Fov-GS: Foveated 3D Gaussian Splatting for Dynamic Scenes

Runze Fan, Jian Wu, Xuehuai Shi, Lizhi Zhao, Qixiang Ma, and Lili Wang

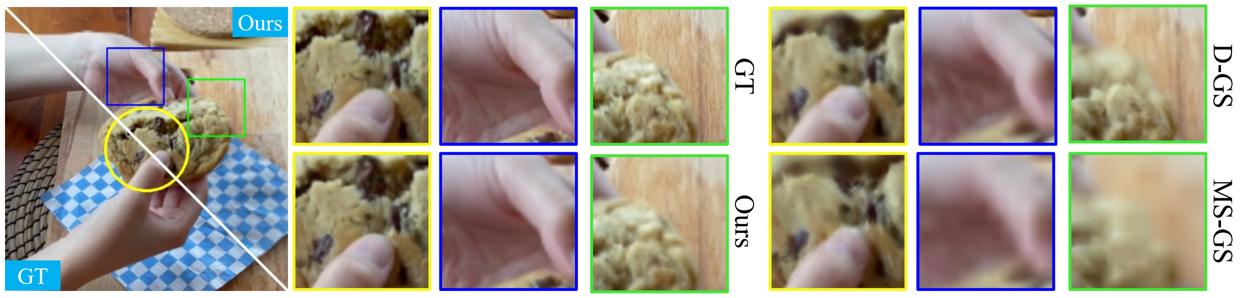

Fig. 1: Left: Comparison of our foveated 3D Gaussian splatting (upper-right) and the ground truth (GT, lower-left). Right: As illustrated in the close-ups of the images of both the foveal region (yellow) and periphery regions (blue and green), compared with D-GS [1] and MS-GS [2], our results are closer to GT, and preserve better visual details in foveal region and salient region.

Abstract—Rendering quality and performance greatly affect the user's immersion in VR experiences. 3D Gaussian Splatting-based methods can achieve photo-realistic rendering with speeds of over 100 fps in static scenes, but the speed drops below 10 fps in monocular dynamic scenes. Foveated rendering provides a possible solution to accelerate rendering without compromising visual perceptual quality. However, 3DGS and foveated rendering are not compatible. In this paper, we propose Fov-GS, a foveated 3D Gaussian splatting method for rendering dynamic scenes in real time. We introduce a 3D Gaussian forest representation that represents the scene as a forest. To construct the 3D Gaussian forest, we propose a 3D Gaussian forest initialization method based on dynamic-static separation. Subsequently, we propose a 3D Gaussian forest optimization method based on deformation field and Gaussian decomposition to optimize the forest and deformation field. To achieve real-time dynamic scene rendering, we present a 3D Gaussian forest rendering method based on HVS models. Experiments demonstrate that our method not only achieves higher rendering quality in the foveal and salient regions compared to the SOTA methods but also dramatically improves rendering performance, achieving up to $11.33X$ speedup. We also conducted a user study, and the results prove that the perceptual quality of our method has a high visual similarity with the ground truth.

Index Terms—3D Gaussian, Dynamic Scene, Foveated Rendering, Dynamic-Static Separation, HVS models.

1 Introduction

Virtual Reality (VR) is being increasingly utilized across diverse fields, including entertainment, culture, and manufacturing. User immersion in VR experiences is mainly influenced by two factors: rendering quality and speed. 3D Gaussian Splatting (3DGS) [3] achieves photo-realistic rendering with speeds of over 100 fps for static scenes. D-GS [1] extends static 3DGS to monocular dynamic scenes, but the rendering performance is inefficient, with frame rates dropping below 10 fps in complex scenes with millions of Gaussians.

One possible solution to improve rendering performance is to reduce the number of Gaussians needed for deformation and rendering. Simply ignoring the Gaussians can cause missing parts, as the original 3D Gaussian representation doesn't encode the signal of different frequencies at different Gaussians[2]. Foveated rendering generates images of varying quality for different visual perception regions based on the characteristics of the human visual system (HVS), providing an idea for reducing the number of Gaussians. While foveated rendering can not be adapted to the 3DGS pipeline directly. This is because foveated rendering is processed in 2D screen space, whereas Gaussians are distributed in 3D space. Moreover, for dynamic scenes, D-GS predicts deformations for all Gaussians in the scene, which is often unnecessary, as most objects in real scenes are static and only a few are dynamic. Thus, to adapt foveated rendering to the 3DGS pipeline and achieve photo-realistic and real-time rendering for dynamic scenes in VR applications, two challenges must be addressed: 1) Develop a new Gaussian-based scene representation that supports foveated rendering, enabling efficient dynamic scene representation and rendering; 2) Determine how to perform rendering based on this representation in a way that aligns with the visual perception of the HVS.

In this paper, we propose Fov-GS, a foveated 3D Gaussian splatting method for rendering the dynamic scenes in real time. Our method takes the monocular videos as inputs, and synthesizes VR binocular real-time foveated images. We introduce a 3D Gaussian forest to represent the scene. To achieve foveated rendering, each tree within this forest has the same layers, with layers determined by the HVS models. Different layers correspond to different levels of detail (LOD) for the scene. To further accelerate rendering, trees are categorized into static and dynamic trees, and only dynamic trees need to be deformed. To construct the 3D Gaussian forest, we introduce a 3D Gaussian forest initialization method based on dynamic-static separation. This method first distinguishes between dynamic and static Gaussians, and then constructs the dynamic and static forest separately. Subsequently, we propose a 3D Gaussian forest optimization method based on deformation field and Gaussian decomposition. This method optimizes the deformation field, dynamic forest, and static forest separately. Next, we present a 3D Gaussian forest rendering method based on HVS models to achieve real-time rendering. This method first selects and de-

- *Lili Wang is with State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China; Peng Cheng Laboratory, Shengzhen, China. Lili Wang is the corresponding author. E-mail: wanglily@buaa.edu.cn.*
- *Runze Fan, Jian Wu, Lizhi Zhao, and Qixiang Ma is with State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China. E-mail: by2106131@buaa.edu.cn, lanayawj@buaa.edu.cn, lizhizhao@buaa.edu.cn, sycamore_ma@outlook.com.*
- *Xuehuai Shi is with Nanjing University of Posts and Telecommunications. E-mail: xuehuai@njupt.edu.cn.*

Manuscript received 17 September 2024; revised 20 December 2024; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

forms the dynamic forest based on the acuity model and then selects and renders dynamic and static forests based on the contrast sensitivity function (CSF) model.

Compared to the state-of-the-art (SOTA) methods, our method achieves higher PSNR and smaller LPIPS in foveal and salient regions. Our method dramatically improves rendering performance with higher visual perceptual quality, achieving up to $11.33X$ speedup. We also conducted a user study, which proves that the perceptual quality of our method has a high visual similarity with the ground truth. Fig. 1 shows the comparison between the results of our method and those of the SOTA methods. The close-ups illustrate the rendering details in the foveal and periphery regions of our method compared to the ground truth (GT), the D-GS [1], and the MS-GS methods [2].

In summary, the contributions of this paper are as follows:

- a foveated 3D Gaussian splatting pipeline for rendering dynamic scenes in real-time. To the best of our knowledge, this is the first method to adapt foveated rendering to 3DGS pipeline.
- a 3D Gaussian forest representation for dynamic scenes, a dynamic-static separation-based initialization method, and a deformation field and Gaussian decomposition-based optimization method.
- a 3D Gaussian forest rendering method based on HVS models, which selects, deforms, and renders Gaussian forest based on the acuity model and contrast sensitivity function model.

2 Releated Work

In this section, we first give a brief review of the research on 3DGS, and then discuss the methods for foveated rendering. For a more comprehensive review of 3DGS and foveated rendering, we recommend the readers refer to the reviews [4, 5].

2.1 3D Gaussian Splatting.

NeRF-based methods has achieved great success in static and dynamic scenes reconstruction[6, 7, 8], but with low rendering performance. Recent 3D Gaussian Splatting [9] has become a prominent technique in computer graphics, offering an efficient method for rendering complex scenes with photo-realistic results. 3DGS begins by initializing 3D Gaussians with SfM points, and then the Gaussians are rendered using differentiable rasterization. Although 3DGS has achieved good results in terms of rendering performance and quality, there is still room for further improvements, including dynamic scenes, multi-scale images, anti-aliasing, and semantic segmentation.

3DGS can't be used directly for dynamic scenes. One way to extend 3DGS to dynamic scenes is to learn deformations instead of modeling the scene at every time step. Research in this area follows two main approaches: one focuses on constructing a deformation field for all Gaussians [1, 10, 11], while the other constructs the field by pairing a small number of Gaussians to control the rest [12].

