Temporal Cluster Assignment for Efficient Real-Time Video Segmentation Supplementary Material

A. Preliminary Extension to ViT-Based Depth Estimation

Method	RMSE↓	$\text{RMSE}_{\log}$↓	AbsRel↓	SqRel↓	SILog↓	$\log_{10}$↓	FPS↑
Baseline (DPT)	2.573	0.0922	0.0624	0.222	8.284	0.0271	30.59
Expedit	2.745	0.0962	0.0638	0.242	8.687	0.0278	35.63
+Ours	**2.682**	**0.0948**	**0.0631**	**0.235**	**8.545**	**0.0275**	36.34

Table 1. Preliminary depth estimation results using a ViT-style (DPT) model with and without TCA.

To explore whether TCA can generalize beyond window-based segmentation transformers like Swin, we conduct a preliminary experiment on a video depth estimation task with the KITTI dataset [2] using DPT [4], which uses a ViT-style backbone with global attention.

As shown in Table 1, integrating TCA into Expedit leads to an improvement in performance across standard depth metrics, alongside a slight increase in FPS. This result is not meant to establish comprehensive generalization, but it provides early evidence that the core idea of TCA—leveraging temporal consistency to refine token usage—may also be applicable to other dense prediction tasks involving ViT-like models. A more systematic investigation is left to future work.

B. Pseudo Code

Algorithm 1 Temporal Cluster Assignment

```
def forward(x, x_ref, α, β, frame_idx, f_max, d):
    '''
    x: current frame clustered tokens
    x_ref: reference frame tokens
    alpha: clustering location
    beta: delayed clustering location
    frame_idx: current frame index
    f_max: keyframe interval
    d: refinement switching parameter
    '''
    key_frame = (frame_idx % f_max == 0)
    if key_frame:
        for i, block in enumerate(blocks):
            # Late clustering
            if i == α + β:
                # Update reference tokens
                x_ref = x
                x = clustering(x)
            x = block(x)
    else:
        for i, block in enumerate(blocks):
            # Early clustering
            if i == α:
                x = clustering(x)
            elif i == α + β:
                # Temporal Cluster
                    Assignment
                # Eq. 1
                distance = distance_fn(x, x
                    ref)
                j = argmin(distance)
                if α <= d:
                    # Eq. 3
                    x = RBS(x, x_ref, j)
                else:
                    # Eq. 2
                    x = CGA(x, x_ref, j)
            x = block(x)
    x = reconstruct(x)
    frame_idx += 1
    return x, x_ref, frame_idx
```

C. Discarded Ideas

C.1. Dynamic Keyframe Interval

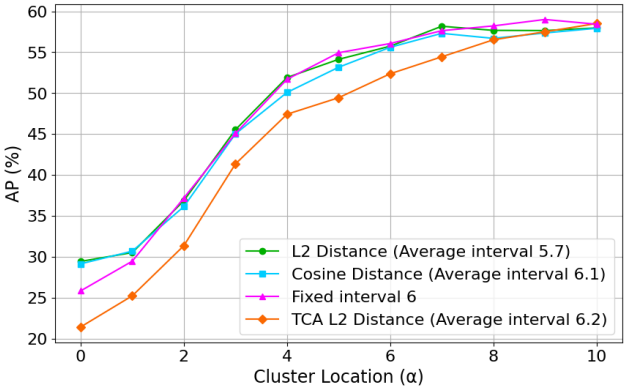

Figure 1. Comparison of dynamic keyframe interval schematic against fixed keyframe interval.

Past work in feature propagation [3, 5, 6] has proposed dynamic keyframe intervals based on a learned module. Given the similarity between our TCA and feature propagation, we also explored the possibility of a training-free dynamic keyframe interval based on existing properties of feature tokens. Specifically, we compute a similarity between current non-keyframe to the reference keyframe. This similarity score accumulates at each non-keyframe, and if the accumulated distance exceeds a predefined threshold, the next frame is considered as a keyframe. Specifically, we tested three different similarity functions:

1) The average distance during the TCA assignment stage, as defined in the main text Eq. 1.

2) The cosine distance between reference frame tokens and current frame tokens, computed at the end of Stage 3 in the Swin Transformer.

