestep τ, with position μ and rotation $\mathbf{R}$. However, to render the scene at target timestep τ', we *fuse* Gaussians from other source timesteps as well. This helps with the partiality of the single view observation at τ'. To do so, we need to compute the motion or the deformation of Gaussian G from τ to τ', which we achieve by querying the Motion Scaffold $(\mathcal{V}, \mathcal{E})$.

From a high level, we find the node trajectories closest to G and use the interpolation of their deformation (given by $\mathbf{Q}_{\tau'}\mathbf{Q}_\tau^{-1}$) as the deformation of G. For computation efficiency, in practice, we first identify the node trajectory $\mathbf{v}^{(i^*)}$ closest to g based on their positions at τ. Specifically, $i^* = \arg\min_i ||\mathbf{t}_\tau^{(i)} - \mu||$. We then use i^*'s K-Nearest neighbors $\{\mathbf{v}^{(i)}\}_{i \in \mathcal{E}(i^*)}$ as an approximation of G's closest nodes. We compute the interpolation weights $\{w_i\}$ as

$$w_i = \text{Normalize}_{i \in \mathcal{E}(i^*)} \left(\exp(-\frac{||\mu - \mathbf{t}_\tau^{(i)}||_2^2}{2r^{(i)}}) + \Delta w_i \right), \tag{1}$$

Intuitively, nodes with closer spatial positions and larger control radius have more influence over G. Δw_i is a learnable offset jointly optimized with other model parameters, allowing more flexibility.

The transformation of G from time τ to τ' is computed as

$$\mathbf{T}_{\tau \to \tau'} = \text{DQB}(\{w_i, \mathbf{Q}_{\tau'}\mathbf{Q}_\tau^{-1}\}) \in SE(3),$$

where DQB represents Dual Quaternion Blending that interpolates $SE(3)$ elements. We apply $\mathbf{T}_{\tau \to \tau'}$ to the position and rotation of G when rendering it at timestep τ'.

A.2. Initialization from Lifted 2D Priors

We rely on 2D priors from pretrained foundation models to initialize and constrain the system. The most important priors include 1) per-frame monocular metric depths $\mathcal{D}$ for rough geometry initialization; 2) long-term pixel trajectories $\mathcal{T}$, where each trajectory describes a point's pixel position at each frame, providing a rough motion initialization in 2D; and 3) per-frame epipolar error maps computed from dense optical flow, separating static background and dynamic foregrounds.

We assume known camera intrinsic and poses for the following and refer the readers to [2] for camera initialization and optimization in case the cameras are unknown.

We first compute foreground-background masks from epipolar error maps $\mathcal{M}$. We initialize the static background 3D Gaussians by back-projecting points in the static background masks using depth estimations.

For the dynamic foreground, we lift 2D pixel trajectories to 3D space with monocular depth predictions and filter out dynamic ones based on epipolar error maps, then sample a subset of them based on trajectory distance metric

D as the initial Motion Scaffold nodes. As these trajectories only contain 3D positions, we initialize the rotational components $\mathbf{R}$ with identity matrices. We initialize the dynamic Gaussians by back-projecting points in the dynamic foreground masks.

A.3. Optimization

We first optimize the static background 3D Gaussians following the standard 3D Gaussian Splatting optimization procedures, using supervision from static regions of the input video frames.

The dynamic foreground is optimized in two stages. The first geometric optimization stage focuses on the Motion Scaffolds and produces a reasonable deformation field, based on which the second photometric/feature stage spawns dynamic Gaussians and optimizes the final RGB and feature rendering objectives. During geometric optimization, the Scaffold trajectories are optimized subject to a set of physics-inspired losses: an as-rigid-as-possible loss that encourages nearby nodes to deform in locally-rigid ways, and smoothness losses on the velocity and acceleration of points to encourage smooth motions. During photometric optimization, we deform the dynamic Gaussians with Motion Scaffolds, combine them with the static Gaussians, rasterize them into RGB images and feature maps, and jointly optimize all parameters including Gaussian parameters and the Motion Scaffolds.

B. Implementation Details

Feature Decoder Configuration While training our unified latent feature field, the rendered feature map from a camera view is passed through a decoder head to obtain the final feature map for each feature type. Our decoder heads are simple MLPs, which enlarge the input feature dimension by a factor of 2 at each layer. For example, our latent feature dimension is 32 but the CLIP feature dimension is 512, so the decoder simply maps 32 to 64, 64 to 128, and finally 128 to 512. Between each linear layer in the decoder, we use a ReLU activation function.

N-Dimensional Feature Map Rendering We utilize the parallel N-dimensional Gaussian rasterizer from Feature 3DGS [13], which leverages a point-based α-blending approach to rasterize feature maps. This method ensures that the rendered feature maps and RGB images are produced at the same resolution. Before calculating the loss, we align the size of the rendered feature map with the ground truth feature map. For SAM2 [9] and CLIP-LSeg [3] features, this is achieved using bilinear interpolation, while area resampling is applied to InternVideo [11] features due to the patchify mechanism used during feature extraction. Moreover, the parallel N-dimensional rasterizer is designed with adaptability, enabling flexible adjustments to the various dimensions of our unified latent feature field.

Training Our model is trained end-to-end to jointly reconstruct the versatile Gaussian feature field and the radiance field. We adopt the same training schedule as MoSca [2]: 4000 iterations for static Gaussian optimization and 5000 for dynamic one on the DAVIS dataset [8], and 8000 iterations for static and 5000 for dynamic Gaussian optimization on the NVIDIA dataset [12].

