Supplementary Material

A. Detailed Derivation.

Here, we present the detailed derivation of time complexity for the decoder to demonstrate the effectiveness of our query pruning strategy. GroundingDINO iteratively processes a large number of query embeddings, resulting in significant computational bottlenecks. As GrounddingDINO is based on Deformable DETR [48], its decoder's computations are mainly occupied by the multi-head self-attention (MHSA) operations and feed-forward layers. MHSA is defined as follows:

$$\mathrm{MHSA}(\mathbf{X}) = \sum_{i=1}^{H} \mathbf{W}_i^o \left[\mathrm{softmax}\left(\mathbf{Q}_i \mathbf{K}_i^T\right) \mathbf{V}_i\right], \quad (7)$$

$$\text{and} \quad \mathbf{Q_i} = \mathbf{X}\mathbf{W}_i^q, \quad \mathbf{K}_i = \mathbf{X}\mathbf{W}_i^k, \quad \mathbf{V}_i = \mathbf{X}\mathbf{W}_i^v, \quad (8)$$

where $\mathbf{X} \in \mathbb{R}^{N\times d}$ indicates the query embeddings, H is the head number, and $\mathbf{W}_i^q$, $\mathbf{W}_i^k$, $\mathbf{W}_i^v$, $\mathbf{W}_i^o \in \mathbb{R}^{d\times d}$ are the linear weights of i-th head. The time complexity of each layer is $O(N^2d + Nd^2)$, hence the total time complexity for a L-layer decoder is $O\left(L(Nd^2 + N^2d)\right)$. Then, if we employ the *confidence-aware query pruning strategy* to identify and retain only $\frac{1}{k}$ queries at each layer, the total time complexity becomes:

$$O\left(\left[1 + \frac{1}{k^2} + \frac{1}{k^4} + \cdots + \frac{1}{k^{2(L-1)}}\right] N^2 d + \left[1 + \frac{1}{k} + \frac{1}{k^2} + \cdots + \frac{1}{k^{L-1}}\right] Nd^2\right) \quad (9)$$

$$= O\left(\frac{k^2}{k^2-1}(1 - \frac{1}{k^{2L}})N^2 d + \frac{k}{k-1}(1 - \frac{1}{k^L})Nd^2\right) \quad (10)$$

$$= O\left(\frac{k^2}{k^2-1}N^2 d + \frac{k}{k-1}Nd^2\right), \quad (11)$$

which is independent of the decoder depth L. Note that in equation 10, the items $1 - \frac{1}{k^L}$ and $1 - \frac{1}{k^{2L}}$ can be omitted because $\frac{1}{k^{2L}}$ and $\frac{1}{k^L}$ are typically much smaller than 1 when $k > 1$ and L is a relatively large positive integer.

B. Additional Implementation Details.

FPN. We follow the prior FPN structure [3, 23], which consists of several 2D convolution, GroupNorm and ReLU Layers. Nearest neighbor interpolation is used for upsampling steps.

Implementation Details. The coefficients for losses are set as $\lambda_{cls} = 4$, $\lambda_{L1} = 5$, $\lambda_{giou} = 2$, $\lambda_{dice} = 5$, $\lambda_{focal} = 5$, $\lambda_{proj} = 5$. We apply image augmentation techniques such as flipping, rotation, cropping, and scaling during training, following previous works [25, 40]. The model is pretrained on Ref-COCO/g/+ for 20 epochs with the batch size of 8 for Swin-T and 10 for Swin-B. Then we individually train the models on Ref-YouTube-VOS for 2 epochs and A2D-Sentence for 6 epochs. For MeViS, we train the models for 15 epochs. The number of sampling frames is 6. We use AdamW as the optimizer with the weight decay of 1e-4 and initial learning rate of 5e-5, which is linearly rescaled with the batch size. All experiments are conducted on 8 NVIDIA A800 GPUs.

C. Comparisons with Vanilla Baselines.

As mentioned in Section 2, some previous works [1, 13] also attempted to employ GroundingDINO for RVOS task. Grounded-SAM2 [1] extracts object regions with GroundingDINO and produces masks with SAM2 [28]. Based on this, AL-Ref-SAM2 [13] further incorporates GPT4 [2] to select key frames and boxes. However, such a manner of model ensemble is not end-to-end differentiable, preventing further refinement of RVOS-specific capability. As shown in Table 6, our ReferDINO outperforms these ensemble methods by large margins on MeViS dataset, the largest RVOS benchmark. These results demonstrate the advantages of our end-to-end adaptation approach.