3DGS will result in strong artifacts when changing the sampling rate. Yu et al. [13] proposed Mip-Splatting, which introduced a 3D smoothing filter and 2D Mip filter to eliminate high-frequency artifacts and mitigate aliasing and dilation issues. Yan et al. [2] proposed a multi-scale Gaussian representation (MS-GS), which aggregates smaller Gaussians into larger ones. Kerbl et al. [14] proposed a hierarchical Gaussian representation (H-GS), which also aggregates smaller Gaussians into larger ones but maintains the relationships between them in a multi-layered binary tree structure. This allows H-GS to efficiently select appropriate Gaussians by searching the tree instead of iterating through all Gaussians, thus improving rendering efficiency. Although this improves Gaussian selection efficiency, ideally allowing it to be done in linear time, the time required increases significantly for lower-resolution images due to the imbalance in the binary tree structure. An alternative is to represent the scene as a forest with a fixed number of layers per tree. Moreover, this strategy of aggregating Gaussians is only suitable for static scenes and not for dynamic scenes, where the relative positions of Gaussians change over time.

3DGS is also extended to scene understanding and semantic segment. Zhou et al. [15] proposed Feature-3DGS, which learns semantic feature for each Gaussian and achieves semantic segmentation in novel view. Ye et al. [16] went a step further and proposed GS-group, which learns identity encoding for each Gaussian and achieves scene editing. However, these methods are only suitable to static scenes.

In this paper, we propose a 3D Gaussian forest representation for dynamic scenes. In previous methods, Gaussians correspond to the same LOD belonging to different scales or hierarchies of Gaussian. While in our representation, Gaussians at the same layer across different trees correspond to the same LOD.

2.2 Foveated Rendering.

Early research in foveated 3D rendering focused on ray tracing-based methods [17, 18, 19, 20, 21, 22]. To enhance rendering efficiency, researchers focus on rasterization-based foveated rendering techniques. Guenter et al. [23] rendered three image layers around the gaze point with discrete sampling rates. Stengel et al. [24] expanded the fovea region linearly based on the gaze motion vector. Turner et al. [25] achieved foveated rendering by aligning multiple low-resolution and one high-resolution renderings of the periphery region. Besides, multiple mapping-based foveated rendering methods have been proposed [26, 27, 28, 29, 30, 31, 32, 33]. Blurred rendering results in the periphery region, especially those containing visual features, can lead to a perceived decrease in visual quality. Many methods have been proposed to enhance the visual perceptual quality in the periphery region [24, 34, 35, 36].

With the advancement of deep learning techniques, researchers have applied these methods to improve the rendering quality and performance [37, 38, 39, 40, 41], and predict the gaze movement [42, 43]. Deng et al. [38] introduced FoV-NeRF, which is the first to combine foveated rendering and NeRF. FoV-NeRF used multiple MLPs to synthesize images for foveal, peripheral, and far-peripheral regions. The foveal region is rendered with the highest quality, while the peripheral and far-peripheral regions with lower quality, and then Fov-NeRF fused them to generate the final foveated images. Bauer et al. [39] developed FovolNet, a fast-foveated deep neural network that improves peripheral visual quality through an efficient reconstruction network.

In this paper, we adapt the foveated rendering to the 3DGS pipeline to achieve real-time rendering for dynamic scenes.

3 Method

We introduce a 3D Gaussian forest to represent the dynamic scene (Sec 3.1). Based on this representation, we propose Fov-GS, a scene-aware foveated 3D Gaussian splatting method. Figure 2 shows the pipeline of our method. In this pipeline, we first initialize the 3D Gaussian forest based on dynamic-static separation (Sec 3.2). Then, we optimize the 3D Gaussian forest based on deformation field and Gaussian decomposition (Sec 3.3). At last, we render the 3D Gaussian forest to a foveated image based on the HVS models (Sec .3.4).

3.1 3D Gaussian Forest Representation

The main idea of foveated rendering is to render different resolutions' images for different regions [44]. However, due to the nature of 3DGS, directly using the original representation to render multi-resolution images may result in significant artifacts and reduce rendering performance [2]. To address multi-resolution rendering, various Gaussian representations have been proposed [2, 13, 14], but these methods are only suitable for static scenes, and there is no one-to-one correspondence between LOD and the scales or hierarchies of Gaussians. In dynamic scenes, the computational cost of deforming Gaussian during rendering is much higher than the rendering itself.

To adapt foveated rendering to 3DGS for real-time rendering in dynamic scenes, we propose a 3D Gaussian forest representation, which uses a tree instead of a Gaussian as the smallest primitive, thus realizing fast localization of Gaussian at different layers and the separation of dynamic and static objects.

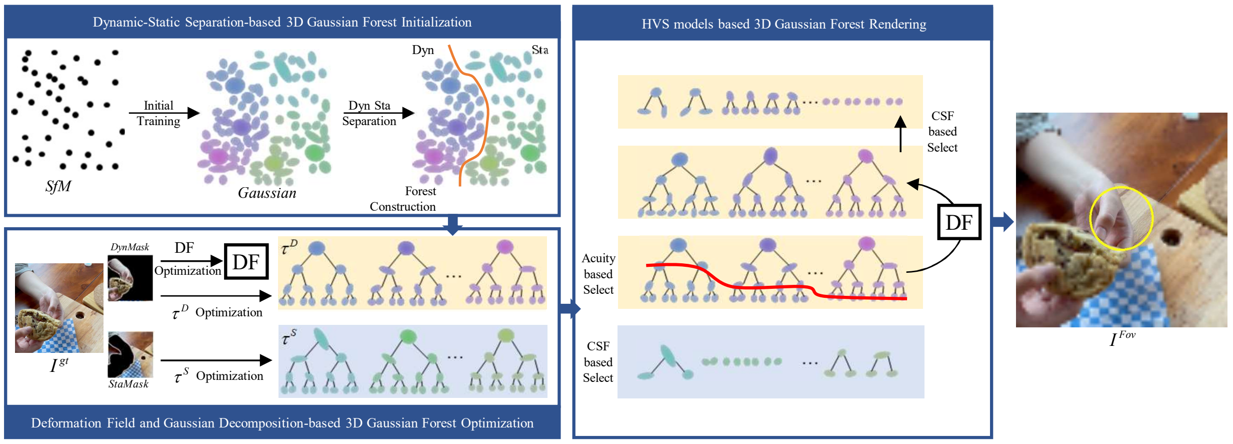

Fig. 2: The pipeline of our proposed Fov-GS method.

As shown in Fig. 2, each scene can be represented as a forest ψ, and a forest contains multiple trees τ. Each tree is a binary tree containing multiple layers of nodes v_l, and each node is a 3D Gaussian g. Each tree has the same number of layers, which are determined by the HVS models. Gaussians on the same tree are not independent, and they represent one primitive in the scene with different LOD. Gaussians at the same layer across different trees correspond to the same LOD. There are two kinds of trees in the forest, which are dynamic trees τ^D and static trees τ^S. Dynamic trees correspond to dynamic objects, with all contained Gaussians being dynamic g^D. Static trees correspond to static objects, with all contained Gaussians being static g^S. For each Gaussian, in addition to the traditional properties: Gaussian center position x, quaternion r, scaling s, color c, opacity σ, we add 2 extra properties: dynamic property Dyn and semantic property Se. Similar to D-GS [1], we adapt a deformation field DF to deform the dynamic Gaussian from the canonical space to space at a specific time. The deformation field is modeled by an MLP. It takes the position x of dynamic 3D Gaussian in canonical space and time t as input and outputs the offsets $\delta x, \delta r, \delta s$ of dynamic 3D Gaussians.

3.2 3D Gaussian Forest Initialization

We propose a dynamic-static separation-based 3D Gaussian forest initialization method with 3 sub-steps.

In the first sub-step, we train initial Gaussians. Similar to D-GS [1], we first train initial static Gaussians $G = \{g\}$ in canonical space, with iterations i_{max}. Then, we initialize the dynamic property Dyn of G as $False$, where $False$ denotes static, and set the semantic property Se to -1, where -1 denotes background.

In the second sub-step, we separate the dynamic and static Gaussians by iterating through all the training views in chronological order to update Dyn and Se properties. Existing 3DGS-based methods for dynamic scenes do not distinguish between dynamic and static objects, applying the deformation field universally. While straightforward, these methods are inefficient, as deforming all these Gaussians could be time-consuming. Most real-world scenes have only a few dynamic objects, with the majority being static, thus we separate the dynamic and static Gaussians to improve rendering performance and reconstruction quality. Given a training view, the corresponding ground-truth image I^{gt}, corresponding semantic segmentation Seg [45], and G, we first renders the image I^{pred} under this view. The optical flow Of is then calculated based on I^{gt} and I^{pred}, and the Dyn property is updated by calculating the ratio of the total optical flow intensity to the number of pixels in each segmentation instance. If this ratio exceeds a pre-defined threshold ε_{DS}, this segmentation instance is considered dynamic under this view. The Dyn of the Gaussians contributing more than a pre-defined threshold ε_{con} to the color of dynamic instance pixels is set to $True$, which denotes dynamic. The contribution con of a Gaussian to the pixel color is calculated by Eq.1.

$$\begin{cases} con = \dfrac{\alpha_m \prod\limits_{n=1}^{m}(1-\alpha_n)}{\sum\limits_{m=1}^{M} \alpha_m \prod\limits_{n=1}^{m}(1-\alpha_n)} \\ \alpha_m = \sigma_m \cdot e^{-\frac{1}{2}(p-\mu_m)^T \Sigma_m^{'}(p-\mu_m)} \end{cases} \tag{1}$$

where α_m is transmittance, μ_m is the center of the projected 2D Gaussian, $\sum_m^{'}$ is the 2D covariance matrix, p is the pixel coordinate, and $m = 1, \ldots, M$ represents the Gaussians covering this pixel, sorted by depth. The Se property is updated similarly. Only Gaussians with con greater than ε_{con} have their Se property set to $Seg(p)$, where $Seg(p)$ denotes the mask ID for p. After Dyn and Se have been updated, the dynamic Gaussians G^D and static Gaussians G^S are separated according to their Dyn property.

