

# Gradient-Driven 3D Segmentation and Affordance Transfer in Gaussian Splatting Using 2D Masks

Joji Joseph<sup>1</sup>, Bharadwaj Amrutar<sup>2</sup> and Shalabh Bhatnagar<sup>3</sup>

**Abstract**—3D Gaussian Splatting has emerged as a powerful 3D scene representation technique, capturing fine details with high efficiency. In this paper, we introduce a novel voting-based method that extends 2D segmentation models to 3D Gaussian splats. Our approach leverages masked gradients, where gradients are filtered by input 2D masks, and these gradients are used as votes to achieve accurate segmentation. As a byproduct, we discovered that inference-time gradients can also be used to prune Gaussians, resulting in up to 21% compression. Additionally, we explore few-shot affordance transfer, allowing annotations from 2D images to be effectively transferred onto 3D Gaussian splats. The robust yet straightforward mathematical formulation underlying this approach makes it a highly effective tool for numerous downstream applications, such as augmented reality (AR), object editing, and robotics. The project code and additional resources are available at <https://jojijoseph.github.io/3dgs-segmentation>.

## I. INTRODUCTION

3D Gaussian splatting (3DGS) is a popular technique for rendering novel viewpoints using 3D Gaussian distributions as rendering primitives[7]. It is particularly known for its speed and flexibility in rendering. Beyond color information, 3DGS can be extended to learn 3D feature fields, enabling applications in 2D and 3D segmentation and localization tasks[13], [19].

However, while feature fields perform well in tasks like localization and 2D segmentation [13], [19], they often encounter challenges with clean 3D segmentation, resulting in artifacts such as floaters—small, disconnected fragments that degrade the quality of the 3D representation. Furthermore, training these models is computationally intensive and slow, limiting their practical use.

Our approach addresses these challenges by using 2D segmentation masks generated by 2D segmentation models and applying inference-time gradient backpropagation. Inspired by 2D techniques for visualization, such as DeepDream[10] and Grad-CAM[15], our method identifies Gaussians responsible for specific 2D masks, enabling more accurate and efficient 3D segmentation.

Although we have utilized 2D segmentation models for mask generation in our experiments, it is also possible to leverage a feature field method like LangSplat [13] for mask

<sup>1</sup>Joji Joseph is with RBCCPS, Indian Institute of Science, Bangalore, India [jojijoseph@iisc.ac.in](mailto:jojijoseph@iisc.ac.in)

<sup>2</sup>Bharadwaj Amrutar is with RBCCPS and ARTPARK, Indian Institute of Science, Bangalore, India [amrutar@iisc.ac.in](mailto:amrutar@iisc.ac.in)

<sup>3</sup>Shalabh Bhatnagar is with RBCCPS and CSA, Indian Institute of Science, Bangalore, India [shalabh@iisc.ac.in](mailto:shalabh@iisc.ac.in)

generation. This would enable the conversion of feature-field-based methods, which are primarily suited for 2D segmentation, into effective tools for 3D segmentation.

In this paper, our key contributions are as follows.

- 1) We present a novel method for lifting segmentation results from 2D models to 3D Gaussian splats.
- 2) We incorporates 2D-to-3D affordance transfer, enhancing manipulation tasks in 3D environments.
- 3) We demonstrate the use of inference-time gradients for pruning trained Gaussians.

## II. RELATED WORKS

Neural Radiance Fields (NeRF) [9] is a state-of-the-art method for novel view synthesis, where a neural network is trained on sparse input images to generate photorealistic views from unseen viewpoints. NeRF learns an implicit volumetric representation of a scene by encoding color and density at each 3D point. However, because NeRF represents scenes implicitly, reconfiguring or modifying objects within the scene is challenging and often requires retraining the network.

In contrast, 3D Gaussian Splatting [7] offers an explicit 3D scene representation, where 3D Gaussians serve as the fundamental graphics primitives. Each Gaussian has associated attributes such as color, opacity, and orientation. Rearranging and editing objects in 3D Gaussian Splatting can be achieved by directly manipulating the Gaussians and their corresponding payloads. This explicit nature provides greater flexibility for interactive applications, such as object editing, augmented reality and real time reconfiguration.

A natural extension of radiance field rendering is feature field rendering, where additional feature embeddings are incorporated. Recent works like [19], [13], [16] have explored this idea by introducing feature fields with an additional embedding payload.