C. Feature Extraction and Inference with Foundation Models

SAM2 Feature We extract per-frame ground truth features sequentially using the image encoder from SAM2 [9]. Specifically, each frame is processed by a Hiera [10] image encoder, which is pre-trained with MAE [1], to produce a feature map with a resolution of 64×64 and a feature dimension of 256. Consequently, the ground truth features for the input video have a shape of $T \times 256 \times 64 \times 64$, where T denotes the number of frames.

Our promptable / promptless segmentation results are obtained by feeding the rendered SAM2 feature map into the SAM2 decoding architecture, which includes a memory attention module, a prompt encoder, and a mask decoder. These components interact with a memory encoder and memory bank to retain and propagate segmentation information. For promptable segmentation, points, boxes, or masks are input into the prompt encoder to define the object's extent in a frame. For promptless segmentation, SAM2's automatic mask generator produces segmentation masks for a frame, which are then input into the prompt encoder. Once segmentation is performed on any initial frame, SAM2's memory modules enable automatic tracking and propagation of masks across subsequent frames by conditioning each frame's features on past predictions.

CLIP-LSeg Feature For CLIP-LSeg, we utilize the CLIP ViT-L/16 image encoder to generate ground truth per-pixel feature maps and the ViT-L/16 text encoder for text feature extraction. This encoder can automatically resize any input image to generate a feature map with the longer side set to 480. We use square input images in our experiments so the CLIP-LSeg image encoder produces ground truth features with a resolution of 480×480 and a feature dimension of 512.

During inference for semantic segmentation, the rendered features with shape $(512, 480, 480)$ are reshaped into $(480 \times 480, 512)$, referred to as the image feature. Meanwhile, the text feature extracted from the CLIP text encoder has a shape of $(C, 512)$, where C is the number of categories. A matrix multiplication is then performed between

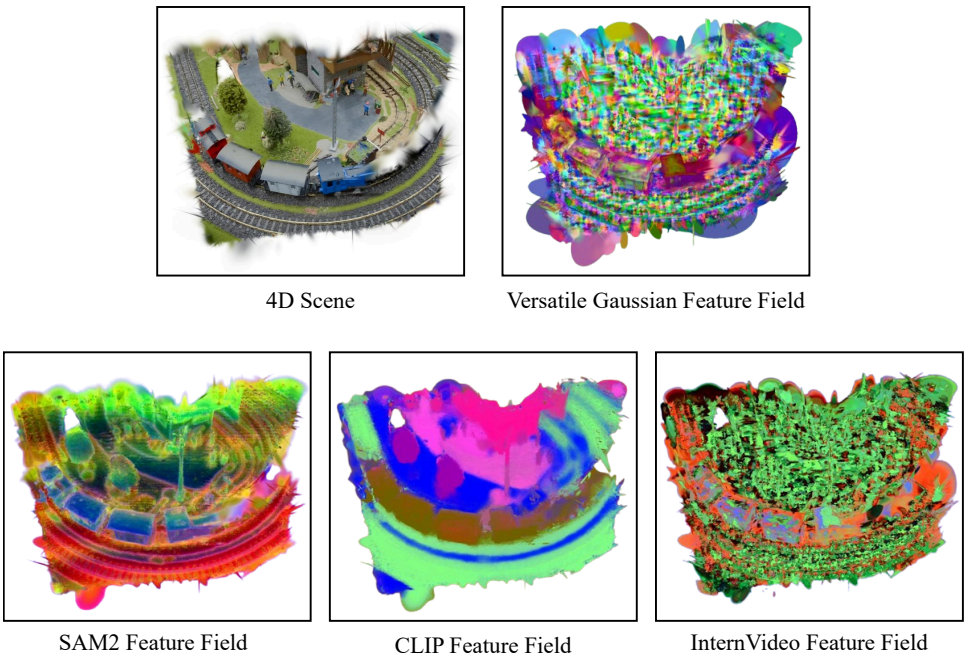

Figure A. **Feature Field Visualizations**. We visualize our versatile Gaussian feature field along with its decoded SAM2, CLIP, and InternVideo feature fields using PCA.

the image feature and the text feature to align pixel-level features with text queries. The resulting features are further processed using LSeg spatial regularization blocks to generate the semantic segmentation masks.

InternVideo Feature Given an input video, internvideo first resizes it to 224×224 resolution, then takes 14×14 as a patch, passing through a convolutional neural network to get the initial feature map with 1408 channels. Then, a pretrained class token input is concatenated to the initial feature map and passed through a transformer encoder to get the final feature map, which is in the shape of $(T \times 16 \times 16 + 1) \times 1408$, where T is the number of frames. As the class token represents the whole video class information and is not dependent on individual pixels, we save and detach it when distilling the 4D feature field.

During inference, we first concatenate the class token back into the novel-view rendered InternVideo feature to obtain the gathered feature. This gathered feature is then directly input into a Video-LLM [4] to perform free-form visual question answering (VQA). Since the feature map can be rendered from any viewpoint of the 4D scene at high speed, our approach enables a seamless connection between the 4D scene and the AI agent (chatbot).