In the third sub-step, we initialize the forest according to the separated dynamic and static Gaussians with Algorithm. 1. This algorithm takes dynamic Gaussians G^D, static Gaussians G^S, pixel size s_L, and total layer number L as inputs, and outputs the initialized 3D Gaussian forest ψ_0.

As shown in Fig.3 (a) and (b), we first layer the Gaussians, with each layer corresponding to a LOD (Line 1). $v_{l=1,\cdots,L}$ denotes to the l-th Gaussian layer. The Layers function has three steps: 1) Create the threshold intervals with the upper and lower thresholds for each layer; 2) Calculate the size of each Gaussian; 3) Compare each Gaussian's size with the layer thresholds and assign it to the appropriate layer. The upper and lower thresholds for each layer are determined as follows. According to the Nyquist-Shannon Sampling Theorem [46], the sampling rate must be at least twice the highest frequency present in the signal. G is trained at the original resolution of $H \times W$ with corresponding pixel size s_L, the highest frequency in G is $\frac{1}{2s_L}$. For a level of detail with resolution $\frac{H}{2^{L-l}} \times \frac{W}{2^{L-l}}$, the pixel size becomes $2^{L-l} \cdot s_L$ with the field of view (FOV) unchanged, and the maximum frequency that can be sampled under this resolution without loss is $\frac{1}{2 \cdot 2^{L-l} \cdot s_L}$. Taking the camera model into account, the size s' of the projected 2D Gaussian can be approximated by Eq. 2.

$$s^{'} = f \cdot \frac{s}{d} \tag{2}$$

where f is the focal of the camera, s is the 3D size of Gaussian, d is the depth from the camera to the Gaussian. Thus, Gaussians larger than $2^{L-l+1} \cdot s_L \cdot d/f$ can be sampled losslessly under this resolution. Based on the above analysis, G^D and G^S are divided into L layers $\{\bigcup\limits_{l=1}^{L} v_l^{D/S}\}$. The L-th layer corresponds to the LOD with the highest quality. Gaussian with 3D sizes between $2^{L-l+1} \cdot s_L \cdot d/f$ and $2^{L-l+2} \cdot s_L \cdot d/f$ are placed in the l-th layer (v_l). Gaussian with 3D sizes smaller

Algorithm 1 Dynamic-Static Separation-based Forest Initialization

```
Input: dynamic Gaussians $G^D$, static Gaussians $G^S$, pixel size $s_L$, and layer number $L$.
Output: initialized 3D Gaussian forest $\psi_0\left(\bigcup_{i=1}^{N^D}\tau_i^D, \bigcup_{j=1}^{N^S}\tau_j^S\right)$.
 1: $\{\bigcup_{l=1}^{L} v_l^{D/S}\} \leftarrow$ Layers($G^{D/S}, L, s_L$)
 2: for $l = L, \dots, 2$ do
 3:     for $g_l^{D/S} \in v_l$ do
 4:         $g_{l-1}^{D/S} \leftarrow$Nearest($g_l^{D/S}, v_{l-1}^{D/S}$)
 5:         if Same($Se$) and NumChild($g_{l-1}^{D/S}$)$< 2$ then
 6:             $g_l^{D/S} \leftarrow$ LeftOrRightChild($g_{l-1}^{D/S}$)
 7:         else
 8:             $g' \leftarrow$ Twice($g_l^{D/S}$)
 9:             $v_{l-1}^{D/S} \leftarrow$ AddNode($g'$)
10:             $g_l^{D/S} \leftarrow$ LeftChild($g'$)
11:         end if
12:     end for
13:     for $g_{i/j}^{D/S} \in v_1$ do
14:         $v_{i/j,1}^{D/S} = g_{i/j}^{D/S}$
15:         for $l = 1, \dots, L-1$ do
16:             $v_{i/j,l+1}^{D/S} = LeftRightChild(v_{i/j,l}^{D/S})$
17:         end for
18:     end for
19: end for
20: $\tau_{i/j}^{D/S} \leftarrow \{\bigcup_{l=1}^{L} v_{i/j,l}^{D/S}\}, i = 1, \dots, N^D, j = 1, \dots, N^S$
21: $\psi_0 \leftarrow \left(\bigcup_{i=i}^{N^D}\tau_i^D, \bigcup_{j=1}^{N^S}\tau_j^S\right)$
```

than $2 \cdot s_L \cdot d/f$ are placed in the L-th layer (v_L), and Gaussian with 3D sizes bigger than $2^L \cdot s_L \cdot f \cdot d$ are placed in the 1-th layer (v_1). The 3D size of a Gaussian is defined as the length of the longest axis of the 3D ellipsoid. The f and d are approximated using the average focal and depth in training views.

Next, as shown in Fig.3 (c), we construct the relationship between Gaussians of different layers (Lines 2-12). The tree is constructed from the bottom up, for all dynamic or static Gaussians $g_l^{D/S}$ in layer l (Lines 2-3), we find the closest dynamic or static Gaussian $g_{l-1}^{D/S}$ in layer $l-1$ (Line 4). If $g_{l-1}^{D/S}$ has the same Se as $g_l^{D/S}$ and $g_{l-1}^{D/S}$ has fewer than 2 child nodes (Line 5), we set $g_l^{D/S}$ as the child node of $g_{l-1}^{D/S}$ (Line 6). If $g_{l-1}^{D/S}$ already has one child node, we determine the size of $g_l^{D/S}$ and the existing child node, the smaller one is the left child node, and the larger one is the right child node. Otherwise(Line 7), we create a new Gaussian g' whose size is twice that of $g_l^{D/S}$ (Line 8), and place g' to layer $l-1$ (Line 9), and $g_l^{D/S}$ is set as the left child of g' (Line 10). Finally, we iterate through layer 1 (Line 13), and set each Gaussian in layer one as the root node of each binary tree (Line 14). Then, we iterate through all layers (Line 15), and the child nodes are determined as the left and right child Gaussians (Line 16). The tree τ is formed from sets of layered Gaussians (Line 20). N^D and N^S are the numbers of dynamic and static Gaussians in the layer 1. The forest ψ_0 is then built from these trees (Line 21).

3.3 3D Gaussian Forest Optimization

D-GS optimized 3D Gaussian in canonical space and the deformable field jointly with a stop-gradient operation on x in the deformable field, which causes the actual gradient propagated to x differ from the theoretical one, and leads to suboptimal optimization of x and the deformable field. Besides, D-GS treats all Gaussians as dynamic, which introduces inaccuracies in the deformation of the dynamic Gaussians due to the influence of nearby static Gaussians.

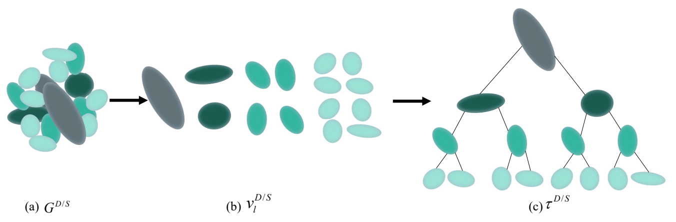

Fig. 3: Given a set of dynamic or static Gaussian $G^{D/S}$ with the same se, Algorithm 1 first divides the Gaussian into L layers, $v^{D/S}$. It then constructs the relationships between $v^{D/S}$ and generates the Gaussian trees $\tau^{D/S}$.

We propose a deformation field and Gaussian decomposition-based optimization method to address these problems. During optimization, we select time t and view $View$ in chronological order from the training set, and randomly select layer l from $[1, \cdots, L]$, for each iteration. The optimization is divided into dynamic forest optimization and static forest optimization.

