Foveated Segment Anything

Ziqi Wang* Tianhua Xia* Haiyu Wang Zhenyuan Dong Qiance Tang
New York University
{zw4763, tx856, hw3689, zd2362, qt2094}@nyu.edu

Jieyu Lin Ziyun Li Barbara De Salvo
Meta Reality Labs
{jyericlin, liziyun, barbarads}@meta.com

Sai Qian Zhang
New York University
sai.zhang@nyu.edu

Abstract

Augmented Reality (AR) encompasses transformative technologies that are redefining how humans interact with their environment. A key component of AR is image segmentation, which breaks down the user's front-view scene into distinct regions for analysis. This process is essential for accurately overlaying digital content onto the physical world by detecting and isolating relevant objects. However, despite its importance, image segmentation poses significant computational demands and latency issues on AR devices, which can severely impact the overall user experience. In this paper, we propose Foveated Segment Anything (FoSAM), a framework built upon the Segment Anything Model (SAM) that utilizes real-time gaze data to focus segmentation on regions of interest, substantially lowering computational cost. Experimental results show that FoSAM reduces computational cost by over $50\times$*, enabling a seamless visual experience for users, as confirmed by our user study. The code is available at* https://github.com/SAI-Lab-NYU/FoSAM.git.

1. Introduction

Image segmentation [4, 21, 29, 52] is a core task in computer vision that involves dividing an image into meaningful regions to support visual content analysis and interpretation. Building on this, instance segmentation [5, 16, 49, 56] identifies and outlines each individual object within a scene. This capability is particularly crucial in augmented reality, where accurate object detection and separation enable precise interaction and seamless overlay of virtual content onto the physical environment, enhancing user experience.

Instance segmentation serves as a foundational component for numerous AR applications. For example, in educational contexts (Figure 1 (a)), segmentation can detect individual components of complex diagrams the user is viewing, enhancing student engagement and understanding. These segmented objects can also be passed to downstream applications (e.g., vision-language models (VLMs)) to provide detailed explanations or context-aware information. In an AR-assisted grocery shopping scenario, real-time segmentation enables users to identify products on a shelf as they look at them. Importantly, many of these applications only require isolating the object of interest rather than assigning a semantic category, making *class-agnostic segmentation* sufficient for interactive AR tasks. Furthermore, segmentation enables direct object manipulation, allowing users to edit or interact with specific elements in their environment, creating a seamless bridge between the physical and virtual worlds. Additional AR use cases leveraging image segmentation are discussed in [3, 13, 14, 46].



Figure 1. (a) and (b) show applications of VLM in AR. (c) An example on gaze movement for the AR scenario.

The Segment Anything Model (SAM) [21, 41] is among the most advanced models for class-agnostic segmentation today. However, despite its strong performance, SAM is computationally intensive, making it difficult to deploy on resource-limited AR devices that handle high-resolution imagery. This results in high overhead and latency, degrading overall system performance and user experience.

Unlike typical image processing scenarios, AR device users display distinct behavioral patterns. **They tend to focus on small, specific areas within a scene before shifting**

*Equal contribution

their attention elsewhere. For instance, as shown in Figure 1 (c), a user wearing AR glasses may first focus on the door (left) and then turn their head to look at the bookshelf (right). This behavior naturally segments the video stream into distinct *video fragments* (VF) based on head movements. In the first VF, where the user's gaze remains on the door across several frames, the segmentation results can be reused, avoiding redundant computation. Similarly, in the second VF, segmentation efforts can be focused solely on the bookshelf. Moreover, as illustrated in Figure 1 (a), it is often beneficial to restrict segmentation to the instances of interest (IOI) currently under the user's gaze, as this typically reflects the user's active focus. This leads to a more efficient approach for AR image segmentation: **prioritize processing for gaze-identified IOIs while ignoring non-essential regions**. This strategy aligns with the principles of *foveated rendering* [35], which improves rendering efficiency by showing full-resolution detail only within the user's line of sight, reducing visual fidelity in peripheral areas to save computations.

In this work, we aim to substantially reduce the computational cost of SAM-style models by leveraging the native user gaze signal in AR, thereby bringing leading segmentation model to AR devices. To achieve this, we introduce a novel segmentation framework that restricts segmentation to only the regions the user is actively viewing. Our proposed solution, called *Foveated SAM* (FoSAM), achieves more than a $50\times$ reduction in processing latency over the original model without compromising the user's visual experience. Our contributions can be summarized as follows:

- We propose *FoSAM*, which processes high-resolution input images and performs instance segmentation on IOI with extremely low computational cost.
- Building on FoSAM, we introduce *FoSAM Streaming Algorithm* (FSA), an efficient instance segmentation framework tailored for real-time AR/VR applications. FSA, exploits temporal continuity across frames and human gaze patterns to optimize segmentation, delivering enhanced performance in dynamic AR environments.
- By leveraging AR users' eye-behavior signals, FoSAM achieves an average $24\times$ speedup and reduces latency to around $20ms$, with minimal impact to segmentation accuracy.

2. Background and Related Work

2.1. Literature Review on Segmentation

Semantic segmentation [4, 7, 9, 10, 24, 29, 33, 48, 53] is a key task in computer vision that involves partitioning an image into distinct regions or segments to simplify content analysis and interpretation. A more advanced form of this task is instance segmentation [6, 16, 25, 34], which aims to distinguish individual instances of the same object class. Unlike semantic segmentation, which labels each pixel without differentiating between instances of the same object class, instance segmentation provides a finer level of detail by distinguishing each instance. To improve segmentation efficiency, previous research has focused on developing learnable input downsampling techniques that adjust sampling resolution in a selective manner [19, 31, 42, 47, 58]. Previous approaches perform image downsampling at the pixel level using Convolutional Neural Networks (CNNs). In contrast, FoSAM is a transformer-based model that uses a learnable mechanism to select image tokens at the token level, guided by the user's gaze location and IOI. Furthermore, previous studies on video instance segmentation [26, 39, 50, 51, 55, 56] typically process multiple consecutive frames jointly. While this design improves segmentation performance, it also introduces additional latency to wait for all frames to be available.

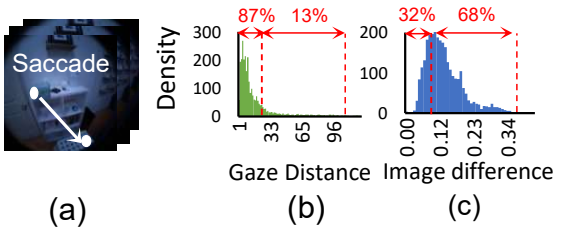


Figure 2. (a) Video Fragments in the user's view. (b) Distribution of the gaze distances. (c) Distribution of the image differences.