Feature Fields Visualization As shown in Fig. A, we leverage Principal Component Analysis (PCA) from the scikit-learn library [7] to visualize feature fields from global novel views. We configure the PCA to use 3 components corresponding to RGB channels and compute the PCA mean by sampling every third element along the $h \times w$ vectors. These vectors have feature dimensions of 32 (for unified latents), 256 (for SAM2), 512 (for CLIP), and 1408 (for InternVideo). The feature map is then transformed using the calculated PCA components and mean. This process involves centering the features using the PCA mean, followed by projection onto the PCA components. The transformed feature is subsequently normalized by removing outliers and scaling based on the minimum and maximum values, standardizing the feature values into a uniform range suitable for visualization.

D. Details of LLM-powered 4D Editing

An overview of the editing pipeline is illustrated in Fig. B. It begins with user inputs, such as "Make the dog's color look like Clifford," which are processed by a GPT-4o model to extract editable configurations (e.g., objects, operations, targets, and thresholds). Each Gaussian x_i is defined by (f_i, α_i, c_i), where $f_i \in \mathbb{R}^{512}$ is the semantic feature, $\alpha_i \in \mathbb{R}$ is the opacity, and $c_i \in \mathbb{R}^3$ is the color. Guided by input specified object categories (e.g., "dog", "cow"), the CLIP ViT-B/32 text encoder encodes the text into features $\{t_1, \ldots, t_C\}$, where $t_i \in \mathbb{R}^{512}$ and C is the number of categories. The inner product between text and semantic features, followed by a softmax, produces a semantic score matrix $scores \in \mathbb{R}^{N \times C}$.

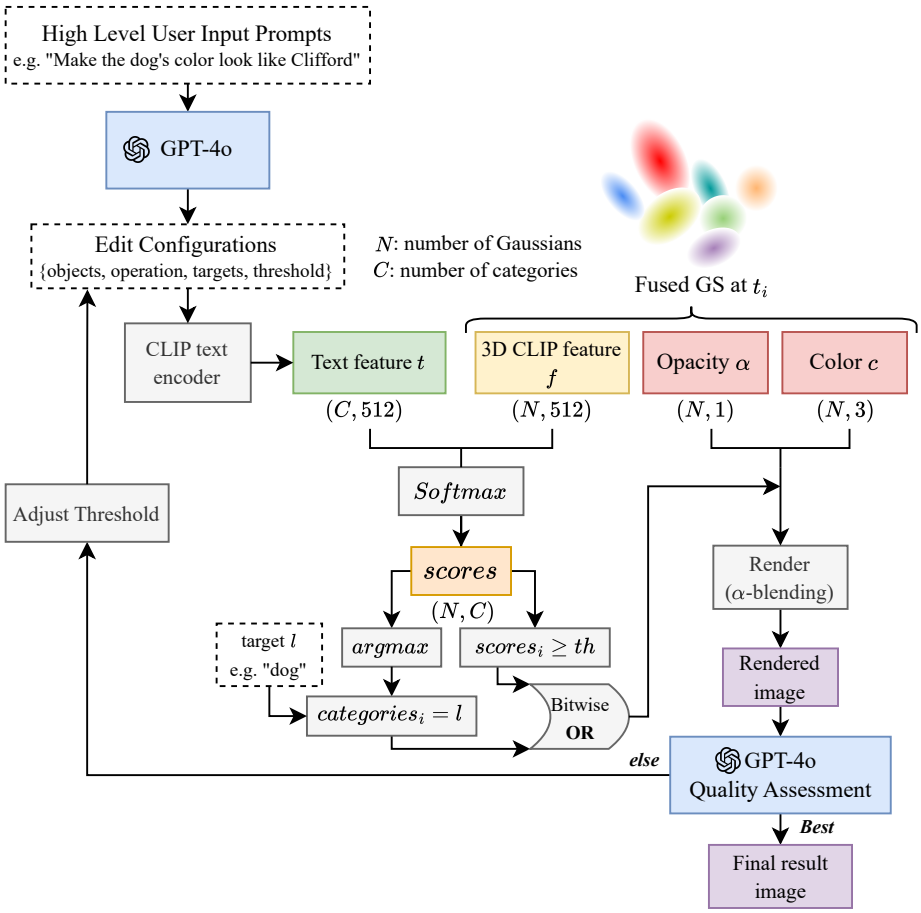

Figure B. **Overview of the editing framework.** GPT-4o generates different editing configurations based on user prompts, selects target regions via hybrid filtering, evaluates their outputs, and selects the best configuration.

To select Gaussians that corresponds to the editing, we use a combination of two masking schemes derived from the semantic scores. The first one is binary thresholding. The query label $l \in \{1, 2, \ldots, C\}$ (or $l \subseteq \{1, 2, \ldots, C\}$ if it is a list of categories) determines the corresponding column of the score matrix $score_l = [s_{1l}, s_{2l}, \ldots, s_{Nl}]^\top$. Indices i where $s_{il} \geq th$ are selected (set to 1), while the rest are excluded (set to 0). The selected indices define the target region, and unselected Gaussians are masked out. The second masking scheme is determined by assigning each Gaussian to the category with the highest score using $argmax$ as shown in Fig. B, producing a category vector $categories = [c_1, c_2, \ldots, c_N]^\top$, where $c_i = \text{argmax}\{s_{i1}, \ldots, s_{iC}\}$. Gaussians are selected if their category aligns with the query label l, and others are excluded. Their resulting masks are combined using a bitwise OR operation. This method balances flexibility and precision, enabling more robust selection.

To optimize editing parameters, GPT-4o generates multiple candidate configurations with varying thresholds or transformations. Each candidate is used to execute an editing operation, and the resulting edited images are rendered. GPT-4o then evaluates these outputs and selects the best configuration, which is subsequently applied across the entire video to ensure consistent and high-quality results.