The dynamic forest optimization contains 2 steps: deformation field optimization and dynamic trees optimization. For the optimization of deformation field DF, we first compute the deformed Gaussian g'_t for each Gaussian $g \in v_l^D$ of l-th layer of dynamic trees at time t with Eq.3.

$$\begin{aligned} g_{t'} &= g^*_{t_{pre}} + DF(x,t) - DF(x,t_{pre}) \\ &\approx g'_{t_{pre}} + DF(x,t) - DF(x,t_{pre}) \end{aligned} \tag{3}$$

where x is the position of g in canonical space, t_{pre} is the time of the previous iteration, and $g^*_{t_{pre}}$ is the optimal Gaussian at time t_{pre}. Due to $g'_{t_{pre}}$ has been optimized in previous iteration, we approximate $g^*_{t_{pre}}$ with $g'_{t_{pre}}$. Next, we render the image I' based on deformed Gaussian at l-th layer $g_t{'} \in v_l^{D'}$. Since DF should only model the deformation of dynamic objects, a masking operation is performed on ground truth full-resolution image I^{gt} based on dynamic mask $DynMask$, i.e., the non-dynamic region of I^{gt} is turned to black. $DynMask$ is generated in the same way as in Sec 3.2. Then the loss is calculated with Eq.4 based on the masked ground truth image I^{mgt} and I', and DF is optimized based on this loss.

$$L = (1-\lambda)L_1(I^{mgt}, I') + \lambda L_{D-SSIM}(I^{mgt}, I') \tag{4}$$

where L_1 denotes the $L1$ loss and L_{D-SSIM} denotes the D-SSIM term. For the optimization of dynamic trees $\tau^D(\bigcup_{l=1}^{L} v_l^D)$, we first compute the deformed dynamic Gaussians $g_t{''}$ at time t using optimized DF based on Eq. 5.

$$g''_t = g + DF(x,t)) \tag{5}$$

Then, we render the image I'' based on deformed Gaussian $g_t{''} \in v_l^{D''}$, and a new $DynMask''$ is generated. The masking operation and loss computation are the same as in the first step, while only v_l^D is optimized in this step.

For the optimization of static forest which is composed with the static trees $\tau^S(\bigcup_{l=1}^{L} v_l^S)$, we first render the image I''' through the Gaussians v_l^S of the l-th layer of the static trees, and the static mask $StaMask$ is generated by inverting $DynMask''$. Different from the optimization of the dynamic forest, the masking operation is not performed on the ground truth image, while the loss is computed only for the region corresponding to $StaMask$. This is because the dynamic region of I^{gt} is occupied by the dynamic objects in the foreground, and the static objects are blocked. The static Gaussians are optimized only when they are not blocked, and $StaMask$ helps achieve this. Then, v_l^S is optimized based on the masked loss.

Similar to 3DGS [3], the Adaptive Density Control (ADC) operation is executed every 100 iterations. The pruning and densifying processes are executed only for the Gaussian at the layer corresponding to this iteration. Besides, we also update the dynamic-static separation and the structure of the forest every 100 iterations. We first deform the dynamic Gaussians and render the image at the time, view, and layer corresponding to this iteration. Then, we compute the optical flow and optimize the *Dyn* and *Se* properties in the same way as in Sec 3.2. Next, the forest is updated in a similar way as in Algorithm 1. We only update the parent and children nodes for the Gaussian at the layer corresponding to this iteration. For the pruning process, we re-search the parent Gaussian for the child Gaussian of the pruned Gaussian. For the densifying process, we re-search the parent Gaussian for the newly added Gaussian. When searching for a parent node, we aim to preserve the original parent-child relationships. If the center of the child Gaussian lies within the ellipsoid defined by the parent Gaussian and both parent and child Gaussians have the same *Dyn* and *Se*, the existing parent-child relationship is retained. Otherwise, a new parent is searched for the child Gaussian.

3.4 HVS Models based 3D Gaussian Forest Rendering

Foveated rendering speeds up rendering while maintaining visual perceptual quality by adjusting rendering quality for each image position based on the characteristics of HVS. For a giving rendering quality, existing methods [2, 14] require complex selection strategies to select the appropriate Gaussians due to there is no one-to-one correspondence between LOD and the scales or hierarchies of Gaussian, and a more complex selection process is needed for these methods to achieve foveated rendering with high quality. While, in our 3D Gaussian forest, Gaussians at the same layer across different trees correspond to the same LOD, which paves the way for foveated rendering.

We propose a HVS models based 3D Gaussian forest rendering method. The core of our method is selecting appropriate layers of Gaussian for each tree according to the acuity model and the CSF model during rendering. For trees in dynamic forest, we first select Gaussian layer to be deformed based on the acuity model and deform them, and then select Gaussian layer to render based on the CSF model. For trees in static forest, we directly select Gaussian layer to render based on the CSF model.

Acuity model based dynamic forest selection and deformation. Compared with D-GS, our forest representation distinguishes between dynamic and static Gaussians, so that only the dynamic Gaussians need to be deformed before rendering, which improves rendering efficiency. To further improve rendering speed, we aim to deform only those Gaussians that are perceived by the user. A contradiction is that we can only determine which Gaussians are perceived after they have been deformed. Since the gaze point changes much faster than the deformation of the dynamic objects, we propose a two-phase strategy to resolve this contradiction. In the first phase, for each dynamic Gaussian tree $\tau \in \left\{\tau^D\right\}$, we first calculate the acuity a corresponding to it using Eq. 6 based on the acuity model.

$$a = w_0 + me(P, \bar{\mu}) \tag{6}$$

where w_0 is the acuity limit, m is the acuity slope, P is the gaze point, e is a function to calculate the eccentricity, $\bar{\mu}$ is the mean projected 2D position, calculated by projecting the 3D positions of all the dynamic Gaussians in τ from the previous frame into the current frame's view. Then, we select the layer l for each τ with appropriate LOD. If $\bar{\mu}$ is in the foveal region ($e < e_0$), the L-th layer is selected to have the best visual perceptual quality. If $\bar{\mu}$ is in the periphery region, the l-th layer is selected to have the same visual perceptual quality as the foveal region, and l is calculated with Eq. 7.

$$l = L - \left\lceil log_2\left(\frac{a}{w_0 + me_0}\right)\right\rceil \tag{7}$$

where $\lceil\rceil$ denotes upward rounding. Next, we calculate the deformation δ_l for the Gaussians in the selected layer with time t, and δ_l is applied to adjust the properties of all Gaussians throughout the tree. In the second phase, we re-select the layer for deformation to address changes of the region caused by the deformation in the first phase. Acuity and selected layer are re-calculated in the same way as in the first phase. If the layers selected in the first and second phases are not the same, the deformation is re-computed and the tree τ is re-deformed. In most cases, the layers selected in two phases are equal since the deformation is less compared to the change in the gaze point. Compared to deforming all Gaussians in a tree, this two-phase strategy reduces the number of Gaussians that need to be deformed.

CSF model-based dynamic and static forest selection. The acuity model gives a roughly linear relationship between visual perception and eccentricity. However, visual perception is influenced by multiple factors, such as visual features. Fan et al. [47] proposed a visual perceptual approach to combine acuity and visual features based on the CSF model. Since we aim for an image whose visual perceptual quality aligns with the capabilities of the HVS, we try to render an image with smooth transitions in visual quality, rather than one with abrupt changes or layering. Instead of rendering different resolutions' images for each layer [23, 38], we render the image at a uniform resolution and use a 2D filter to control the visual perceptual quality. First, we render a coarse image, perform saliency detection on this image to detect visual features, and obtain the saliency map *Sal*. This image is rendered using only the first layer of trees to quickly obtain a rough saliency detection result, which guides the subsequent selection. Then, for each tree τ in the dynamic and static forest, we select the appropriate layer and construct filters. For each τ, the projected 2D mean position is first computed, and the saliency $sal \in [0, 1]$ for the tree is obtained as the value of *Sal* corresponding to the pixel at the projected 2D mean position. The acuity a and the layer l are then calculated based on Eq.6 and Eq.7. If $l = L$, then only Gaussians at layer L are selected to render. Otherwise, Gaussians at layers l and $l+1$ are selected to synthesize images with smooth transitions. Gaussians in layer l can be perceived perfectly, while Gaussians in layer $l+1$ can't. Inspired by [13], we design 2D Gaussian low-pass filters F_{low} to filter the high-frequency component of Gaussians in layer $l+1$, which are designed using Eq.8.

$$\begin{cases} g(x)_{filted} = (g \otimes F_{low})(x) \\ g(x)_{filted} = \sqrt{\frac{\Sigma'}{\Sigma' + \frac{1}{f}\mathbf{I}}}\, e^{-\frac{1}{2}(x-\mu)^T\left(\Sigma' + \frac{1}{f}\mathbf{I}\right)^{-1}(x-\mu)} \end{cases} \tag{8}$$

where μ and Σ' are the projected 2D center and covariance matrix of Gaussian, and f is the maximum spatial frequency that can be perceived. f is calculated using Eq.9 based on the CSF model according to [47].

$$f = \frac{1 + k \cdot sal}{a} \tag{9}$$

where k is a pre-defined weight of sal, which indicates the degree to which salient regions attract human visual attention and thus influence human visual perception.

Finally, the foveated image I^{Fov} is rendered by simultaneous rendering the selected dynamic and static Gaussians and applying the 2D filters.