Although SAM [21] and its variants such as ESAM [54] support point-based visual prompts, where the mask is displayed only for the point selected IOI. However, SAM first processes the entire image using its visual encoder, where the majority of the computation occurs, and then selectively decodes and displays the mask corresponding to the selected IOI. As a result, the computation remains, leading to substantial overhead. In contrast, **FoSAM mitigates this issue at a fundamental level by processing only the selected image region from scratch**, thereby eliminating unnecessary global computation and significantly reducing the overall computational cost at its source. Compared to Segment This Thing [43], which applies a fixed patch pattern centered around the gaze location, our approach uses learnable downsampling to dynamically adapt the sampling pattern to the size of objects, improving segmentation efficiency. Finally, FSNet [58] also produces adaptively subsampled inputs. However, this strategy is less effective for SAM, since SAM cannot directly exploit smaller inputs to reduce computation. Instead, for SAM to function correctly, **the downsampled input is resized back to the original resolution (e.g., (1024×1024)) before being fed into the subsequent SAM blocks, which largely negates the intended computational savings**. FoSAM directly mitigates this issue by reducing the number of tokens processed by the SAM visual encoder.

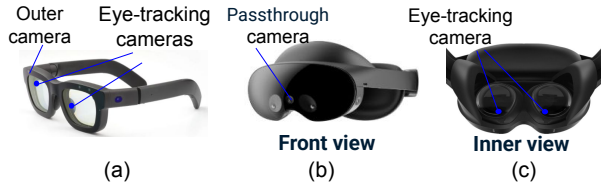


Figure 3. (a) Meta Orion AR glass. (b) Front view and (c) inner view of Meta Quest Pro VR headset.

2.2. Human Eye Behavior in AR/VR Environment

The human eye operates in three main modes of movement: fixation, where the eye remains stationary and focuses on a single point; saccadic movements, rapid, jerky shifts in gaze from one target to another; and smooth pursuit, where the eye follows a moving object in a smooth manner. Among these, fixation dominates the majority of visual activity. During fixation, the gaze remains locked on a single point, and visual acuity is concentrated around this focal region. In addition to fixation, the eye performs saccadic movements, rapidly shifting gaze between points of interest to gather visual information. During a saccade, visual sensitivity is temporarily reduced, a phenomenon known as saccadic suppression, which prevents the perception of motion blur caused by rapid eye movement [23, 32].

Figure 3 (a-c) present the Meta Aria AR glasses [12] and Meta Quest Pro [18], respectively. Both devices feature an outward-facing camera that continuously captures high-resolution images of the user's front view, along with an inward-facing eye-tracking (ET) camera that records monochrome images of the user's eyes. Most modern AR devices are equipped with eye trackers that take the eye images and provide highly precise gaze direction estimates with minimal latency. To investigate human gaze behavior while using AR devices, we perform an in-depth analysis using the Aria Everyday Activities Dataset [30]. This dataset contains 143 sequences of frames captured by the AR device, which align with the user's field of view and track gaze movements in different environments.

As the user adjusts their head orientation, the front view also changes. To quantify these head movements, we calculate the image difference by measuring the Euclidean distance between corresponding pixels of consecutive frames. If the pixel difference is below a certain threshold, the frames are considered highly similar and nearly indistinguishable to the human eye. These frames are grouped into a single video fragment (VF), as shown in Figure 2 (a). As shown in Figure 2 (c), 32% of consecutive frames exhibit less than 9% pixel value changes, **suggesting a similarity between the consecutive frames and potential for reusing segmentation results across frames**.

Furthermore, within each VF, segmentation results can be reused if the gaze remains relatively stable and consistently points to the same IOI. To support this, we analyze the distances (in pixels) between consecutive gaze locations within a VF. Our analysis reveals that a threshold of 22 pixels effectively groups gaze locations during the fixation phase, where the user focuses on a single IOI. Gaze distances exceeding this threshold indicate a saccade, as seen with the rapid gaze changes in VF2 of Figure 2 (a). As shown in Figure 2 (b), 87% of the frames within each VF have a gaze distance of less than 22 pixels, indicating that AR users typically focus on one or two IOIs during each VF. This provides an opportunity to enhance segmentation efficiency by **focusing processing on the IOI and reusing segmentation results when gaze shifts are minimal**.

Table 1. latencies of segmentation models on AR platform.

Image Size	SAM-B	ESAM-S	ESAM-T
1024^2	1897.4 ms	600.3 ms	279.7 ms
640^2	441.8 ms	102.6 ms	46.7 ms

2.3. Segmentation Task Latency

Despite its significance, the segmentation task presents considerable computational challenges, especially on resource-constrained AR devices, primarily because of the high resolution of images these devices capture. This heavy data load leads to significant computational latency, severely limiting performance and responsiveness. To explore this issue, we evaluate the processing latency of the image segmentation task using well-established neural networks, including Segment Anything Base model (SAM-B) [21], Efficient SAM Tiny (ESAM-T) [54], and ESAM Small (ESAM-S). The latencies are profiled on the Qualcomm XR2 Gen2 platform [38] using the Qualcomm AI Hub toolkit [37], which is equipped by Meta Quest Pro [18] and Meta Aria AR glasses [12]. Table 1 shows that the processing latencies for 1024×1024 input images are 1897.4ms, 600.3ms, and 279.7ms for SAM-base (SAM-B), Efficient SAM-small (ESAM-S), and Efficient SAM-tiny (ESAM-T), respectively. Furthermore, even with a smaller input resolution of 640×640, the latency still ranges from 441.8ms to 46.7ms. These latencies exceed the threshold required for a seamless visual experience, as prior studies indicate that latencies below 30 ms are essential for maintaining optimal visual fluidity [1, 2, 20].

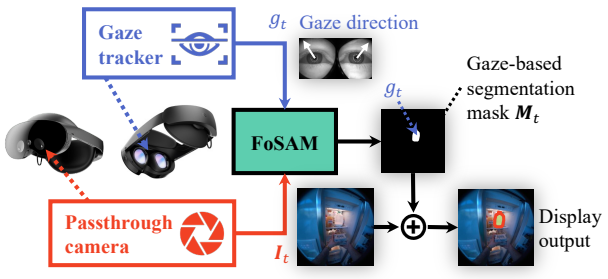


Figure 4. System Deployment of FoSAM.