E. Baseline Comparisons

Segment Anything (SAM2) In the main paper, we highlighted that our SAM2 inference implementation from the rendered unified latent feature map only needs to interact with the SAM2 decoding architecture. This is an improvement over the naive approach, which renders the RGB images (ordinary novel view synthesis like MoSca [2]) first and then processes it through the entire SAM2 encoder-decoder pipeline.

We quantitatively evaluate our approach (denoted as Feature) against the naive baseline (rendered novel view RGB + SAM2, denoted as RGB) on two datasets (Nvidia [5] and Nerfies [6]) in Tab. A. A point prompt is randomly selected on the dynamic object in the first frame of the ground-truth

NVIDIA	Exp1	Exp2	Exp3	Mean↑	Time (s)↓
RGB	0.656	0.246	0.467	0.456	1.83
Feature	**0.761**	**0.728**	**0.727**	**0.739**	**1.01**

Nerfies	Exp1	Exp2	Exp3	Mean↑	Time (s)↓
RGB	0.484	0.536	0.538	0.519	9.10
Feature	**0.560**	**0.662**	**0.561**	**0.594**	**3.10**

Table A. **SAM2 Quantitative Results (mIoU) on NVIDIA and Nerfies Datasets.**

Scene	Method	PSNR↑	SSIM↑	LPIPS↓	accuracy↑	mIoU↑	Static Model Size (MB)	Dynamic Model Size (MB)	Size (MB)
Jumping	MoSca [2]	24.558	0.792	0.092	-	-	29.08	29.70	58.78
Jumping	MoSca + Feature 3DGS [13]	24.516	0.793	0.092	0.840	0.483	271.30	212.58	483.88
Jumping	Ours (single CLIP head)	24.633	0.795	0.090	0.836	0.495	44.15	30.51	74.66
Jumping	Ours (full model)	24.616	0.793	0.090	0.831	0.483	44.09	30.84	74.93
Skating	MoSca [2]	31.478	0.926	0.059	-	-	32.49	4.90	37.39
Skating	MoSca + Feature 3DGS [13]	31.568	0.927	0.059	0.835	0.446	302.50	35.40	337.90
Skating	Ours (single CLIP head)	31.572	0.927	0.059	0.838	0.450	49.32	4.90	54.22
Skating	Ours (full model)	31.666	0.926	0.059	0.819	0.418	47.52	5.60	53.12
Truck	MoSca [2]	26.688	0.824	0.115	-	-	38.31	11.84	50.15
Truck	MoSca + Feature 3DGS [13]	26.619	0.824	0.115	0.973	0.880	353.97	90.18	444.15
Truck	Ours (single CLIP head)	26.630	0.820	0.122	0.971	0.878	58.35	12.27	70.62
Truck	Ours (full model)	26.610	0.822	0.117	0.969	0.868	58.16	13.51	71.67
Umbrella	MoSca [2]	23.355	0.706	0.185	-	-	71.29	11.42	82.71
Umbrella	MoSca + Feature 3DGS [13]	23.362	0.708	0.176	0.875	0.556	657.96	75.12	733.08
Umbrella	Ours (single CLIP head)	23.433	0.707	0.185	0.869	0.559	107.87	11.28	119.15
Umbrella	Ours (full model)	23.392	0.708	0.180	0.880	0.565	107.58	11.50	119.08
Balloon1	MoSca [2]	22.666	0.760	0.117	-	-	56.88	18.59	75.47
Balloon1	MoSca + Feature 3DGS [13]	22.687	0.762	0.115	0.901	0.377	525.09	129.88	654.97
Balloon1	Ours (single CLIP head)	22.668	0.760	0.118	0.905	0.435	85.95	18.84	104.79
Balloon1	Ours (full model)	22.691	0.759	0.117	0.903	0.446	85.72	19.90	105.62
Balloon2	MoSca [2]	26.827	0.850	0.082	-	-	51.82	15.22	67.04
Balloon2	MoSca + Feature 3DGS [13]	27.018	0.854	0.080	0.819	0.350	475.71	108.96	584.67
Balloon2	Ours (single CLIP head)	26.904	0.851	0.081	0.821	0.321	78.22	15.18	93.40
Balloon2	Ours (full model)	26.871	0.853	0.078	0.813	0.319	77.77	15.36	93.13
Playground	MoSca [2]	20.591	0.777	0.130	-	-	92.74	9.80	102.54
Playground	MoSca + Feature 3DGS [13]	20.569	0.776	0.124	0.922	0.447	857.32	61.38	918.70
Playground	Ours (single CLIP head)	20.463	0.775	0.130	0.922	0.430	141.02	9.20	150.22
Playground	Ours (full model)	20.536	0.775	0.130	0.913	0.419	140.44	10.21	150.65
Mean	MoSca [2]	25.166	0.805	0.111	-	-	53.230	14.496	67.726
Mean	MoSca + Feature 3DGS [13]	25.191	0.806	0.109	0.881	0.506	491.979	101.929	593.907
Mean	Ours (single CLIP head)	25.186	0.805	0.112	0.880	0.510	80.697	14.597	95.294
Mean	Ours (full model)	25.197	0.805	0.110	0.876	0.503	80.183	15.274	95.457

Table B. **Detailed Performance of 7 scenes from the NVIDIA Dataset.**

novel view video, and SAM2 is used to generate per-frame ground-truth masks. We evaluate *mIoU* across the entire dataset and repeat the experiments three times (Exp1-3) to assess generalizability. Our approach outperforms the baseline, which, though yielding smoother masks, lacks robustness. We observe that artifacts in RGB rendering can mislead SAM2 during encoding, causing ambiguity and highly inaccurate segmentation. In contrast, our feature space inference mitigates these issues, enhancing robustness and increasing speed.