4 Experiment

4.1 Implementation

Datasets. We evaluated the quality and performance of our method for foveated image synthesis on Hyper-NeRF [8] and NeRF-DS [48] datasets. The Hyper-NeRF dataset is a monocular real-world dataset comprising 15 complex real-world physical scenes captured using handheld cameras, with a resolution of 1920×1072. Consistent with D-GS, we selected 3 real-world scenes for a detailed comparison, which are cookie, americano, and torchocolate. The NeRF-DS dataset is a monocular real-world dataset comprised of 8 complex real-world physical scenes captured using handheld cameras, with a resolution of 480×270. We selected 3 real-world scenes for detailed comparison, which are basin, plate, and cup. For a fair comparison, we conducted experiments on these two datasets with the same training and testing split as [1].

Implementation Details. We implemented our method on top of the original 3DGS [3]implementation in C++ and PyTorch, the deformable field model DF is the same as in D-GS [1]. We conducted training for a total of 40k iterations. We first performed the initial training used in Sec 3.2 for $i_{max} = 3k$ iterations. Subsequently, we optimized the 3D Gaussian forest and the deformation field, and the hyper-parameters are the same as in D-GS. The ADC operation is executed every 100 iterations. The dynamic-static separation and the structure of the forest is updated every 100 iterations. For real-time rendering, we used an HTC Cosmos HMD with a Droolon aGlass to track the gaze point of the user. The parameter of the HVS models is set as $w_0 = 1/48^\circ$, $m = 1.32' per^\circ$, $k = 0.4$ which is the same as in [44] and [47]. The fovea region is defined with $e < 10^\circ$ which is the same as in [47], and the periphery region is defined with $e > 10^\circ$. The salient region is defined where $sal > 0.3$. The threshold ε_{DS} and ε_{con} are set as 0.5. The total number of layer L is set as 4. All experiments are performed on a PC workstation with a 3.8 GHz Intel(R) Core(TM) i7-10700KF CPU, 64 GB of memory, and an NVIDIA GeForce GTX 4090 GPU.

4.2 Comparison

We compare our method with the SOTA 3D Gaussian-based methods, D-GS [1] and MS-GS [2]. Similar to our approach, D-GS extends the original 3DGS by training a deformation field to control Gaussian deformation. However, it does not distinguish between dynamic and static Gaussians, deforming all Gaussians uniformly. It cannot be used for foveated rendering because it is trained only with full-resolution images. For a fair comparison, we utilize the optimal parameters reported in its paper and render full-resolution images for D-GS. MS-GS trains different scales' Gaussians for images of different resolutions, but there is no relationship between the Gaussians of different scales. Foveated rendering can be achieved with MS-GS by rendering in different regions with different scales of Gaussian. Since it is designed for static scenes, we extend it to dynamic scenes by training a deformation field for each scale of Gaussian and rendering in different regions with different scales of Gaussian to synthesize foveated images.

4.2.1 Quality

Visualization Results. The quality of our results is compared with those of ground truth and the comparison methods in Fig. 4. The first column of images shows the comparison between our method and the ground truth. The yellow circles on the image indicate the foveal region. The second column of images shows the close-ups of the rendered images for comparison (foveal region, blue and green rectangles). Our results demonstrate higher similarity to the ground truth, with clearer details in both the foveal and salient regions. In contrast, the comparison methods exhibit varying degrees of blurriness and artifacts. We summarize the qualitative improvements of our method as follows: 1)Accurate rendering of rigid body motion. As shown in cookie and torchcolate scenes, our method renders high-fidelity cookie and blowtorch. The cookie rendered by the D-GS and MS-GS methods loses the chocolate detail on top of the cookie, and the position of the rendered blowtorch is far from the real position. 2) Accurate rendering of fluid deformation. As shown in americano scene, our method renders the flow of coffee into the cup and its diffusion in the water with more detail, while the renderings of D-GS and MS-GS are more blurred, and the shape of the coffee changes a lot. 3) Successful rendering of physical phenomena. As shown in torchocolate scene, our method renders the blue flame of the blowtorch and the chocolate lit with high precision, while the images rendered by D-GS and MS-GS are blurry and incomplete. 4) Accurate rendering of specular reflections. As shown in the basin, plate, and cup scenes, our method renders the specular reflections much closer to the ground truth, while D-GS and MS-GS produce blurry specular reflections with large artifacts. These improvements can be explained as follows: 1) Our method models dynamic objects more accurately by separating them from static ones. 2) Our method uses the HVS models to guide rendering, rather than simply using layered Gaussians.

Quantitative Results. We use peak signal-to-noise ratio (PSNR) and learned perceptual image patch similarity (LPIPS) to quantitatively evaluate the rendering quality. To validate the effectiveness of foveated rendering, we partition the image into four regions for quality evaluation: the whole image (whole), the foveal region (foveal), the periphery region (periphery), and the salient region (salient), and compute PSNR and LPIPS for each region. Table 1 shows the quantitative rendering quality comparison in different regions between our method and prior methods on the Hyper-NeRF dataset and NeRF-DS dataset.

Our method demonstrates the best rendering quality on foveal and salient regions across all scenes. For the foveal region, our method outperforms D-GS and MS-GS because people's attention is typically attracted to dynamic objects, making the foveal region often lie on these dynamic objects. For the salient region, our method outperforms D-GS and MS-GS because most of the salient regions are dynamic objects and our method reconstructs them more accurately than D-GS. Additionally, we select Gaussians based on both acuity and saliency, using more accurate Gaussians for salient regions during rendering. As a result, despite rendering foveated images, our method achieves better results in salient regions compared to D-GS, which renders full-resolution images. Our method models dynamic objects more accurately by separating them from static ones. This separation allows the deformation field to fit only the dynamic object's deformation, avoiding interference from nearby static objects. For the whole and periphery regions, the image quality of our method is lower than that of D-GS. This is because D-GS renders the full-resolution images, while our method renders the foveated image. Although the rendering quality is reduced, most of the degradation occurs in periphery regions, which are less important since users are primarily focused on dynamic and foveal regions. Our method is better than MS-GS in all regions, both of which render foveated images. This is because our method uses the HVS models to guide rendering, while MS-GS realizes foveated rendering by simply rendering scaled Gaussians.

4.2.2 Performance

Table 2 shows the quantitative performance comparison between our method and the SOTA methods in different scenes. The results show that our method improves the performance by $4.87X \sim 11.33X$ compared to D-GS while maintaining higher image quality in the foveal and salient regions. The performance improvement is due to two factors: first, our method deforms only dynamic Gaussians, whereas D-GS deforms all Gaussians; second, D-GS renders full-resolution images, while our method renders foveated images. Compared to MS-GS, our method improves the performance by $2.51X \sim 7.57X$ with higher image quality in all regions. The performance improvement is because our method deforms only dynamic Gaussians, whereas MS-GS deforms all Gaussians. Compared to 4DGS [49], our method improves the performance by $3.28X \sim 7.79X$. Additionally, our method achieves higher rendering performance compared to NeRF-based methods, such as Hyper-NeRF [8], which has a frame rate of less than 1 fps, thanks to the Gaussian representation. In our experiments, users wear VR HMDs that require rendering binocular images, and our method renders binocular images with 60 fps to 170 fps for different dynamic scenes. Compared to rendering monocular images, the performance of rendering binocular images decreases by about 1%. This is because 73% of the total rendering time is spent on Gaussians deformation, and only 27% is spent on image rendering. In addition, only one deformation calculation is needed for rendering binocular images, and we render binocular images in parallel on the GPU, so the performance degradation of rendering binocular images is minimal.

4.3 Ablation Studies

We conducted ablation studies to validate the effectiveness of our proposed components, including dynamic-static separation (DS-S), acuity model-based selection and deformation (Acuity), and CSF model-based selection and rendering (CSF). First, we apply the MS-GS as the baseline by replacing the scaled Gaussians with the 3D Gaussian forest. The baseline constructs the 3D Gaussian forest in the same