3. Methodology

Figure 4 illustrates the computational pipeline of the FoSAM framework. During runtime, the inward-facing sensor of the AR/VR device continuously captures images of the user's eye and sends them to the gaze tracker, which estimates the gaze direction with high accuracy (5 to 10 milliseconds) [17, 44]. At timestep t, this estimated gaze direction g_t, along with a high-resolution image I_t captured by the front-facing camera, is provided as input to FoSAM. FoSAM then produces a segmentation map M_t focused on the IOI, which can be reused across frames where the gaze location remains similar. To simplify notation, we omit the subscript t in the following sections. The core idea of FoSAM is to employ a lightweight token selector to identify and select a subset of image tokens, which are then passed to the segmentation network. By processing only the selected tokens, the segmentation network significantly reduces its computational overhead.

3.1. Preliminaries

SAM and its variants (e.g., ESAM) consist of three main components: an *image encoder* $E(.)$, a *prompt encoder* $R(.)$, and a *mask decoder* $D(.)$. Given an input RGB image $I \in \mathbb{R}^{H\times W\times 3}$, it is first tokenized and positionally embedded into a set of tokens $X = \{x_{ij}\}$ with a size of $H_e \times W_e$, where each token $x_{ij} \in \mathbb{R}^{1\times d_t}$. H_e and W_e equal $\frac{H}{P}$ and $\frac{W}{P}$, where P is the patch size of the tokenizer. These tokens are passed through the image encoder to produce $Y = \{y_{ij}\} = E(X)$, which retains the same spatial resolution. In parallel, a point-based gaze prompt g is processed by the prompt encoder $R(.)$, and its output is fused with Y in the decoder to generate the final segmentation mask $M \in \mathbb{R}^{H\times W}$. In ESAM, the image encoder $E(.)$ is a 12-layer Vision Transformer, and the decoder $D(.)$ consists of a pre-trained two-layer CNN and a two-layer ViT. Since $E(.)$ dominates the computational cost, we focus on optimizing it by introducing a *gaze-guided Token Importance Encoder (TIEncoder)* $S(.)$, a *Gaussian Predictor* $G(.)$, and an *Adaptive Sampling module*. Together, they will select a subset of K important tokens from X based on the gaze direction g, reducing the input size to $E(.)$ from $H_e \times W_e$ to K and greatly decreasing the computational cost of the segmentation task. The entire selection process is end-to-end trainable, resulting in strong accuracy. Next is the design of our TIEncoder $S(.)$.

3.2. FoSAM framework

3.2.1. Gaze-Guided Token Importance Encoder

The gaze-guided TIEncoder is designed to assign a Token Importance Score (TIScore) to each token X based on the user's gaze direction g. A common approach predicts token-wise importance scores independently for each token [27, 28, 40, 45], and then removes those with low scores. However, this strategy often leads to poor accuracy because it lacks spatial coherence. To address these limitations, we propose estimating the TIScores of tokens using a 2D Gaussian distribution. The top portion of Figure 5 illustrates the overall pipeline, with the black and blue arrows showing the inference flow. The input image I is first uniformly downsampled from $H \times W$ to $\frac{H}{v} \times \frac{W}{v}$, resulting in a lower-resolution image I', where v denotes the downsampling ratio. This step effectively reduces the computational cost of the TIEncoder function $S(.)$. Next, I' is passed through a pre-trained ESAM tokenizer and a positional embedding module to generate token features $X' = \{x'_{ij}\}$ with dimensions $H'_e \times W'_e$. X' is then processed by the TIEncoder $S(.)$, which outputs $Z = S(X')$ containing the same number of tokens. Tokens Z are selected based on their spatial proximity to the gaze point g obtained from the gaze tracker. The token nearest to g, denoted as $z_g \in Z$, captures essential information about the IOI, such as its shape and scale, and serves as the foundation for predicting a bounding window that encloses the IOI.

To simplify the window generation process while preserving spatial continuity, we introduce a Gaussian predictor to estimate the TIScore of each token $x \in X$. This predictor, $G(\cdot)$, takes z_g as input and outputs the parameters of a 2D Gaussian distribution: the mean (μ_x, μ_y), standard deviations (σ_x, σ_y), and correlation coefficient ρ. Let $f_{G(z_g)}(.,.)$ represent the probability density function of Gaussian distribution with the parameter specified by $G(z_g)$. The TIScore for each token $x \in X$ is then computed as its likelihood under the predicted 2D Gaussian distribution, denoted as $f_{G(z_g)}(c_i(x), c_j(x))$, where $c_i(x)$ and $c_j(x)$ is the 2D coordinate of x within the token map.

3.2.2. Adaptive Token Sampling Mechanism

Based on the TIScores generated by the 2D Gaussian map, only a subset of tokens with the highest scores are retained. Specifically, a larger IOI results in more tokens receiving high TIScores, corresponding to a Gaussian distribution with larger variance. Conversely, smaller IOIs produce lower-variance distributions and fewer high-importance tokens. As a result, we retain the top K tokens, where K is determined adaptively based on the Gaussian distribution. Specifically, let $\tau \in [0, 1]$ denotes a threshold of the token elimination, token x_i is kept only if specified as follows:

$$f_{G(z_g)}(c_i(x), c_j(x)) \geq \max_{x\in X}[f_{G(z_g)}(c_i(x), c_j(x))] \times \tau \quad (1)$$

Here, τ denotes the threshold applied to the TIScores. Additionally, we define a minimum token count ϵ to ensure a sufficient number of tokens are selected. If the number of tokens satisfying the condition in Equation 1 falls below ϵ, the top-ϵ tokens are selected based on their TIScores. By setting τ to a predefined value (e.g., 0.01), Equation 1 en-