For instance, Feature-3DGS [19] focuses on training high-dimensional features, while LangSplat [13] emphasizes training compressed, low-dimensional features. Both approaches have shown excellent results for 2D segmentation of rendered outputs. However, they struggle with 3D object segmentation. As illustrated in Figure 1, we compare an object segmented using Feature-3DGS with our method.

Feature-3DGS performs 3D segmentation by matching language embeddings to the feature embeddings of the Gaussians. While effective for 2D segmentation, it often fails in 3D because the features of individual Gaussians do not fully correspond to the final rendered feature, which is a

weighted sum of contributions from multiple Gaussians (see Equation 1). This mismatch limits Feature-3DGS’s ability to reliably segment objects in 3D, whereas our method addresses this limitation by directly leveraging gradient information.

Another class of methods that segments objects from few visual prompts [3], [6]. Our approach is similar to that of SAGD Boundary-Enhanced Segmentation [6] in that both methods are training free and use multi-view labeling. Similar to SAGD, our approach is training-free and utilizes multi-view labeling. However, while SAGD uses a binary voting system, we assign influence-based voting to each Gaussian, which allows for more accurate segmentation in complex scenes.

Our method is based on the observation that the backpropagated color gradient of each Gaussian indicates its influence on the final rendering (see Section III). While inference-time backpropagation is often used for explainability [15] and special effects [10] in 2D models, gradient-based methods are generally less suited for clean segmentation. Fortunately, the underlying mathematical formulation of 3D Gaussian Splatting allows for clean segmentation using gradients.

In some applications, especially manipulation tasks, segmenting regions without clear boundaries, such as grasp affordance regions, is challenging, making annotation through language prompting difficult. In such cases, DINO [12] features can be employed to match and transfer affordance regions. These features are effective for identifying semantically similar regions and have been successfully applied in semantic segmentation[2], [1] and affordance transfer[5] without any fine-tuning. We utilize DINO-based affordance transfer to map annotated regions from example images to 3D Gaussians. Although the initial transferred affordances may lack alignment, distillation into Gaussian splats improves the result, making it suitable for manipulation tasks.

### III. METHODS

#### A. 3D Segmentation

3D Gaussian Splatting is a rendering technique that represents scenes using Gaussian distributions as primitives [7]. Each Gaussian is characterized by its mean position, which defines its center in 3D space, and its covariance, which controls its spatial extent and orientation. This covariance determines the Gaussian’s influence in different directions. Each Gaussian also has associated anisotropic color and opacity.

To render a scene, the Gaussians are depth-ordered relative to the camera, ensuring that nearer Gaussians are rendered on top of those farther away. The 3D Gaussian distributions are then projected onto a 2D plane using the Jacobian of the projection transformation [20]. This projection determines the size and shape of each Gaussian on the 2D image plane, where the span of each Gaussian represents the region it influences. The opacity of each Gaussian decreases exponentially from its center, resulting in a smooth blending effect. Once the 3D-to-2D transformation is complete, the contributions of all Gaussians are combined using alpha blending to produce the final rendered image (see equation 1).

Listing 1: Pseudocode of our method

```
def get_3d_mask(gaussians, viewpoints, masks):
    accumulated_grads = [0] * len(gaussians)

    for camera_params, mask_2d in zip(viewpoints, masks):

        # forward propagation in training mode
        frame = rasterize(gaussians, camera_params)

        # Gradient backpropagation
        accumulated_grads += mask_gradient(
            gaussians, frame, mask_2d)
        accumulated_grads -= mask_gradient(
            gaussians, frame, ~mask_2d)
    mask_3d = accumulated_grads > 0
    return mask_3d
```

Consider the color  $C$  of a pixel at  $(x,y)$  in a 3DGS rendering,

$$C = \sum_{n \leq N} c_n \alpha_n \prod_{m < n} (1 - \alpha_m) \quad (1)$$

$$= \sum_{n \leq N} c_n \alpha_n T_n \quad (2)$$

Where  $N$  is the total number of Gaussians, each indexed by its sorted position,  $c_n$  is the color associated with the  $n$ th Gaussian,  $\alpha_n$  is the opacity of the  $n$ th Gaussian at  $(x,y)$  adjusted with exponential falloff, and  $T_n = \prod_{m < n} (1 - \alpha_m)$  is the transmittance of  $n$ th Gaussian at  $(x,y)$ .

Taking the derivating with respect to color of  $k$ th Gaussian  $c_k$ ,

$$\frac{\partial C}{\partial c_k} = \alpha_k T_k \quad (3)$$

This derivative is zero only if either the transmittance  $T_k$  or opacity  $\alpha_k$  is zero, indicating that the Gaussian does not contribute to the final color. A non-zero gradient indicates that the Gaussian influences the pixel color.