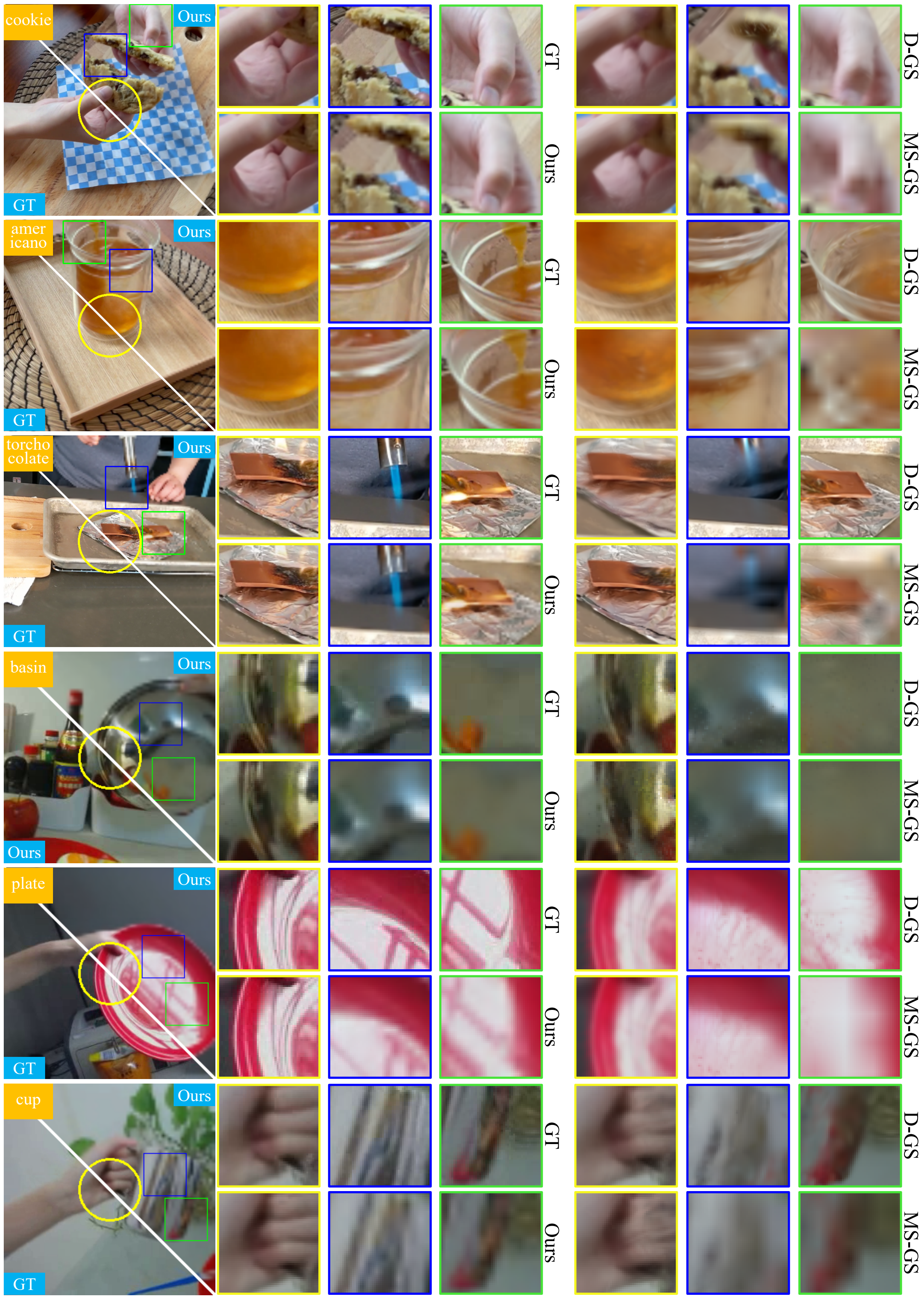

Fig. 4: Left: Comparison of the proposed Fov-GS (upper-right) and the ground truth image (GT, lower-left). Right: The close-ups of the images of both the foveal region (yellow) and periphery regions (blue and green). Compared with D-GS [1] and MS-GS [2], our results are closer to GT, with higher rendering quality.

Table 1: Quality comparison between our method, D-GS, and MS-GS on HyperNeRF and NeRF-DS datasets.

Region	whole			foveal			periphery			salient		
Method	Ours	D-GS	MS-GS	Ours	D-GS	MS-GS	Ours	D-GS	MS-GS	Ours	D-GS	MS-GS
Metric	PSNR↑											
cookie	26.54	**30.32**	24.04	**28.97**	28.86	28.71	23.57	**30.59**	23.63	**28.78**	28.41	20.81
americano	26.76	**29.41**	22.84	**30.15**	28.76	28.84	24.07	**29.51**	22.40	**27.59**	27.03	18.82
torchocolate	27.17	**30.11**	22.70	**29.83**	25.72	25.63	22.79	**31.02**	22.49	**28.45**	27.00	19.67
basin	20.57	**21.13**	19.99	**19.03**	18.30	18.29	20.99	**21.11**	20.36	**18.47**	17.89	16.98
plate	18.99	**20.24**	16.49	**21.66**	20.72	21.63	17.50	**21.09**	16.38	**18.88**	18.09	16.31
cup	19.64	**23.27**	16.19	**20.72**	20.08	20.09	17.08	**23.95**	15.86	**21.11**	20.95	17.00
Metric	LPIPS↑											
cookie	0.284	**0.071**	0.394	**0.031**	0.058	0.058	0.452	**0.061**	0.386	**0.154**	0.226	0.550
americano	0.161	**0.060**	0.327	**0.052**	0.059	0.058	0.278	**0.045**	0.312	**0.159**	0.167	0.482
torchocolate	0.182	**0.047**	0.387	**0.009**	0.063	0.063	0.406	**0.037**	0.376	**0.157**	0.162	0.540
basin	0.299	**0.234**	0.421	**0.213**	0.240	0.240	0.370	**0.162**	0.351	**0.409**	0.461	0.495
plate	0.260	**0.210**	0.343	**0.142**	0.143	0.145	0.335	**0.154**	0.286	**0.371**	0.473	0.516
cup	0.227	**0.154**	0.376	**0.143**	0.160	0.159	0.385	**0.119**	0.341	**0.400**	0.411	0.468

Table 2: Performance comparison between our method, D-GS, MS-GS and 4DGS on HyperNeRF and NeRF-DS datasets.

Performance/ms↓	cookie	americano	torchocolate	basin	plate	cup
Ours	**16.21**	**14.91**	**16.37**	**6.98**	**6.13**	**5.87**
D-GS	123.65	168.92	126.33	34.48	32.25	28.57
MS-GS	74.31	112.9	72.58	17.57	15.20	17.65
4DGS	88.34	116.22	86.79	24.21	22.04	19.28

way as in Sec 3.2 and Sec 3.3, without separating dynamic and static Gaussians, and the Gaussians to deform and render is selected based on its position of last frame. Then, we integrate our dynamic-static separation to the baseline (baseline+DS-S), constructing 3D Gaussian forest by separating dynamic and static trees and deforming only the dynamic Gaussians during rendering. Next, we incorporate our acuity model-based selection and deformation to the baseline (baseline+DS-S+Acuity), deforming only the selected dynamic Gaussians based on acuity during rendering. Finally, we apply our CSF model-based selection and rendering to the baseline to form the complete Fov-GS (baseline+DS-S+Acuity+CSF). Numerical results are presented in Table 3, and qualitative results are visualized in Fig. 5.

As shown in Table 3, compared with Baseline, the Baseline+DS-S improves the quality in PSNR by 0.62, and the performance in ms by 32.16, demonstrating separating dynamic Gaussians from static Gaussians not only improves reconstruction accuracy but also dramatically improves rendering performance. With the addition of the Acuity, the Baseline+DS-S+Acuity further improves the quality in PSNR by 0.71, illustrating that acuity-based selection and deformation can more accurately select the Gaussian for deformation compared to selection based on the position of the previous frame. While there is a reduction in rendering performance, it is minimal. With the addition of the CSF, our complete Fov-GS (Baseline+DS-S+Acuity+CSF) further improves the rendering quality by 1.62. The large increase in quality is due to that the images are rendered with more detailed Gaussians, thus making the visual perceptual quality align with the capabilities of the HVS. The small decrease in performance is worth compared to the large increase in quality.

The visualization results in Fig. 5 further demonstrate the effectiveness of our proposed components. The comparison results between Ours and the Baseline indicate that separating dynamic Gaussians from static Gaussians improves quality. The other two components affect the periphery region mostly, thus the foveal regions of Ours, Baseline+DS-S+Acuity, and Baseline+DS-S+Acuity+CSF are similar. The comparison results between Baseline+DS-S+Acutiy and Baseline+DS-S indicate that acuity-based selection can more accurately select the Gaussian for deformation. The comparison results between Ours and Baseline+DS-S+Acutiy indicate that rendering with more detailed Gaussians improves rendering quality.

Table 3: Quality and Performance Ablation Study

Method	Quality/PSNR↑	Performance/ms↓
Baseline	20.65	49.71
Baseline+DS-S	21.27	**8.55**
Baseline+DS-S+Acuity	21.98	9.18
Baseline+DS-S+Acuity+CSF (Ours)	**23.57**	11.71

We illustrate the results of dynamic-static separation in different scenes in Fig. 6. Although the images rendered only with static Gaussian exhibit some floaters due to poor dynamic-static separation at the edges of dynamic objects, our method successfully separates the dynamic hand and cookie. Static Gaussians well reconstruct the static parts that are occluded by the dynamic objects. Accurately reconstructing and rendering the occluded static part is very difficult because it is not visible under the current view, and these Gaussians can be reconstructed only based on the images of other views. Due to we separate the dynamic Gaussians from the static Gaussians, these occluded static Gaussians can be reconstructed accurately. However, the images rendered only with static Gaussian has some floaters. This is because the Gaussian at the edges of the dynamic objects has poor dynamic-static separation.