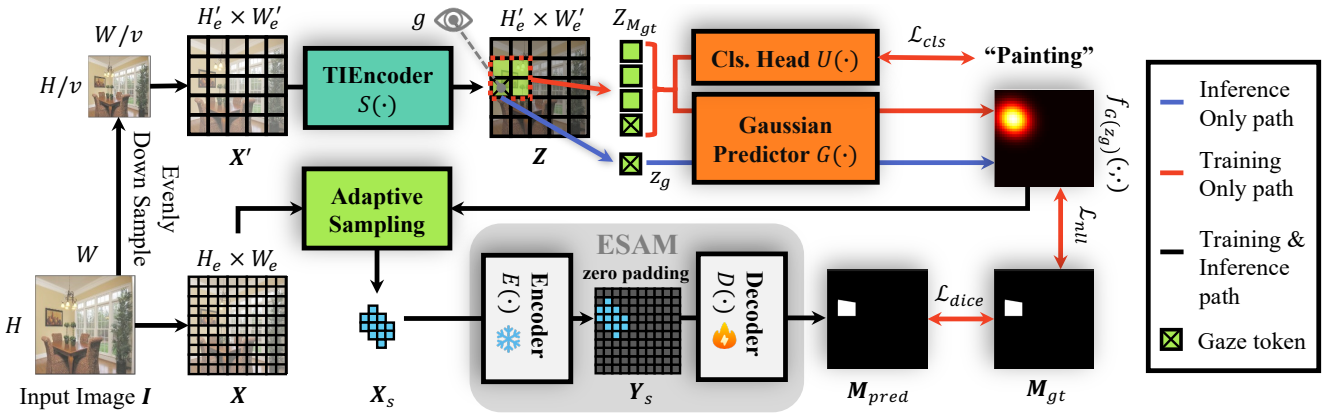


Figure 5. **Inference: (1)** The image I is downsampled and fed into TIEncoder to produce token embedding map Z. **(2)** The token (z_g) closest to the gaze point is selected and passed into Gaussian predictor $G(\cdot)$ to get a Gaussian distribution $f_G(z_g)$. **(3)** $f_G(z_g)$ is then used to adaptively sample tokens X_s from original image. **(4)** X_s are passed to the ESAM encoder to get predicted mask. **Training: (a)** Same with (1). **(b)** Select all tokens ($Z_{M_{gt}}$) in Z that fall within the ground truth segmentation mask M_{gt}. **(c)** Pass $Z_{M_{gt}}$ into Cls. Head $U(\cdot)$ and $G(\cdot)$. **(d)** $G(\cdot)$ is trained to predict Gaussian that can cover M_{gt}, $U(\cdot)$ is trained to classify all $Z_{M_{gt}}$ into the same correct label.

ables adaptive token selection, allowing the number of retained tokens to vary according to the size of the IOI. Let $X_s \subseteq X$ represent the set of tokens selected under equation 1. The selected tokens are passed through the image encoder, yielding $E(X_s)$, which is then fed into the mask decoder $D(.)$ as shown by the low portion of Figure 5. As noted in Section 3.1, $D(.)$ expects a fixed input size of $H_e \times W_e$. To satisfy this requirement, $E(X_s)$ is zero-padded to the target dimensions, the result is denoted as Y_s. The decoder then generates the predicted segmentation mask M_{pred}. The following section details the training procedure.

3.2.3. Training and Loss Function

Figure 5 illustrates the training and inference workflows of FoSAM, with data flows for inference and training shown in black+blue and black+red, respectively. During inference, only the token z_g, which is closest to the gaze location g, is used to predict the TIScore. Let M_{gt} represent the ground-truth binary mask corresponding to the input I, where $M_{gt,ij} = 1$ indicates that the pixel lies within the IOI, and 0 otherwise. Additionally, let $X_{M_{gt}} \subseteq X$ denote the subset of tokens whose spatial positions fall within the IOI as defined by M_{gt}, and let $Z_{M_{gt}} = S(X_{M_{gt}})$ represent the outputs of the $S(.)$ of the corresponding tokens. The loss function used during training comprises three components: the negative log-likelihood, $\mathcal{L}_{\text{nll}}$, the classification loss $\mathcal{L}_{\text{cls}}$, and the Dice loss $\mathcal{L}_{\text{dice}}$. The objective of $\mathcal{L}_{\text{nll}}$ is to train the TIEncoder module $S(\cdot)$ and the Gaussian predictor $G(\cdot)$ to generate a 2D Gaussian map $f_{G(S(x))}(\cdot)$, where $x \in X_{M_{gt}}$, that closely approximates the ground-truth mask M_{gt}, as:

$$\mathcal{L}_{\text{nll}} = -\sum_{x_m \in X_{M_{gt}}} \sum_{x'_m \in X_{M_{gt}}} log[f_{G(S(x_m))}(c_i(x'_m), c_j(x'_m))] \tag{2}$$

where $\sum_{x'_m \in X_{M_{gt}}} log[f_{G(S(x_m))}(c_i(x'_m), c_j(x'_m))]$ represents the sum of the log-probabilities for each token x'_m within the ground truth mask M_{gt}. These probabilities are computed using a 2D Gaussian distribution, where the distribution parameters are generated based on the reference token x_m. In other words, we aim to ensure that each token x_m within the ground truth mask M_{gt} can generate a 2D Gaussian distribution that effectively covers the IOI.

In practice, obtaining an accurate Gaussian distribution that mimics the ground-truth mask poses a granular alignment challenge. For example, when the gaze point lands on the headlight of a car, it is often ambiguous whether the target object is the headlight itself or the entire vehicle. Similarly, if the gaze is fixed on a chess piece, it is unclear whether the segmentation should cover the individual piece or the whole chessboard. To mitigate this ambiguity, we introduce an additional classification loss $\mathcal{L}_{\text{cls}}$, which leverages category labels to enforce granular alignment. Beyond alignment, $\mathcal{L}_{\text{cls}}$ also provides a clustering effect, markedly enhancing the inter-class diversity of the Gaussian maps. As an auxiliary supervision signal, it encourages the TIEncoder $S(.)$ to learn class-specific features, leading to better Gaussian maps. Therefore, as shown in Figure 5, a shallow neural network $U(.)$ is attached to the output of $S(.)$ for classification task. Let $M_{cls}(x_m)$ denote the class label associated with the IOI where x_m is mapping to, $\mathcal{L}_{\text{cls}}$ can be expressed as:

$$\mathcal{L}_{\text{cls}} = -\sum_{x_m \in X_{M_{gt}}} CE[U(S(x_m)), M_{cls}(x_m)] \tag{3}$$

Here, $CE(\cdot)$ denotes the cross-entropy loss. Finally, to ensure that the output M_{pred} of the image decoder $D(\cdot)$ aligns with the ground-truth mask M_{gt}, we apply the Dice loss.

$$\mathcal{L}_{\text{dice}} = DICE[M_{pred}, M_{gt}] \tag{4}$$

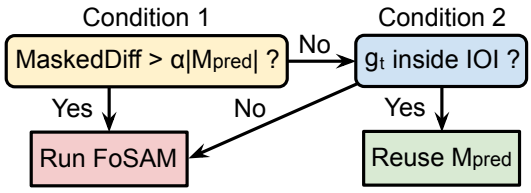


Figure 6. The computational flow of FSA.