Semantic Segmentation (CLIP-Lseg) In main paper Tab. 1, we provide the average performance metrics of our method and the comparison baselines across the 7 scenes of NVIDIA dataset. Here in Tab. B, we present the detailed performance metrics on each scene. We can conclude that our full model with a versatile feature field is not only fast and compact but also on par with any other task-specific feature field distillation methods. Additionally, Fig. C presents the qualitative results of semantic segmentation on "Jumping" scene using our full model, demonstrating the reliability of our approach.

We provide additional quantitative results on the Nerfies dataset [6] in Tab. C to evaluate our method's generalization capability. While we do not claim improving downstream task performance as a contribution, our method surpasses the naive baseline (novel view rendered per-frame RGB + LSeg) on this dataset while achieving 7.7× faster inference.

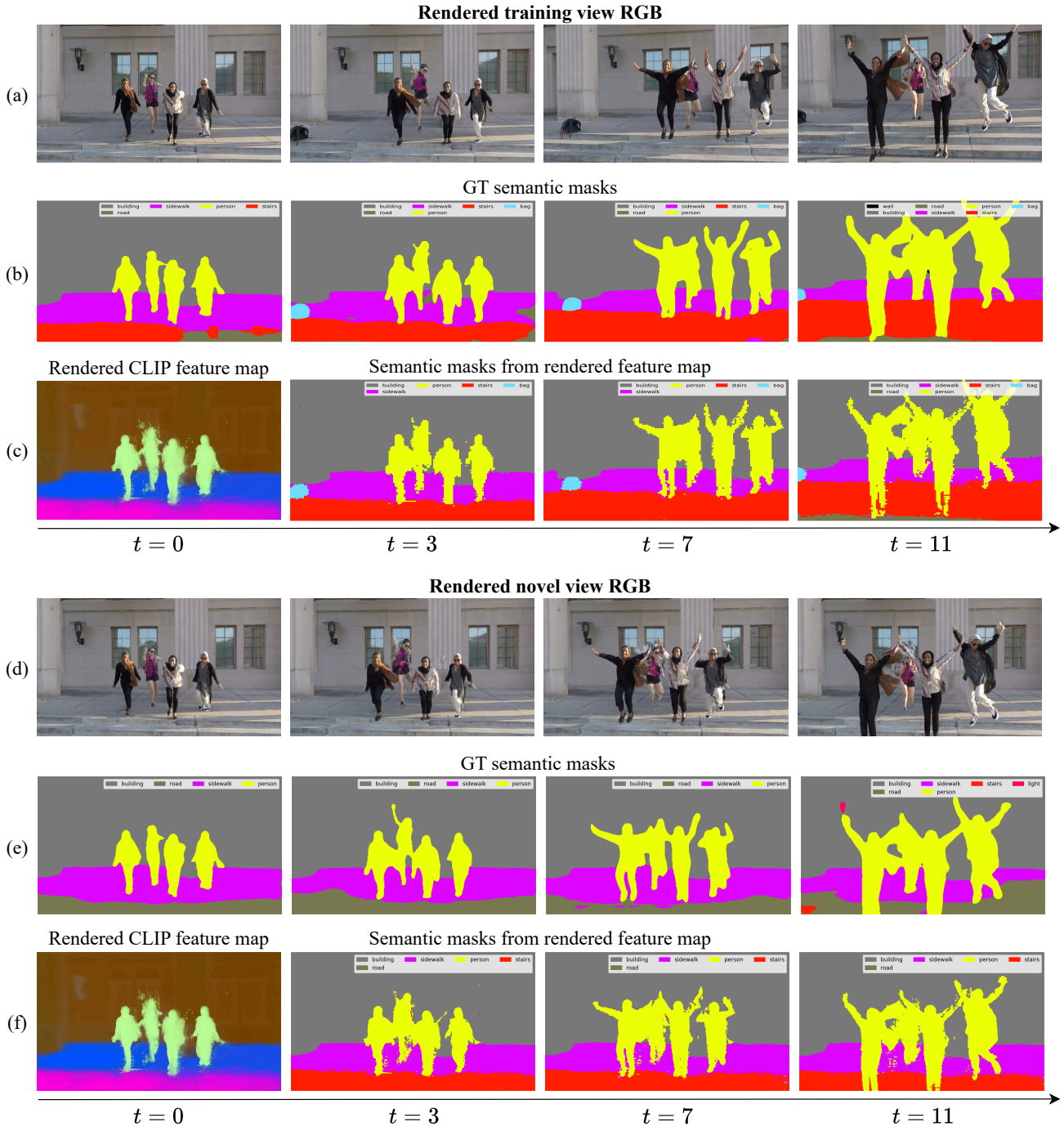

Figure C. **CLIP semantic segmentation quality comparison**. We compare the CLIP semantic segmentation quality between ground-truth (inference from RGB) and our implementation (inference from feature map) for both training and novel views.

F. Ablation Studies

In all our experiments, we set the default dimension of the unified latent feature to 32, striking a balance between speed and quality. In this section, we present an ablation study to explore the impact of varying the feature dimensions. In Fig. E and Fig. F we compare the training and rendering time for different dimensions of our unified latent feature. Notably, a dimension of 32 marks the threshold where further increases in dimensions lead to a significant increase in training and rendering times. This makes larger dimensions impractical for our use case.