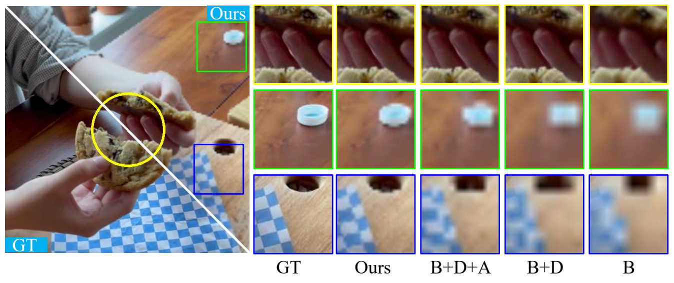

Fig. 5: Qualitative comparison for ablation study. As illustrated in the close-ups, our results (Ours) are closer to ground truth (GT), compared with Baseline (B), Baseline+DS-S (B+D), and Baseline+DS-S+Acuity (B+D+A).

5 User Study

We design a within-subject study to evaluate the visual perceptual quality on the Hyper-NeRF and NeRF-DS datasets of our method compared with the previous methods.

5.1 User Study Design

Participants and Setup. 15 participants (10 males and 5 females, aged between 21-30) were recruited in this study, and all of them have

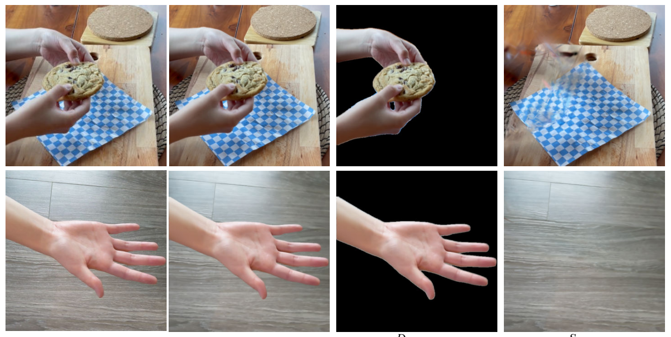

Fig. 6: Visualization of the ground truth image (a, GT), the image rendered with full forest (b, Ours), the images rendered with dynamic (c) and static trees (d) separately. The first line is the images corresponding to the cookie scene and the second line is the images corresponding to the hand scene.

had experiences in VR HMDs. Each participant wore an HTC Cosmos HMDs for the user study. The research was performed under the oversight of Biology and Medical Ethics Committee of Beihang University, with protocol number BM20240277. Consent from the human subjects in the research was obtained.

Conditions. The conditions included the ground truth images (GT), our method (Ours), D-GS, and MS-GS.

Task and Procedure. There are 6 scenes used in the experiment, which are cookie, americano, torchocolate, basin, plate, and cup. We randomly select the camera parameters and time parameters from the test split, and the selected parameters are arranged in chronological order, so as to obtain reasonably deformed dynamic scene videos, each of which lasts for 8s. We asked each participant to participate in the experiment with 4 conditions in 6 scenes. Initially, we presented the GT sequence to the participants, informing them that this was the benchmark result. Subsequently, we displayed the videos generated by GT, Ours, D-GS, and MS-GS in a random order, asking participants to rate the visual perceptual quality of each video sequence. The viewing counts for all methods in the experiment were kept balanced. The visual perceptual quality score consists of 5 confidence levels: 5 means that no artifacts were perceived at all, 4 means they perceived acceptable artifacts for a few very short moments, 3 means they perceived acceptable artifacts, 2 means they perceived noticeable artifacts, 1 means they perceived obvious artifacts. To mitigate the effects of visual fatigue, after completing the ratings, participants are given a 10-second rest before proceeding to the next test.

Statistical analysis. We compared the values of different conditions. First, the normality of the data was assessed using the Shapiro-Wilk test. Then the comparison was performed with a repeated-measures ANOVA if the values showed a normal distribution. When values did not follow a normal distribution, the comparison was performed using a Wilcoxon signed-rank test. In addition to the p-value of the statistical test, we also estimate the size of the effect using Cohen's d. The d values are translated to qualitative effect size estimates of Huge ($d > 2.0$), Very Large ($2.0 > d > 1.2$), Large ($1.2 > d > 0.8$), Medium ($0.8 > d > 0.5$), Small ($0.5 > d > 0.2$), and Very Small ($0.2 > d > 0.01$).

5.2 Results and Discussion

As shown in Fig. 7, we calculate the average score of all conditions, and use the p-value and Cohen's d to estimate the difference between the comparison conditions and Ours. The results indicate a significant improvement in our average score compared to both D-GS and MS-GS, which is closest to the GT. The p-value$= 0.15$ of scores between Ours and GT, with Cohen's d$= 0.49$, indicates a Small effect size. This suggests that our method has statistically visual perceptual similarity with the ground truth. The p-value< 0.001 of scores between Ours and D-GS, with Cohen's d$= 1.44$, indicates a Very Large effect size. The p-value< 0.001 of scores between Ours and D-GS, with Cohen's d$=$ 1.85, indicates a Very Large effect size. These results demonstrate that compared to D-GS and MS-GS, our method significantly enhances the visual perceptual quality of synthesized foveated images.

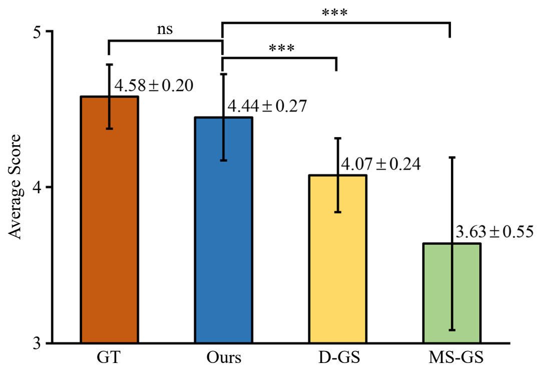

Fig. 7: The average scores and standard deviations for all conditions in our user study. Error bars indicate standard deviation. Asterisks denote statistical significance between different conditions.

6 Conclusion

We propose a 3D Gaussian splatting method for foveated rendering of dynamic scenes, named Fov-GS. In this method, we propose a 3D Gaussian forest representation for dynamic scenes. We adopt a forest initialization method based on dynamic-static separation to construct the 3D Gaussian forest, and a forest optimization method based on deformation field and Gaussian decomposition to optimize the 3D Gaussian forest and deformation field. For efficient foveated rendering, we propose a 3D Gaussian forest rendering method based on HVS models which selects, deforms, and renders Gaussian forest based on the acuity model and CSF model. Quantitative evaluation and visualization experiments indicate that the proposed Fov-GS remarkably outperforms existing methods in terms of rendering quality and efficiency, achieving state-of-the-art performance. Our ablation studies further validate the efficiency of the proposed components, and the user study shows that our rendering results significantly improve the visual perceptual quality compared to previous methods.

Our method has several limitations. The first limitation is that the quality of the dynamic-static separation is strongly influenced by the quality of the input semantic segmentation. During the initialization of the 3D Gaussian forest, semantic segmentation is used for dynamic-static separation. Poor segmentation quality affects the reconstruction accuracy of dynamic object edges, leading to misclassification of some dynamic Gaussians as static Gaussians and resulting in artifacts, such as floaters, in the rendered static images. In the future, we intend to use more advanced segmentation algorithms. In addition, for Gaussians located at the boundary of dynamic and static separation, we will further refine the separation by calculating their motion consistency with the center Gaussians. The second limitation is that the rendering performance decreases for scenes containing objects with fast motion. This is because we use a two-step selection strategy to determine which Gaussians need to be deformed during rendering. When dynamic objects move slowly, the second step only needs to update a few Gaussians, allowing for quick deformation computations. However, this strategy is less effective when the objects are moving quickly. In the future, we plan to predict deformation along with the speed and direction of the objects. This will allow us to estimate the position of each Gaussian in the current frame based on the speed and direction predicted in the previous frame, thereby simplifying the selection of Gaussians in the current frame. The third limitation is that our method is less effective when dealing with moving shadows due to misclassified of the Gaussian for reconstructing shadows. In the future, we intend to reconstruct the lighting concurrently with the scene geometry, thereby improving the rendering quality of dynamic shadow.

Acknowledgments

This work is supported by the National Natural Science Foundation of China through Projects 62372026 and 61932003, Beijing Science and Technology Plan Project Z221100007722004, and the National Key R&D plan 2019YFC1521102.