The overall loss function can be calculated as $\mathcal{L}_{\text{total}} = \lambda_1\mathcal{L}_{\text{nll}}+\lambda_2\mathcal{L}_{\text{cls}}+\lambda_3\mathcal{L}_{\text{dice}}$, where λ_1,λ_2 and λ_3 represent the relative importance of the loss functions. During training, only the TIEncoder $S(\cdot)$, classification head $U(\cdot)$, Gaussian head $G(\cdot)$, and image decoder $D(\cdot)$ are updated, while the image encoder $E(\cdot)$ remains frozen, significantly reducing the training cost of FoSAM.

3.3. FoSAM Streaming Algorithm

In this section, we describe how to adapt the FoSAM pipeline in Section 3.2 for efficient instance segmentation on streaming video frames. Motivated by the eye-behavior study in Section 2.2, FoSAM is invoked only when the front-facing camera view changes substantially or when the user's gaze direction shifts. Figure 6 summarizes the resulting FoSAM Streaming Algorithm (FSA). Let I_{ini} denote the reference initial frame of the current video fragment and I_t the incoming frame at time step t. To determine whether the current view has drifted beyond the previously segmented state, FSA compares each I_t directly with I_{ini} (Condition 1), rather than relying on differences between consecutive frames. This design avoids the accumulation of small RGB changes over time. While the difference between consecutive frames may appear minor, these changes can accumulate and eventually lead to a significant drift from the previously segmented state.

In addition, to eliminate the influence of irrelevant regions on the reuse decision, we evaluate changes only within the previously predicted segment instance region. Specifically, we use the segmentation mask M_{pred} from I_{ini} and compute differences only within this masked region, rather than performing pixel-wise comparisons over the entire image. Moreover, because the accumulated masked difference scales with the size of the gazed object, we use an area-adaptive threshold by scaling α with the mask area. Concretely, a full FoSAM re-execution is triggered when: $\sum_{i,j}|(I_{t,ij} - I_{\text{ini},ij})M_{\text{pred},ij}| > \alpha\sum_{i,j}M_{\text{pred},ij}$, where M_{pred} is a binary mask and $\sum_{i,j}M_{\text{pred},ij}$ denotes the masked area. If the condition holds, FoSAM is re-run and the reference is refreshed by setting I_{ini} to I_t. Otherwise, FSA checks whether the current gaze location g_t remains inside the IOI region defined by M_{pred} (Condition 2). If so, M_{pred} is reused; if not, FoSAM is executed conditioned on g_t and M_{pred} is updated accordingly. Details of FSA are provided in the supplementary materials. We set the threshold conservatively so that segmentation is not reused under significant head drift, ensuring accuracy during head movements. More efficient streaming approaches that explicitly account for head drift will be explored in future work.

Table 2. IoU and FLOP comparison of FoSAM and baseline algorithms. K denotes the token budget.

Method	K	ADE20K	LVIS	Cityscapes	GFLOP
SAM-B [21]	4096	0.511	0.537	0.347	744
ESAM-S [54]	4096	0.411	0.359	0.304	188.2
ESAM-T [54]	4096	0.350	0.236	0.248	56.1
SlimSAM [8]	4096	0.477	0.474	0.293	15.1
FSNet-DL [58]	NA	0.34	0.38	0.24	18.8
FSNet-SF [58]	NA	0.33	0.36	0.21	12.6
AD-S	200	0.273	0.350	0.221	14.6
LC-S	200	0.368	0.477	0.385	14.6
NU-S	200	0.296	0.349	0.308	14.6
FoSAM-S	200	**0.495**	**0.487**	**0.404**	**14.6**
AD-S	100	0.252	0.343	0.203	10.4
LC-S	100	0.294	0.444	0.339	10.4
NU-S	100	0.215	0.268	0.237	10.4
FoSAM-S	100	**0.465**	**0.461**	**0.362**	**10.4**
AD-T	200	0.256	0.322	0.198	7.4
LC-T	200	0.327	0.453	0.371	7.4
NU-T	200	0.264	0.322	0.283	7.4
FoSAM-T	200	**0.471**	**0.469**	**0.377**	**7.4**
AD-T	100	0.239	0.318	0.176	5.3
LC-T	100	0.264	0.421	0.302	5.3
NU-T	100	0.218	0.241	0.197	5.3
FoSAM-T	100	**0.457**	**0.435**	**0.328**	**5.3**

4. Evaluation

We evaluate FoSAM using publicly available datasets: ADE20K [59], LVIS [15], and Cityscapes [11]. As these datasets are originally designed for generic segmentation tasks to cover all the objects in the image, we introduce a gaze-aware masking preprocessing step to enable segmentation over the IOI only. Specifically, for each training and testing sample, a gaze location is randomly selected within the image, and the corresponding IOI region is defined based on this location. The size of the input images is 640×640 for all datasets.

We use a lightweight two-layer ViT as the TIEncoder $S(.)$. The input images have a resolution of $H \times W = 640 \times 640$, and are average downsampled to $H'_e \times W'_e = 160 \times 160$ prior to processing. The FoSAM framework is integrated with ESAM Small [54] and ESAM Tiny, referred to as FoSAM-S and FoSAM-T, respectively. Three baseline methods are developed for comparison. The first method, *Average Downsampling* (AD), enhances segmentation efficiency by directly applying average pooling to the original input X, reducing its spatial resolution and thus lowering computational cost. The second method, *Local Cropping (LC)*, improves efficiency by extracting a fixed-size patch centered around the gaze point g, thereby narrowing the focus to a smaller input region. We also evalu-

ate a distance-based non-uniform token sampling baseline, denoted as *Non-Uniform (NU)*, by dividing the image into multiple concentric regions centered at the gaze location and using a higher sampling density for patches closer to the gaze while progressively reducing the sampling density for patches farther away. These baselines are evaluated on both ESAM Small and ESAM Tiny, resulting in AD-S, AD-T, LC-S, LC-T, NU-S, and NU-T configurations. Furthermore, we evaluate performance against SAM Base (SAM-B), ESAM Small (ESAM-S), ESAM Tiny (ESAM-T), and SlimSAM [8], which improves segmentation efficiency through a combination of knowledge distillation, embedding pruning, and bottleneck pruning. We also compare FoSAM with FSNet, which integrates DeepLab [7] and Segformer [52] backbones to enable efficient segmentation via learnable input downsampling strategies, referred to as FSNet-DL and FSNet-SF, respectively. During training, the weights for the loss terms $\mathcal{L}_{\text{nll}}$, $\mathcal{L}_{\text{cls}}$, and $\mathcal{L}_{\text{dice}}$ are all set to 1. We evaluate segmentation performance using the Intersection over Union (IoU) metric, which measures the overlap between the predicted segmentation and the ground truth IOI region. To control the average number of retained tokens K, we adjust the TIScore threshold τ and the minimum token count ϵ as defined in Equation 1. More evaluation results are shown in the supplementary materials.