Segment Anything (SAM2) In Fig. D we compare and contrast the SAM2 inference experiment results derived from different dimensions unified latent feature maps (8, 16, 32, 64, 128, 256, 512) to justify our choice of 32 dimensions. For dimensions higher than 32, we can see that the segmentation masks are of poor quality and cannot be accurately propagated through the video frames. For feature dimension 8, we see that while the segmentation mask tracks our intended object, it also erroneously includes much of the empty background. The segmentation masks for the 16 dimension experiment track accurately but do not fully cover the intended object at $t = 55$ and $t = 85$. Overall, the segmentation masks produced from 32 dimensional unified latent feature maps are the best in terms of segmentation quality and tracking accuracy. Given the relatively fast training and rendering, 32 should be the optimal dimension for our unified latent feature field in regards to SAM2 segmentation results.

Semantic Segmentation (CLIP-LSeg) We study the effect of different latent scene feature dimensions on the "Jumping" scene of NVIDIA. In Tab. D (with plots Fig. G and Fig. H), we report the training time as well as the performance of the semantic segmentation. The results show that compared to 512, which is the original dimension of CLIP-LSeg feature, our method with $dim = 32$ achieves comparable performance on mIoU and accuracy, while being $5.23\times$ faster on training. We also report the image quality metrics in Tab. E (with plots Fig. I and Fig. J) which shows that our feature field distillation method does not affect the image quality.

Nerfies Scene	mIoU ↑		Accuracy ↑		Time (s) ↓	
	RGB	Feature	RGB	Feature	RGB	Feature
Broom	0.193	**0.333**	0.321	**0.610**	207.59	**30.22**
Curls	**0.514**	0.443	**0.877**	0.872	155.25	**20.82**
Tail	0.261	**0.338**	0.652	**0.860**	389.89	**46.43**
Toby-sit	**0.504**	0.470	**0.757**	0.737	355.82	**45.96**
Mean	0.368	**0.396**	0.652	**0.770**	277.14	**35.86**

Table C. **Semantic Segmentation Quantitative Results on Nerfies Dataset.**

Dimension	8	16	32	64	128	256	512
Training Time (h)	2.30	2.48	2.83	3.45	4.22	8.75	14.80
Rendering Time (s)	4.818	4.898	4.872	4.967	5.815	10.167	16.313
mIoU↑	0.468	0.483	0.482	0.485	0.471	0.494	**0.497**
Accuracy↑	0.827	0.831	0.830	0.834	0.832	0.837	**0.841**

Table D. **Evaluation of Semantic Segmentation Performance On NVIDIA Jumping Scene Across Different Dimensions.** This table presents the Time, mIoU, and Accuracy corresponding to each dimension level.

Dimension	8	16	32	64	128	256	512	MoSca
PSNR↑	24.466	24.595	24.616	24.623	24.585	**24.629**	24.491	24.558
SSIM↑	0.788	0.793	0.793	0.790	0.792	**0.796**	0.789	0.792
LPIPS↓	0.092	0.091	0.090	0.091	0.091	0.090	**0.095**	0.092

Table E. **Evaluation of Image Quality Metrics On NVIDIA Jumping Scene Across Different Dimensions.** This table presents the PSNR, SSIM, and LPIPS values corresponding to each dimension level.