Listing 2: Pseudocode of 2D-3D affordance transfer

```
def affordance_transfer(gaussians,
    labelled_examples):
    """
    labelled_examples are 2D images annotated with
    affordance regions.
    Unannotated regions are treated as background.
    """

    votes = zeros(m, n) # m in the number of labels,
    n is the number of gaussians
    example_features = concat([example.features for
        example in labelled_examples])
    for camera in get_cameras(): # get_cameras is a
        function that returns an iterator of
        camera params
        img = rasterize(gaussians, camera)
        features = get_dino_features(img)
        mask = get_init_mask(img)
        # 2d-2d transfer - can be vectorized
        for feature in features:
            mask[feature.region] = kNNSearch(
                feature, example_features)
    # 2d-3d transfer
    for label in labels: # Including background
        votes[label] += mask_gradient(img, mask
            == label)
    return argmax(votes, dim=0)
```



(a) Extraction - Feature 3DGS      (b) Extraction - Ours      (c) Deletion - Feature 3DGS      (d) Deletion - Ours

Fig. 1: Comparison of our approach with Feature-3DGS. The segmentation produced by Feature-3DGS is less clean than our method because individual Gaussians may not have features that are fully representative of the final rendered features.

Our approach leverages the gradients of the Gaussians that influence specific 2D segmentation regions. We maintain a buffer, sized according to the number of Gaussians, where gradients are accumulated based on the segmentation masks. Foreground gradients are added to the buffer, while background gradients (from the inverted mask) are subtracted to prevent misclassification (due to influence of background Gaussians near to the surface on foreground region). Once accumulated, the 3D Gaussians with positive gradients in the buffer are classified as part of the segmented region. The pseudocode for this process is provided in Listing 1.

#### B. 2D-3D affordance transfer

We first perform a 2D-2D affordance transfer from example images to rendered images. The transferred images serve as segmentation masks. Next, we apply gradient-based voting to the transferred images.

We annotate different affordance regions in example images and extract feature vectors from the output feature map of DINO[12] corresponding to those examples. With feature vectors for each affordance region (including the background), we classify each patch in the rendering using kNN with cosine similarity, where the example feature vectors act as the training set. Each patch is then assigned a label.

Finally, we apply the voting algorithm (shown in Listing 2). Even if the input affordance map contains inaccurate patches, the error is compensated after voting across multiple frames.

## IV. EXPERIMENTS

### A. Gradient Calculation

We utilized gsplat [18] as our 3D Gaussian rasterizer, where the color of each Gaussian was represented by spherical harmonics (SH) coefficients. Since the gradients were independent of color values, as demonstrated in Equation 3, we treated the DC components of the SH coefficients as analogous to color and calculated gradients with respect to these DC components.

Once the image was rendered, we multiplied the output by the mask element-wise, computed the mean, and backpropagated the gradients to the input color (represented by SH

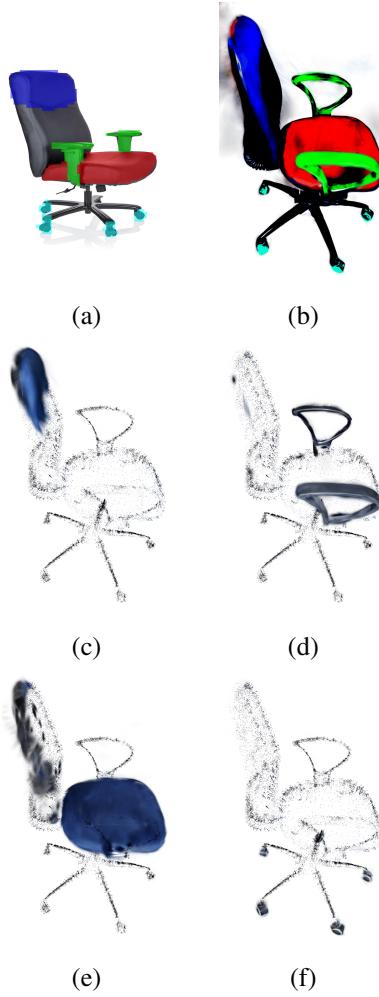


Fig. 2: (a) Example image with annotated affordances. (b) Segmentation map showing transferred affordances in a 3D scene. (c)-(f) Visualizations of different parts, with the remaining Gaussians represented as a point cloud for clarity, allowing better understanding of regions in relation to the whole scene.