4.1. Evaluation Results of FoSAM

Table 2 presents the evaluation results. The compared methods, FoSAM, all operate on input images at their original resolution. FoSAM adopts an adaptive token sampling strategy with average token budgets of $K = 200$ and $K = 100$ across the entire test dataset. To ensure a fair comparison, the input sizes of AD, LC, and NU are adjusted so that their computational cost aligns with that of FoSAM. For ESAM, SlimSAM, and SAM, regardless of the original input size, the built-in resizing layer by default resizes all inputs to 1024×1024, resulting in a fixed number of 4096 tokens. For SlimSAM, we tune its pruning ratio to achieve a similar computational cost to FoSAM. In the case of FSNet-DL and FSNet-SF, input images are downsampled to a resolution of 80×80, which is its default setting. As shown in the results, FoSAM consistently outperforms the AD, LC, and NU baselines across all datasets. In contrast to different versions of ESAM, SlimSAM and FSNet, FoSAM achieves both higher accuracy and reduced computation. For instance, in Cityscapes datasets, FoSAM-T with 100 token budget achieves 3.5% higher IoU than SlimSAM with only one third of the computation cost. This highlights the effectiveness of IOI guided adaptive token sampling in FoSAM. Compared to SAM, although FoSAM does not achieve equally high IoU, it provides comparable accuracy while reducing computational cost by over $50\times$. As shown in the user study in Section 4.4, FoSAM also offers a significantly better visual experience, whereas SAM introduces noticeable delays that degrade user experience.

To assess FoSAM's processing latency on AR device, we profile the performance of FoSAM-S and FoSAM-T with average token budgets of K=200 and 100 on the Qualcomm XR2 Gen2 platform [38], utilizing the Qualcomm AI Hub toolkit [37]. Qualcomm XR2 Gen2 is integrated into the Meta Quest Pro for AR/VR task processing [12, 18]. The average latencies, computed over 2000 LVIS samples and 100 runs for each sample, are reported in Table 3. Compared to the latency of SAM and ESAM shown in Table 1, FoSAM achieves an average of $24\times$ reduction in inference time and meets the 30 ms latency threshold required for a seamless visual experience [1, 2, 20].

4.2. Results of FoSAM Streaming Algorithm

The FSA mechanism introduced in Section 3.3 reduces segmentation overhead by reusing results from previous frames. However, as illustrated in Figure 6, varying the parameter can influence the average IoU across frames. To evaluate this effect, we analyze the performance of FoSAM-S and FoSAM-T with FSA evaluated on the Cityscapes dataset under different settings of α, which represents the threshold for masked area difference shown in Figure 6. The token budget is set to 200 for both FoSAM-S and FoSAM-T. We use 10 video sequences which contain 5000 images in total. The results are described in Figure 7, as α increases from 0.01 to 0.1, more frames are skipped by reusing segmentation results of previous frames. For instance, skipping 40% of the frames leads to only a 0.03 reduction in average IoU across frames, showing that FSA can greatly reduce computation with negligible degradation.

4.3. Ablation Study

Impact of TIEncoder Size: In this section, we examine how the computational cost of the TIEncoder $S(\cdot)$ influences FoSAM's accuracy under an average token budget of $K = 200$ on the ADE20K dataset. Specifically, we evaluate the impact on FoSAM performance by varying both the parameter size of $S(\cdot)$ and the input resolution of X' fed into $S(\cdot)$. As shown in Table 5, increasing the number of encoder layers from 2 to 3 and raising the input resolution from 160×160 to 640×640 leads to only a slight improvement in IoU, while the computational cost rises substantially from 0.48 GFLOPs to 10.4 GFLOPs

Table 5. Impact of the TIEncoder. LN denotes layer number

LN	Image Size	IoU	GFLOP
3	640×640	0.499	10.4
3	320×320	0.499	2.8
2	320×320	0.497	1.88
2	160×160	0.495	0.48
2	80×80	0.492	0.12
1	320×320	0.492	0.94

Latency (ms)	Token 200	Token 100
FoSAM-S	22.7	17.9
FoSAM-T	19.3	15.7

Table 3. Processing latencies of FoSAM on an AR platform.

Method	LVIS
Gaussian (ours)	0.487
Token-wise	0.231

Table 4. Gaussian versus Token-wise with 200 token budget.

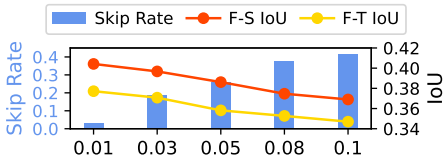


Figure 7. Average IoU and frame skip rate of FSA on FoSAM-S and FoSAM-T under varying α.



Figure 8. Participants are doing user study.

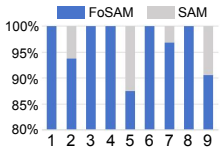


Figure 9. Preference rates of the participants.

Method	K	AEA IoU	GFLOP
ESAM-S	4096	0.321	188.2
ESAM-T	4096	0.306	56.1
AD-S	400	0.133	19.8
LC-S	400	0.496	19.8
FoSAM-S	**400**	**0.588**	**19.8**
FoSAM-S	**200**	**0.555**	**14.6**

Table 6. Experiment on AEA.

for the TIEncoder processing. Additionally, reducing the number of encoder layers results in a more significant accuracy drop than downsampling the input image. The TIEncoder design in FoSAM strikes an effective balance between efficiency and performance.

TIEncoder design: Gaussian vs. Token-wise: Selecting tokens based on scores is a common concept in computer vision, but most of them predict a score for each token separately, as described in Section 3.2.1. In contrast, FoSAM predicts a single Gaussian distribution and uses it to assign all TIScores at once. Table 4 demonstrates the effect of switching to a common token-wise design in our case by adding Importance Head for each token. The token-wise approach performs significantly worse under the same token budget, because unlike the smooth and concentrated Gaussian TIScore map, the token-wise design produces fragmented and discontinuous token selection, and SAM's mask decoder is vulnerable to this discontinuity.