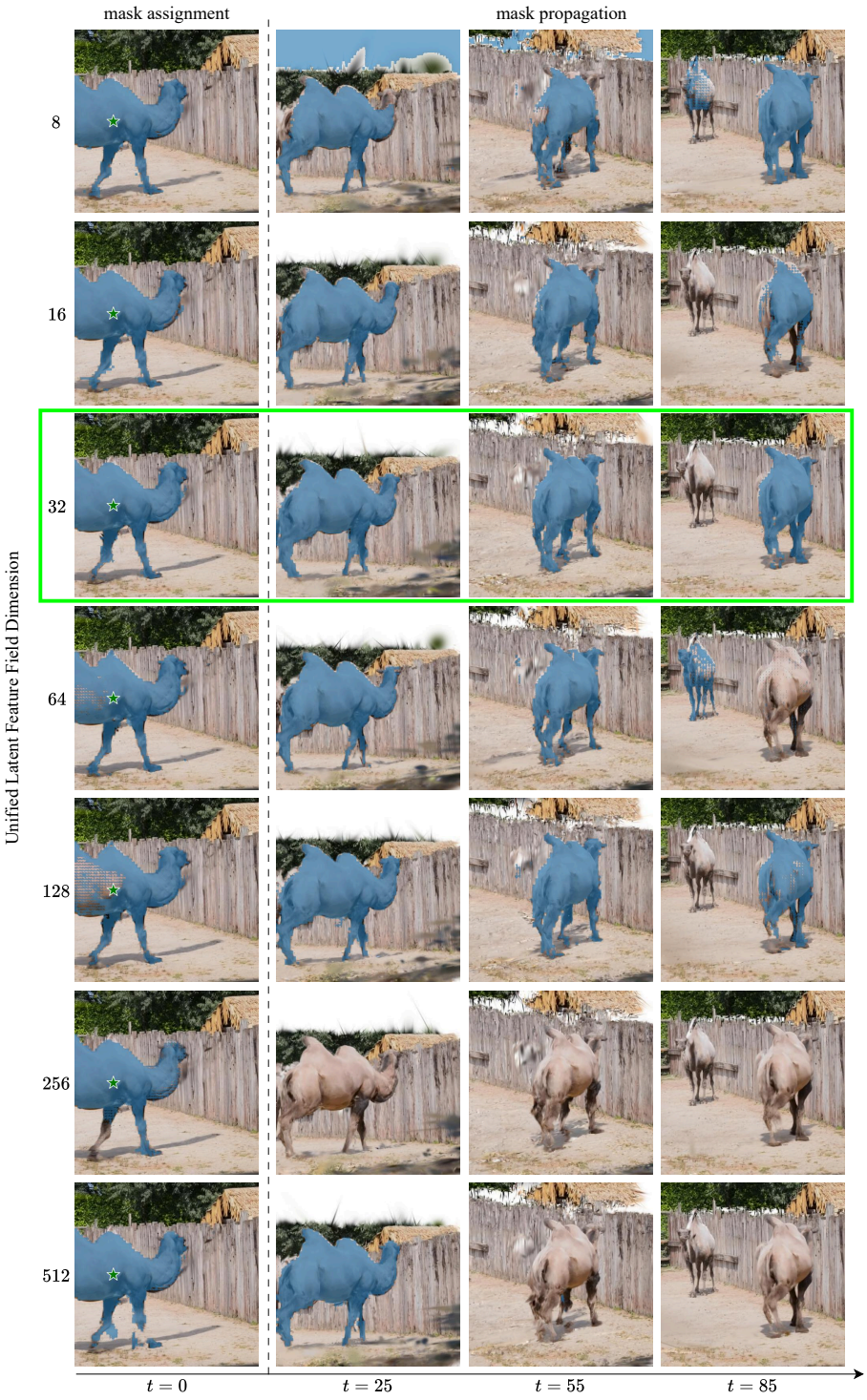

Figure D. **SAM2 segmentation quality comparison for different dimensions of unified latent feature maps** Best performing SAM2 segmentation is derived from the 32-dimensional unified latent feature map.

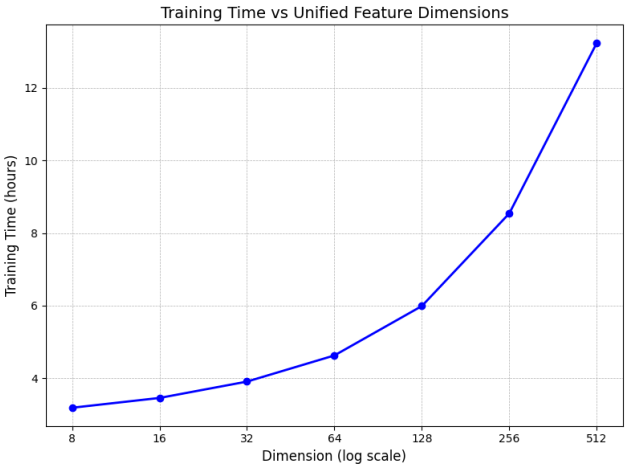

Figure E. **Training Time vs Unified Latent Feature Dimensions** We show the training time required with different dimensions of unified latent feature map.

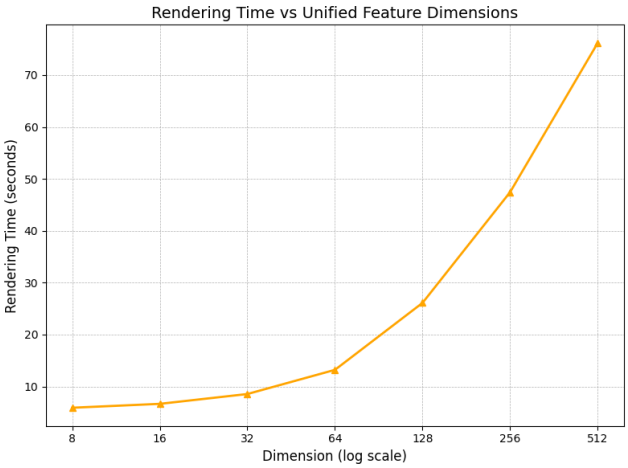

Figure F. **Rendering Time vs Unified Latent Feature Dimensions** We show the rendering time required for different dimensions of unified latent feature map.

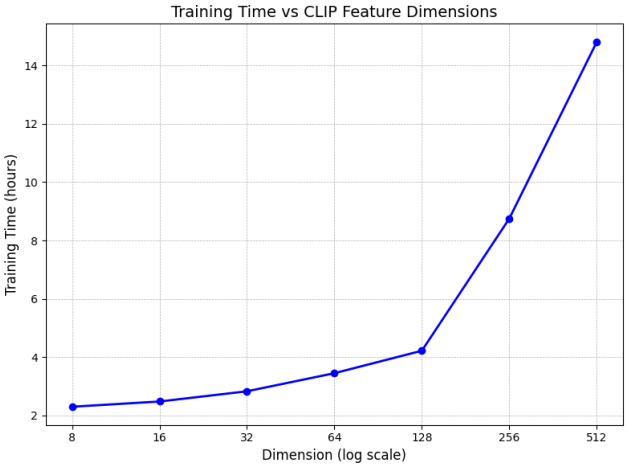

Figure G. **Training Time vs CLIP Feature Dimensions** We show the training time required with different dimensions of rendered CLIP features.