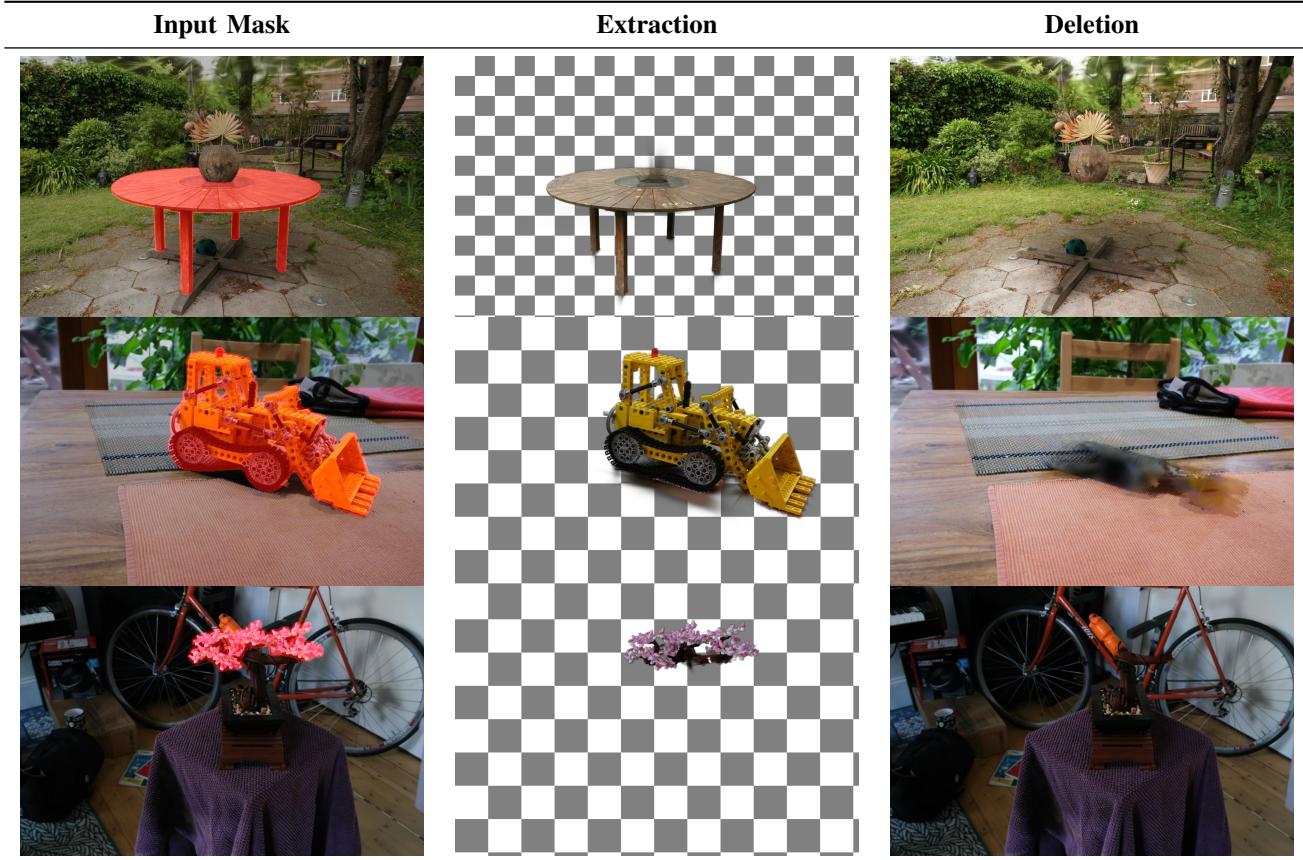


Fig. 3: The first column shows a rendered frame along with its corresponding input mask. In the second column, we present the results after extracting the 3D Gaussians that align with the generated 3D mask. The third column illustrates the rendered output from the remaining Gaussians. Notably, the background is visible instead of a blank space, as the segmentation occurs directly within the 3D space. For enhanced clarity, zooming in is recommended.

coefficients). All experiments were conducted on an NVIDIA A6000 GPU.

### B. 3D Segmentation

To identify the segmentation mask, we used YOLO-world [4] for the initial bounding box detection, followed by SAM 2 [14] to estimate and track masks throughout the rendered frames.

Some of the qualitative results were shown in Figure 3. We provided additional results on the project page.