4.4. User Study

To assess the user experience benefits of FoSAM over the standard SAM method on AR device, we conduct a two-interval forced-choice (2IFC) [57] user study simulating both methods' effects on the Meta Quest Pro headset [18]. For both methods, their corresponding segmentation masks were precomputed and gaze-contingent visualizations were rendered with artificially introduced latencies reflecting each method's runtime profiled on the Qualcomm XR2 Gen2's NPU [36]. Gaze was tracked live from each participant via the headset's gaze tracker. Each of the nine participants compared results across four test images, selecting the preferred output in 32 trials, as shown in Figure 8. As shown in Figure 9, FoSAM was preferred in 96.5%±4.8% of trials overall, consistently outperforming SAM across all images. Scene-level preferences were 97.2%, 95.8%, 96.1%, and 96.9% for the four images. Under the null hypothesis of no preference ($p = 0.5$), across $N = 288$ trials participants chose FoSAM in $k = 278$ cases, yielding a two-sided binomial test against the 50% chance level ($p \ll 0.001$), indicating a statistically significant preference for FoSAM.

5. Discussion

Realistic Gaze Analysis: One potential limitation of FoSAM is that its performance can be affected by imperfections in the gaze input. To evaluate FoSAM with realistic gaze input, we further evaluate it on Aria Everyday Activities (AEA), a real-world AR dataset with gaze traces, using the same evaluation protocol as in Section 4. As shown in Table 6, FoSAM achieves a superior accuracy-computation tradeoff compared with the baselines. We further evaluate the impact of realistic gaze tracking errors, gaze latency, and gaze location ambiguity in the supplementary material.

Comparison to prior gaze-guided token masking method: Prior work [22] uses gaze trajectories to select image patches according to fixed selection rules. In contrast, the Gaussian predictor in FoSAM adapts token selection to the spatial extent of the gaze-selected object. Compared with the Local Cropping (LC) baseline in Table 2, which follows the principle of [22], FoSAM achieves higher accuracy under the same computation through more accurate, object-adaptive token selection. In addition, the Gaussian produces a spatially concentrated region for the subsequent SAM encoder, avoiding the degradation from independently selected tokens shown in Table 4.

6. Conclusion

In this paper, we presented *FoSAM*, a gaze-guided segmentation framework designed to reduce the computational cost of SAM for AR/VR devices. By focusing segmentation on the user's instance of interest (IOI), FoSAM reduces unnecessary processing while preserving segmentation quality. We further introduce the *FoSAM Streaming Algorithm (FSA)* to exploit temporal continuity across frames. FoSAM achieves over $50\times$ reduction in computation while maintaining competitive accuracy, enabling real-time deployment on AR platforms and better user experience. User studies further demonstrate that the reduced latency leads to a significantly improved user experience.

References

[1] Understanding vr performance metrics. https://moldstud.com/articles/p-understanding-vr-performance-metrics-for-developers. 3, 7

[2] Rachel Albert, Anjul Patney, David Luebke, and Joohwan Kim. Latency requirements for foveated rendering in virtual reality. ACM Trans. Appl. Percept., 14(4), 2017. 3, 7

[3] Hassan Abu Alhaija, Siva Karthik Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. Augmented reality meets deep learning for car instance segmentation in urban scenes. In British machine vision conference, 2017. 1

[4] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39:2481–2495, 2015. 1, 2

[5] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In Proceedings of the IEEE/CVF international conference on computer vision, pages 9157–9166, 2019. 1

[6] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. ArXiv, abs/1708.02551, 2017. 2

[7] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation, 2017. 2, 7

[8] Zigeng Chen, Gongfan Fang, Xinyin Ma, and Xinchao Wang. Slimsam: 0.1% data makes segment anything slim. In Advances in Neural Information Processing Systems, pages 39434–39461. Curran Associates, Inc., 2024. 6, 7

[9] Bowen Cheng, Maxwell D. Collins, Yukun Zhu, Ting Liu, Thomas S. Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 12472–12482, 2019. 2

[10] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. In Neural Information Processing Systems, 2021. 2

[11] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. 6

[12] Jakob Engel, Kiran Somasundaram, Michael Goesele, Albert Sun, Alexander Gamino, Andrew Turner, Arjang Talattof, Arnie Yuan, Bilal Souti, Brighid Meredith, et al. Project aria: A new tool for egocentric multi-modal ai research. arXiv preprint arXiv:2308.13561, 2023. 3, 7

[13] Santiago González Izard, Juan A Juanes Méndez, Pablo Ruisoto Palomera, and Francisco J García-Peñalvo. Applications of virtual and augmented reality in biomedical imaging. Journal of medical systems, 43(4):102, 2019. 1

[14] Santiago González Izard, Ramiro Sánchez Torres, Oscar Alonso Plaza, Juan Antonio Juanes Mendez, and Francisco José García-Peñalvo. Nextmed: automatic imaging segmentation, 3d reconstruction, and 3d model visualization platform using augmented and virtual reality. Sensors, 20 (10):2962, 2020. 1

[15] Agrim Gupta, Piotr Dollár, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation, 2019. 6

[16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In Proceedings of the IEEE international conference on computer vision, pages 2961–2969, 2017. 1, 2

[17] Baosheng James Hou, Yasmeen Abdrabou, Florian Weidner, and Hans Gellersen. Unveiling variations: A comparative study of vr headsets regarding eye tracking volume, gaze accuracy, and precision. In 2024 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW), pages 650–655. IEEE, 2024. 4

[18] Meta Platform Inc. Meta quest pro. https://www.meta.com/quest/quest-pro/, 2022. 3, 7, 8

[19] Chen Jin, Ryutaro Tanno, Thomy Mertzanidou, Eleftheria Panagiotaki, and Daniel C Alexander. Learning to downsample for segmentation of ultra-high resolution images. arXiv preprint arXiv:2109.11071, 2021. 2

[20] Topi Kaaresoja, Stephen Brewster, and Vuokko Lantz. Towards the temporally perfect virtual button: touch-feedback simultaneity and perceived quality in mobile touchscreen press interactions. ACM Transactions on Applied Perception (TAP), 11(2):1–25, 2014. 3, 7

[21] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 4015–4026, 2023. 1, 2, 3, 6

[22] Jacqueline Kockwelp, Daniel Beckmann, and Benjamin Risse. Human gaze improves vision transformers by token masking. In Proceedings of the Winter Conference on Applications of Computer Vision, pages 396–405, 2025. 8

[23] Bart Krekelberg. Saccadic suppression. Current Biology, 20 (5):R228–R229, 2010. 3

[24] Xiangtai Li, Henghui Ding, Wenwei Zhang, Haobo Yuan, Jiangmiao Pang, Guangliang Cheng, Kai Chen, Ziwei Liu, and Chen Change Loy. Transformer-based visual segmentation: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 46:10138–10163, 2023. 2

[25] Yanwei Li, Hengshuang Zhao, Xiaojuan Qi, Liwei Wang, Zeming Li, Jian Sun, and Jiaya Jia. Fully convolutional networks for panoptic segmentation. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 214–223, 2020. 2