References

[1] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20331–20341, 2024. 1, 2, 3, 6, 7

[2] Zhiwen Yan, Weng Fei Low, Yu Chen, and Gim Hee Lee. Multi-scale 3d gaussian splatting for anti-aliased rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20923–20931, 2024. 1, 2, 5, 6, 7

[3] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. 1, 5, 6

[4] Lili Wang, Xuehuai Shi, and Yi Liu. Foveated rendering: A state-of-the-art survey. *Computational Visual Media*, 9(2):195–228, 2023. 2

[5] Guikun Chen and Wenguan Wang. A survey on 3d gaussian splatting. *arXiv preprint arXiv:2401.03890*, 2024. 2

[6] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I*, page 405–421, Berlin, Heidelberg, 2020. Springer-Verlag. 2

[7] Tianhao Wu, Fangcheng Zhong, Andrea Tagliasacchi, Forrester Cole, and Cengiz Oztireli. D2nerf: self-supervised decoupling of dynamic and static objects from a monocular video. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2024. Curran Associates Inc. 2

[8] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *ACM Trans. Graph.*, 40(6), dec 2021. 2, 5, 6

[9] 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. 2

[10] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Wang Xinggang. 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*, 2023. 2

[11] Bardienus P Duisterhof, Zhao Mandi, Yunchao Yao, Jia-Wei Liu, Mike Zheng Shou, Shuran Song, and Jeffrey Ichnowski. Md-splatting: Learning metric deformation from 4d gaussians in highly deformable scenes. *arXiv preprint arXiv:2312.00583*, 2023. 2

[12] Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. Dynmf: Neural motion factorization for real-time dynamic view synthesis with 3d gaussian splatting. *arXiV*, 2023. 2

[13] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19447–19456, 2024. 2, 5

[14] Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A hierarchical 3d gaussian representation for real-time rendering of very large datasets. *ACM Transactions on Graphics (TOG)*, 43(4):1–15, 2024. 2, 5

[15] 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

[16] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. *arXiv preprint arXiv:2312.00732*, 2023. 2

[17] Masahiro Fujita and Takahiro Harada. Foveated real-time ray tracing for virtual reality headset. *Light Transport Entertainment Research*, 2014. 2

[18] Martin Weier, Thorsten Roth, Ernst Kruijff, André Hinkenjann, Arsène Pérard-Gayot, Philipp Slusallek, and Yongmin Li. Foveated real-time ray tracing for head-mounted displays. In *Computer Graphics Forum*, volume 35, pages 289–298. Wiley Online Library, 2016. 2

[19] Matias K Koskela, Kalle V Immonen, Timo T Viitanen, Pekka O Jääskeläinen, Joonas I Multanen, and Jarmo H Takala. Instantaneous foveated preview for progressive monte carlo rendering. *Computational Visual Media*, 4:267–276, 2018. 2

[20] Youngwook Kim, Yunmin Ko, and Insung Ihm. Selective foveated ray tracing for head-mounted displays. In *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 413–421. IEEE, 2021. 2

[21] Nicholas T Swafford, José A Iglesias-Guitian, Charalampos Koniaris, Bochang Moon, Darren Cosker, and Kenny Mitchell. User, metric, and computational evaluation of foveated rendering methods. In *Proceedings of the ACM Symposium on Applied Perception*, pages 7–14, 2016. 2

[22] Fabio Policarpo and Manuel M Oliveira. Relief mapping of non-height-field surface details. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 55–62, 2006. 2

[23] Brian Guenter, Mark Finch, Steven Drucker, Desney Tan, and John Snyder. Foveated 3d graphics. *ACM transactions on Graphics (tOG)*, 31(6):1–10, 2012. 2, 5

[24] Michael Stengel, Steve Grogorick, Martin Eisemann, and Marcus Magnor. Adaptive image-space sampling for gaze-contingent real-time rendering. In *Computer Graphics Forum*, volume 35, pages 129–139. Wiley Online Library, 2016. 2

[25] Eric Turner, Haomiao Jiang, Damien Saint-Macary, and Behnam Bastani. Phase-aligned foveated rendering for virtual reality headsets. In *2018 IEEE conference on virtual reality and 3D user interfaces (VR)*, pages 1–2. IEEE, 2018. 2

[26] Karthik Vaidyanathan, Marco Salvi, Robert Toth, Theresa Foley, Tomas Akenine-Möller, Jim Nilsson, Jacob Munkberg, Jon Hasselgren, Masamichi Sugihara, Petrik Clarberg, et al. Coarse pixel shading. In *Proceedings of High Performance Graphics*, pages 9–18. 2014. 2

[27] Lei Yang, Dmitry Zhdan, Emmett Kilgariff, Eric B Lum, Yubo Zhang, Matthew Johnson, and Henrik Rydgård. Visually lossless content and motion adaptive shading in games. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 2(1):1–19, 2019. 2

[28] Anjul Patney, Marco Salvi, Joohwan Kim, Anton Kaplanyan, Chris Wyman, Nir Benty, David Luebke, and Aaron Lefohn. Towards foveated rendering for gaze-tracked virtual reality. *ACM Transactions on Graphics (TOG)*, 35(6):1–12, 2016. 2

[29] ARAUJO Helder. An introduction to the log-polar mapping. In *II Workshop on Cibernetic Vision, December, 1996*, 1996. 2

[30] V Javier Traver and Alexandre Bernardino. A review of log-polar imaging for visual perception in robotics. *Robotics and Autonomous Systems*, 58(4):378–398, 2010. 2

[31] Xiaoxu Meng, Ruofei Du, Matthias Zwicker, and Amitabh Varshney. Kernel foveated rendering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(1):1–20, 2018. 2

[32] Jiannan Ye, Anqi Xie, Susmija Jabbireddy, Yunchuan Li, Xubo Yang, and Xiaoxu Meng. Rectangular mapping-based foveated rendering. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 756–764. IEEE, 2022. 2

[33] David Li, Ruofei Du, Adharsh Babu, Camelia D Brumar, and Amitabh Varshney. A log-rectilinear transformation for foveated 360-degree video streaming. *IEEE Transactions on Visualization and Computer Graphics*, 27(5):2638–2647, 2021. 2

[34] Xuehuai Shi, Lili Wang, Jian Wu, Runze Fan, and Aimin Hao. Foveated stochastic lightcuts. *IEEE Transactions on Visualization and Computer Graphics*, 28(11):3684–3693, 2022. 2

[35] David R Walton, Rafael Kuffner Dos Anjos, Sebastian Friston, David Swapp, Kaan Akşit, Anthony Steed, and Tobias Ritschel. Beyond blur: Real-time ventral metamers for foveated rendering. *ACM Transactions on Graphics*, 40(4):1–14, 2021. 2

[36] Brooke Krajancich, Petr Kellnhofer, and Gordon Wetzstein. Towards attention–aware foveated rendering. *ACM Transactions on Graphics (TOG)*, 42(4):1–10, 2023. 2

[37] Xuehuai Shi, Lili Wang, Xinda Liu, Jian Wu, and Zhiwen Shao. Scene-aware foveated neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics*, 2024. 2

[38] Nianchen Deng, Zhenyi He, Jiannan Ye, Budmonde Duinkharjav, Praneeth Chakravarthula, Xubo Yang, and Qi Sun. Fov-nerf: Foveated neural radiance fields for virtual reality. *IEEE Transactions on Visualization and Computer Graphics*, 28(11):3854–3864, 2022. 2, 5

[39] David Bauer, Qi Wu, and Kwan-Liu Ma. Fovolnet: Fast volume rendering using foveated deep neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):515–525, 2022. 2

[40] Weikai Lin, Yu Feng, and Yuhao Zhu. Rtgs: Enabling real-time gaussian splatting on mobile devices using efficiency-guided pruning and foveated

rendering. *arXiv preprint arXiv:2407.00435*, 2024. 2

[41] Linus Franke, Laura Fink, and Marc Stamminger. Vr-splatting: Foveated radiance field rendering via 3d gaussian splatting and neural points. *arXiv preprint arXiv:2410.17932*, 2024. 2

[42] Zhiming Hu, Congyi Zhang, Sheng Li, Guoping Wang, and Dinesh Manocha. Sgaze: A data-driven eye-head coordination model for real-time gaze prediction. *IEEE Transactions on Visualization and Computer Graphics*, 25(5):2002–2010, 2019. 2

[43] Zhiming Hu, Sheng Li, Congyi Zhang, Kangrui Yi, Guoping Wang, and Dinesh Manocha. Dgaze: Cnn-based gaze prediction in dynamic scenes. *IEEE Transactions on Visualization and Computer Graphics*, 26(5):1902–1911, 2020. 2

[44] Brian Guenter, Mark Finch, Steven Drucker, Desney Tan, and John Snyder. Foveated 3d graphics. *ACM transactions on Graphics (tOG)*, 31(6):1–10, 2012. 2, 6

[45] Ho Kei Cheng, Seoung Wug Oh, Brian Price, Alexander Schwing, and Joon-Young Lee. Tracking anything with decoupled video segmentation. In *ICCV*, 2023. 3

[46] Harry Nyquist. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, 47(2):617–644, 1928. 3

[47] Runze Fan, Xuehuai Shi, Kangyu Wang, Qixiang Ma, and Lili Wang. Scene-aware foveated rendering. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–10, 2024. 5, 6

[48] Zhiwen Yan, Chen Li, and Gim Hee Lee. Nerf-ds: Neural radiance fields for dynamic specular objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8285–8295, 2023. 5

[49] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20310–20320, June 2024. 6