For the quantitative comparison, we employed the MipNeRF 360 dataset. To ensure a fair, apple-to-apple comparison, we compared our method against two baseline voting methods.

**Baseline 1:** This method assigns a vote if the projected Gaussian falls within the input 2D mask. A limitation of this approach is that it still votes for Gaussians that are occluded.

**Baseline 2:** This method uses gradient magnitude to decide which Gaussians receive votes, but the voting magnitude is constant for all Gaussians. A drawback is that the foreground need not be fully opaque, even though it appears so. As a result, some background Gaussians may still receive votes.

Since we did not have access to the actual 3D volume, we repurposed the 2D mIoU metric for evaluation. We uniformly

sampled 10% of the masks for segmentation purposes, while using the remaining 90% for evaluation.

To produce the 2D mask from the estimated 3D mask, we assigned black to the background and white to the foreground. Then we thresholded the grayscale rendered output to obtain the estimated mask. We did not use segmented 3D region to produce 2D masks because the ground truth 2D masks were often partially occluded, while the 2D masks from extracted regions were unoccluded.

The results are shown in Table I. We report the mean Intersection over Union (mIoU) between the estimated masks and the ground truth masks. We chose not to report pixelwise accuracy because, due to the typically larger background regions, it tends to yield a high value that does not accurately reflect the quality of the segmentation.

As seen in Table I, our method outperforms both baselines across all scenes. In particular, the Bicycle scene posed a challenge due to the presence of two objects (a bicycle and a bench), both of which contain many thin structures and holes. These structures were difficult to segment accurately because the ground truth masks covered the holes, leading to a reduction in mIoU for this scene.

TABLE I: Table showing the comparison of mIoU between our method and baselines on scenes where we were able to track object using SAM 2. We used 10% of the images for segmentation purposes, while the remaining 90% were used for evaluation.

Scene	Baseline 1	Baseline 2	Ours
Bicycle	52.03	55.55	<b>59.46</b>
Bonsai	77.48	78.67	<b>81.44</b>
Garden	66.17	81.78	<b>95.23</b>
Kitchen	80.70	82.79	<b>92.89</b>
Room	93.79	89.60	<b>94.57</b>
Mean	74.03	77.68	<b>84.72</b>



Fig. 4: Examples of images used for affordance transfer with annotations. Note that these instances, though belonging to the same class, are different from the objects in the target frames.

### C. Affordance Transfer

For each scene, we annotated a few examples with affordance categories using LabelMe [17]. The images were generated using DALL-E and, although they belong to the same category, they are different from the objects in the 3D scene. Figures 4 shows few of the example images along with labels. We rendered each image and transferred the affordance regions from the example images to the rendering. The voting algorithm, using masked gradients, was applied as described in Listing 2.

For quantitative evaluation, we used the RGB-D Part Affordance Dataset [11]. We constructed the Gaussian splats using the provided cluttered scenes and then calculated the mIoU after the affordance transfer. The quantitative results are shown in Table II.

Since the affordance regions were estimated using 2D annotated examples rather than the ground truth, we used all manually annotated images (provided by the authors of [11]) for evaluation. Some images in the original dataset were automatically annotated, which resulted in significant differences in the ground truth. Therefore, we relied solely on manually annotated frames for a more consistent evaluation.



(a) 2D-2D Affordance Transfer (b) 2D-3D Affordance Transfer

Fig. 5: Results of 2D-2D and 2D-3D affordance transfer. The labels generated during the 2D-2D affordance transfer serve as input to the 2D-3D affordance transfer. Despite not having perfectly aligned labels, voting over multiple frames makes 2D-3D affordance transfer more precise.

TABLE II: Table showing comparison of 2D-2D and 2D-3D affordance transfer.

Scene	mIoU		Recall	
	2D-2D	2D-3D	2D-2D	2D-3D
1	40.09	<b>47.87</b>	61.37	<b>67.77</b>
2	45.92	<b>55.63</b>	69.80	<b>81.07</b>
3	49.48	<b>60.50</b>	73.46	<b>86.95</b>
Mean	45.16	<b>54.67</b>	68.21	<b>78.60</b>

Since the affordance transfer is conducted patch-wise, objects with narrow regions may result in nearby background regions receiving votes. To address this, we introduced recall—true positive pixels divided by the total number of pixels in the ground truth mask—to discard pixels that fall outside the ground truth mask, providing a clearer interpretation of the results.