Method	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$	FPS
Video-Swin-T / Swin-T				
Grounded-SAM2 [1]	37.4	31.0	43.7	6.1
ReferDINO (ours)	**48.0**	**43.6**	**52.3**	**28.0**
Video-Swin-B / Swin-B				
Grounded-SAM2 [1]	40.5	34.5	46.4	6.1
AL-Ref-SAM 2 [13]	42.8	39.5	46.2	< 6.1
ReferDINO (ours)	**49.3**	**44.7**	**53.9**	**26.6**

Table 6. Performance comparison on MeViS.

D. Additional Latency Comparisons.

In Table 7, we compare with SOTA dynamic-head methods and our baseline on Ref-Youtube-VOS. The results show that our mask decoder reduces memory usage by 11.2G, enabling superior performance over SOTAs at comparable or faster speed.

Model	$\mathcal{J}\&\mathcal{F}\uparrow$	Memory$\downarrow$	FPS$\uparrow$
SgMg [25]	62.0	16.5G	50.3
MUTR [42]	64.0	34.0G	41.4
G-DINO+DH	64.2	25.3G	50.2
ReferDINO (Ours)	**67.5**	**14.1G**	**51.0**

Table 7. Comparison of training memory and inference speed.

E. Additional Ablation Studies.

Momentum Coefficient. The momentum coefficient α in our tracker controls the amplitude of memory updating. We visualize a case in Figure 7 to show its impact on temporal consistency. In this case, a smaller α (i.e., long-term memory) yields the best performance. This is because the identification in the initial frames is crucial as it captures the clue "*started at the back*". While we set $\alpha = 0.1$ by default in main experiments, it is promising to explore an α-adaptive strategy in future work. In our main experiments, we set $\alpha = 0.1$ by default.

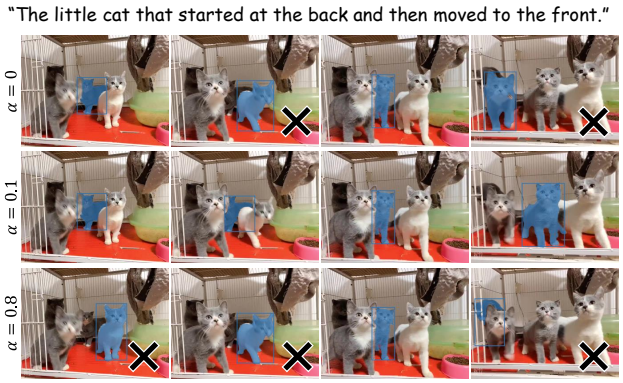

Figure 7. Qualitative impacts of α in memory-augmented tracker. We use ✕ to highlight the incorrect results.

F. More Visualizations.

We provide more visualizations of diverse objects in Figure 8 to demonstrate the robustness of our model.

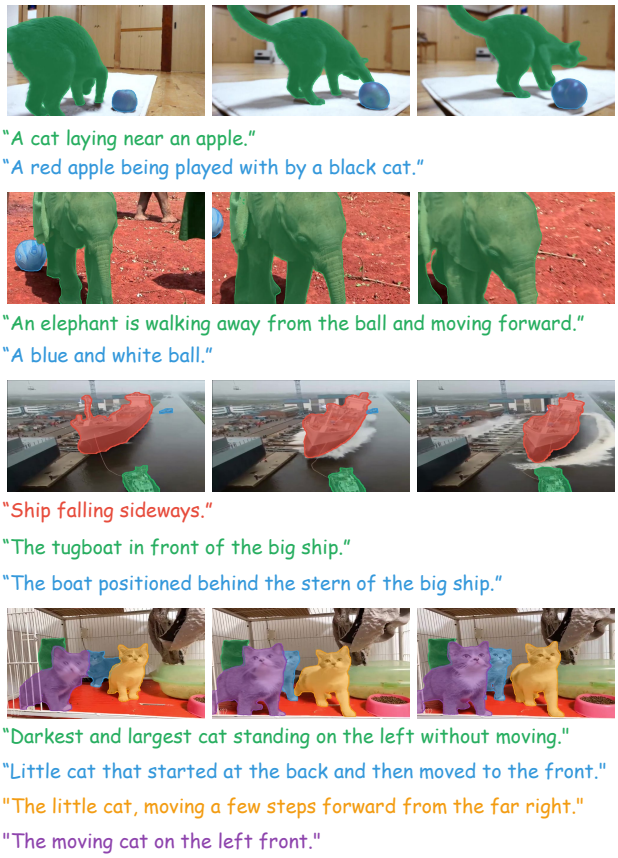

Figure 8. Visualization of our ReferDINO for multiple text references.