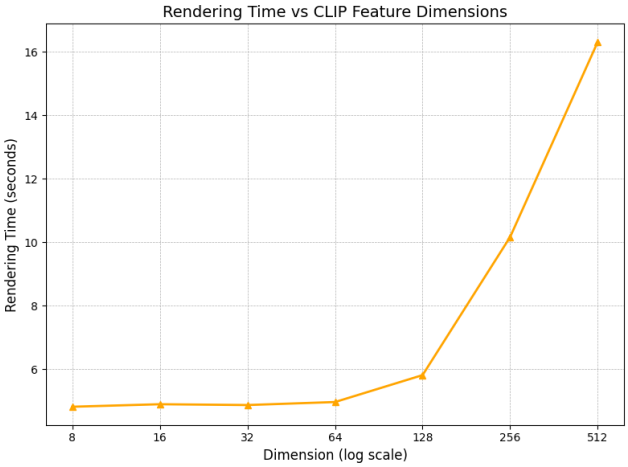

Figure H. **Rendering Time vs CLIP Feature Dimensions** We show the rendering time required for different dimensions of rendered CLIP features.

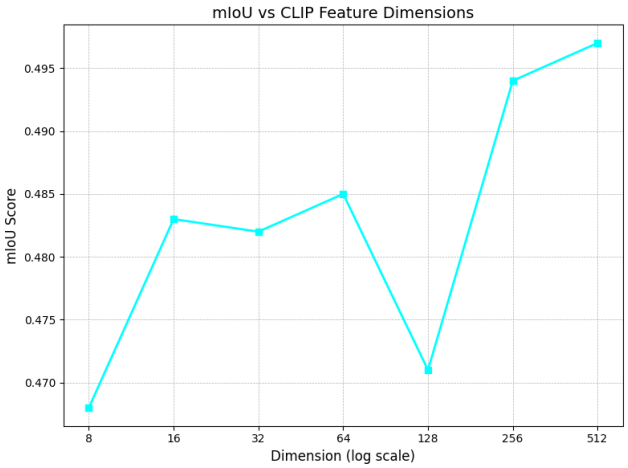

Figure I. **mIoU vs CLIP Feature Dimensions** We show mIoU with respect to different rendered CLIP feature dimensions.

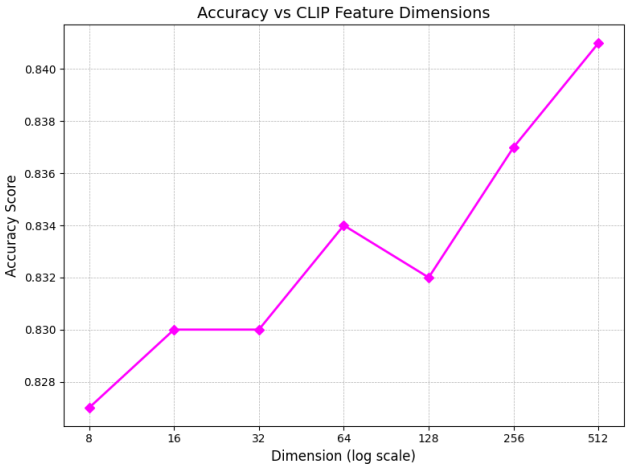

Figure J. **Accuracy vs CLIP Feature Dimensions** We show accuracy with respect to different rendered CLIP feature dimensions.

References

[1] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2022. https://arxiv.org/abs/2111.06377. 2

[2] Jiahui Lei, Yijia Weng, Adam Harley, Leonidas Guibas, and Kostas Daniilidis. Mosca: Dynamic gaussian fusion from casual videos via 4d motion scaffolds. *arXiv preprint arXiv:2405.17421*, 2024. 1, 2, 4, 5

[3] Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and René Ranftl. Language-driven semantic segmentation. *arXiv preprint arXiv:2201.03546*, 2022. 2

[4] KunChang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhai Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. Videochat: Chat-centric video understanding. *arXiv preprint arXiv:2305.06355*, 2023. 3

[5] Yu-Lun Liu, Chen Gao, Andreas Meuleman, Hung-Yu Tseng, Ayush Saraf, Changil Kim, Yung-Yu Chuang, Johannes Kopf, and Jia-Bin Huang. Robust dynamic radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 4

[6] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2021. 4, 5

[7] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011. 3

[8] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 724–732, 2016. 2

[9] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. 2

[10] Chaitanya Ryali, Yuan-Ting Hu, Daniel Bolya, Chen Wei, Haoqi Fan, Po-Yao Huang, Vaibhav Aggarwal, Arkabandhu Chowdhury, Omid Poursaeed, Judy Hoffman, Jitendra Malik, Yanghao Li, and Christoph Feichtenhofer. Hiera: A hierarchical vision transformer without the bells-and-whistles. *arXiv preprint arXiv:2306.00989*, 2023. https://arxiv.org/abs/2306.00989. 2

[11] Yi Wang, Kunchang Li, Xinhao Li, Jiashuo Yu, Yinan He, Chenting Wang, Guo Chen, Baoqi Pei, Rongkun Zheng, Jilan Xu, Zun Wang, et al. Internvideo2: Scaling video foundation models for multimodal video understanding. *arXiv preprint arXiv:2403.15377*, 2024. 2

[12] Jae Shin Yoon, Kihwan Kim, Orazio Gallo, Hyun Soo Park, and Jan Kautz. Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5336–5345, 2020. 2

[13] Shijie Zhou, Haoran Chang, Sicheng Jiang, Zhiwen Fan, Zehao Zhu, Dejia Xu, Pradyumna Chari, Suya You, Zhangyang Wang, and Achuta Kadambi. Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21676–21685, 2024. 2, 5