Our hypothesis was that less accurate 2D-2D transfer could still lead to accurate 2D-3D transfer due to the large number of pooled inputs in the voting process. The results in Table II and Figure 5 confirm this hypothesis, demonstrating that pooling from multiple frames improves the accuracy of the 2D-to-3D affordance transfer.

### D. Pruning Gaussians with Inference-Time Backpropagation

For pruning, we considered the entire image as a mask and voted for each Gaussian based on its gradient magnitude. At the end of the process, we retained only the Gaussians with non-zero votes.

We show the results of applying our method to the MipNeRF 360 dataset in Table III.

The size of each scene was capped at 1,000,000 Gaussians, and the training process followed the MCMC strategy from [8]. We verified the maximum absolute pixel error across all viewpoints after pruning, which remained at zero, indicating that our method efficiently pruned Gaussians without compromising accuracy.

## V. CONCLUSION

We presented a method that utilizes inference-time back-propagation to segment 3D Gaussian splats and few shot

TABLE III: Each scene in the Mip-NeRF dataset and the percentage of Gaussians pruned.

Scene	Gaussians Removed (%)
bicycle	3.06
bonsai	11.92
garden	0.65
kitchen	3.97
room	21.49
counter	7.06
stump	1.40
Mean	7.08

affordacne transfer. With a minor modification, our approach was able to prune trained Gaussian splatting models without any loss in accuracy.

However, this method does have limitations, primarily its dependence on accurate 2D masks and chosen viewpoints. Moreover, the gradient calculations across multiple frames result in slower performance compared to feature-field-based methods.

Despite these drawbacks, the method’s high segmentation quality makes it highly suitable for downstream applications that require one-time or soft-real-time segmentation, such as digital twins, augmented reality, and asset generation.

## REFERENCES

- [1] S. Amir, Y. Gandelsman, S. Bagon, and T. Dekel. Deep vit features as dense visual descriptors. *arXiv preprint arXiv:2112.05814*, 2021.
- [2] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers, 2021.
- [3] J. Cen, J. Fang, C. Yang, L. Xie, X. Zhang, W. Shen, and Q. Tian. Segment any 3d gaussians. *arXiv preprint arXiv:2312.00860*, 2023.
- [4] T. Cheng, L. Song, Y. Ge, W. Liu, X. Wang, and Y. Shan. Yolo-world: Real-time open-vocabulary object detection. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [5] D. Hadjivelichkov, S. Zwane, M. P. Deisenroth, L. Agapito, and D. Kanoulas. One-shot transfer of affordance regions? affcorrs!, 2022.
- [6] X. Hu, Y. Wang, L. Fan, J. Fan, J. Peng, Z. Lei, Q. Li, and Z. Zhang. Sagd: Boundary-enhanced segment anything in 3d gaussian via gaussian decomposition, 2024.
- [7] 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.
- [8] S. Kheradmand, D. Rebain, G. Sharma, W. Sun, J. Tseng, H. Isack, A. Kar, A. Tagliasacchi, and K. M. Yi. 3d gaussian splatting as markov chain monte carlo, 2024.
- [9] B. Mildenhall, P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.
- [10] A. Mordvintsev, C. Olah, and M. Tyka.
- [11] A. Myers, C. L. Teo, C. Fermüller, and Y. Aloimonos. Affordance detection of tool parts from geometric features. In *ICRA*, 2015.
- [12] M. Oquab, T. Darisetty, T. Moutakanni, H. Vo, M. Szafraniec, V. Khaldidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. Dinov2: Learning robust visual features without supervision, 2024.
- [13] M. Qin, W. Li, J. Zhou, H. Wang, and H. Pfister. Langsplat: 3d language gaussian splatting. *arXiv preprint arXiv:2312.16084*, 2023.
- [14] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädele, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.
- [15] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359, Oct. 2019.
- [16] J.-C. Shi, M. Wang, H.-B. Duan, and S.-H. Guan. Language embedded 3d gaussians for open-vocabulary scene understanding. *arXiv preprint arXiv:2311.18482*, 2023.
- [17] K. Wada. Labelme: Image Polygonal Annotation with Python.
- [18] V. Ye and A. Kanazawa. Mathematical supplement for the `gsplat` library, 2023.
- [19] S. Zhou, H. Chang, S. Jiang, Z. Fan, Z. Zhu, D. Xu, P. Chari, S. You, Z. Wang, and A. 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.
- [20] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Surface splatting. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH ’01, page 371–378, New York, NY, USA, 2001. Association for Computing Machinery.