[26] Huaijia Lin, Ruizheng Wu, Shu Liu, Jiangbo Lu, and Jiaya Jia. Video instance segmentation with a propose-reduce paradigm. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 1739–1748, 2021. 2

[27] Nian Liu, Ni Zhang, Kaiyuan Wan, Ling Shao, and Junwei Han. Visual saliency transformer. In Proceedings of

the IEEE/CVF international conference on computer vision, pages 4722–4732, 2021. 4

[28] Tianpeng Liu, Jing Li, Jia Wu, Lefei Zhang, Jun Chang, Jun Wan, and Lezhi Lian. Tracking with saliency region transformer. IEEE Transactions on Image Processing, 33:285–296, 2023. 4

[29] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3431–3440, 2015. 1, 2

[30] Zhaoyang Lv, Nicholas Charron, Pierre Moulon, Alexander Gamino, Cheng Peng, Chris Sweeney, Edward Miller, Huixuan Tang, Jeff Meissner, Jing Dong, Kiran Somasundaram, Luis Pesqueira, Mark Schwesinger, Omkar Parkhi, Qiao Gu, Renzo De Nardi, Shangyi Cheng, Steve Saarinen, Vijay Baiyya, Yuyang Zou, Richard Newcombe, Jakob Julian Engel, Xiaqing Pan, and Carl Ren. Aria everyday activities dataset, 2024. 3

[31] Dmitrii Marin, Zijian He, Peter Vajda, Priyam Chatterjee, Sam Tsai, Fei Yang, and Yuri Boykov. Efficient segmentation: Learning downsampling near semantic boundaries. In Proceedings of the IEEE/CVF international conference on computer vision, pages 2131–2141, 2019. 2

[32] Ethel Matin. Saccadic suppression: a review and an analysis. Psychological bulletin, 81(12):899, 1974. 3

[33] Shervin Minaee, Yuri Boykov, Fatih Murat Porikli, Antonio J. Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44:3523–3542, 2020. 2

[34] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 8829–8837, 2019. 2

[35] 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

[36] Inc. Qualcomm Technologies. Qualcomm hexagon npu, 2025. Accessed: 2025-05-08. 8

[37] Qualcomm Technologies, Inc. Qualcomm ai hub. https://app.aihub.qualcomm.com/docs/index.html, 2024. Accessed: 2025-04-19. 3, 7

[38] Qualcomm Technologies, Inc. Snapdragon xr2 gen 2 platform. https://www.qualcomm.com/products/mobile/snapdragon/xr-vr-ar/snapdragon-xr2-gen-2-platform, 2024. Accessed: 2025-04-19. 3, 7

[39] Frano Rajič, Lei Ke, Yu-Wing Tai, Chi-Keung Tang, Martin Danelljan, and Fisher Yu. Segment anything meets point tracking. arXiv preprint arXiv:2307.01197, 2023. 2

[40] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. Advances in neural information processing systems, 34:13937–13949, 2021. 4

[41] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. arXiv preprint arXiv:2408.00714, 2024. 1

[42] Adria Recasens, Petr Kellnhofer, Simon Stent, Wojciech Matusik, and Antonio Torralba. Learning to zoom: a saliency-based sampling layer for neural networks. In Proceedings of the European conference on computer vision (ECCV), pages 51–66, 2018. 2

[43] Tanner Schmidt and Richard Newcombe. Segment this thing: Foveated tokenization for efficient point-prompted segmentation. In 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 29428–29437. IEEE, 2025. 2

[44] Niklas Stein, Diederick C Niehorster, Tamara Watson, Frank Steinicke, Katharina Rifai, Siegfried Wahl, and Markus Lappe. A comparison of eye tracking latencies among several commercial head-mounted displays. i-Perception, 12(1): 2041669520983338, 2021. 4

[45] Yehui Tang, Kai Han, Yunhe Wang, Chang Xu, Jianyuan Guo, Chao Xu, and Dacheng Tao. Patch slimming for efficient vision transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 12165–12174, 2022. 4

[46] Leonardo Tanzi, Pietro Piazzolla, Francesco Porpiglia, and Enrico Vezzetti. Real-time deep learning semantic segmentation during intra-operative surgery for 3d augmented reality assistance. International Journal of Computer Assisted Radiology and Surgery, 16(9):1435–1445, 2021. 1

[47] Chittesh Thavamani, Mengtian Li, Nicolas Cebron, and Deva Ramanan. Fovea: Foveated image magnification for autonomous navigation. In Proceedings of the IEEE/CVF international conference on computer vision, pages 15539–15548, 2021. 2

[48] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herv'e J'egou. Training data-efficient image transformers & distillation through attention. In International Conference on Machine Learning, 2020. 2

[49] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and fast instance segmentation. Advances in Neural information processing systems, 33:17721–17732, 2020. 1

[50] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 8741–8750, 2021. 2

[51] Junfeng Wu, Yi Jiang, Song Bai, Wenqing Zhang, and Xiang Bai. Seqformer: Sequential transformer for video instance segmentation. In European Conference on Computer Vision, pages 553–569. Springer, 2022. 2

[52] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. Advances in neural information processing systems, 34: 12077–12090, 2021. 1, 7

[53] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. Upsnet: A unified panoptic segmentation network. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 8810–8818, 2019. 2

[54] Yunyang Xiong, Bala Varadarajan, Lemeng Wu, Xiaoyu Xiang, Fanyi Xiao, Chenchen Zhu, Xiaoliang Dai, Dilin Wang, Fei Sun, Forrest Iandola, et al. Efficientsam: Leveraged masked image pretraining for efficient segment anything. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 16111–16121, 2024. 2, 3, 6

[55] Bin Yan, Yi Jiang, Jiannan Wu, Dong Wang, Ping Luo, Zehuan Yuan, and Huchuan Lu. Universal instance perception as object discovery and retrieval. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 15325–15336, 2023. 2

[56] Linjie Yang, Yuchen Fan, and Ning Xu. Video instance segmentation. In Proceedings of the IEEE/CVF international conference on computer vision, pages 5188–5197, 2019. 1, 2

[57] Yaffa Yeshurun, Marisa Carrasco, and Laurence T Maloney. Bias and sensitivity in two-interval forced choice procedures: Tests of the difference model. Vision research, 48(17):1837–1851, 2008. 8

[58] Hongyi Zeng, Wenxuan Liu, Tianhua Xia, Jinhui Chen, Ziyun Li, and Sai Qian Zhang. Foveated instance segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 24496–24505, 2025. 2, 6

[59] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. International Journal of Computer Vision, 127(3):302–321, 2019. 6