3) The L2 distance between reference frame tokens and current frame tokens, also computed at the end of Stage 3 in the Swin Transformer.

Fig. 1 presents a comparison of all three dynamic interval methods against a fixed keyframe interval. While both cosine distance and L2 distance-based dynamic intervals show a slight advantage at $\alpha = 0$ and 1, none of the methods yield a noticeable improvement at later α values. Given the computational overhead required for distance calculations, we opted for a fixed keyframe interval instead.

C.2. Memory Module

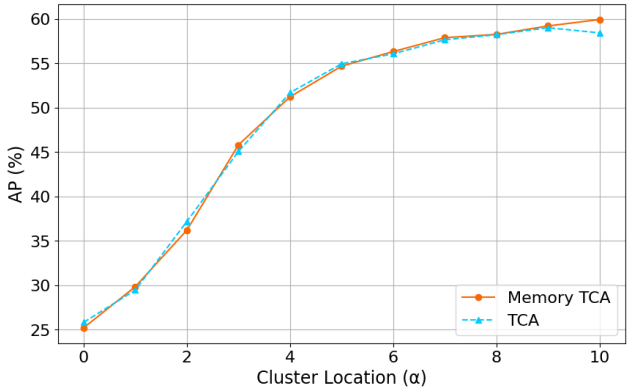

Figure 2. Effect of adding a memory module to TCA compared to using TCA alone.

Inspired by recent methods in memory consolidation for long-context video understanding [1], we explored the addition of a non-parametric memory module during TCA. Specifically, we initialize a set of memory tokens x_{mem} using the refined cluster $x_{\text{cluster}}^{\text{refined}} \in \mathbb{R}^{K\times N\times L}$ at the first non-keyframe, where K is the number of windows, N is the number of clustered tokens, and L is the token dimension.

During TCA in subsequent frames, and before refinement, this memory is concatenated with the reference tokens along the window dimension, forming $x_{\text{ref}}^{\text{mem}} \in \mathbb{R}^{K\times(M+N)\times L}$, where M represents the number of reference tokens. The concatenated tokens are then treated as the new reference tokens and used for assignment and refinement, following the original TCA process. Similar to x_{ref}, x_{mem} are assigned and used to refine x_{cluster}. By taking into account features from past frames stored within, x_{mem} acts as a correction to feature drifting caused by differences between the reference and current frames, mitigating the impact from scene changes in video.

In [1], the memory is updated by concatenating a k-means clustered form of its current token to the existing memory. Since both the k-means operation and the increase in memory size across frames cause extra overhead in TCA, we instead update the memory by directly replacing it with the newly refined tokens $x_{\text{cluster}}^{\text{refined}}$ after the refinement step. This serves as a variant to [1] where the previous memory is included into the newly updated memory through the assignment and refinement process.

Fig. 2 illustrates the effect of incorporating the memory module. While a slight advantage is observed at $\alpha = 10$, the memory module does not yield noticeable improvements at other values of α. Given the additional computational overhead introduced by the increased concatenated reference size, we opt to use only the reference tokens in TCA.

Table 2. Setting of cluster size N and cluster location α used in main results of Table 1 and 2.

Config	Method	YTVIS19		YTVIS21		OVIS	
		Cluster Size (N)	α	Cluster Size (N)	α	Cluster Size (N)	α
Baseline (Unclustered size M)		12×12	/	12×12	/	12×12	/
Highest	AiluRus	7×7	10	8×8	10	8×8	10
	+Ours	7×7	9	8×8	9	8×8	9
	Expedit	8×8	10	8×8	10	8×8	9
	+Ours	7×7	10	8×8	9	6×6	9
30% Speed Up	AiluRus	8×8	7	6×6	8	6×6	9
	+Ours	6×6	7	6×6	6	7×7	6
	Expedit	6×6	7	6×6	8	6×6	8
	+Ours	6×6	6	6×6	6	6×6	7
50% Speed Up	AiluRus	6×6	4	6×6	4	6×6	5
	+Ours	6×6	0	5×5	3	6×6	3
	Expedit	6×6	3	6×6	3	6×6	3
	+Ours	4×4	4	4×4	4	4×4	4

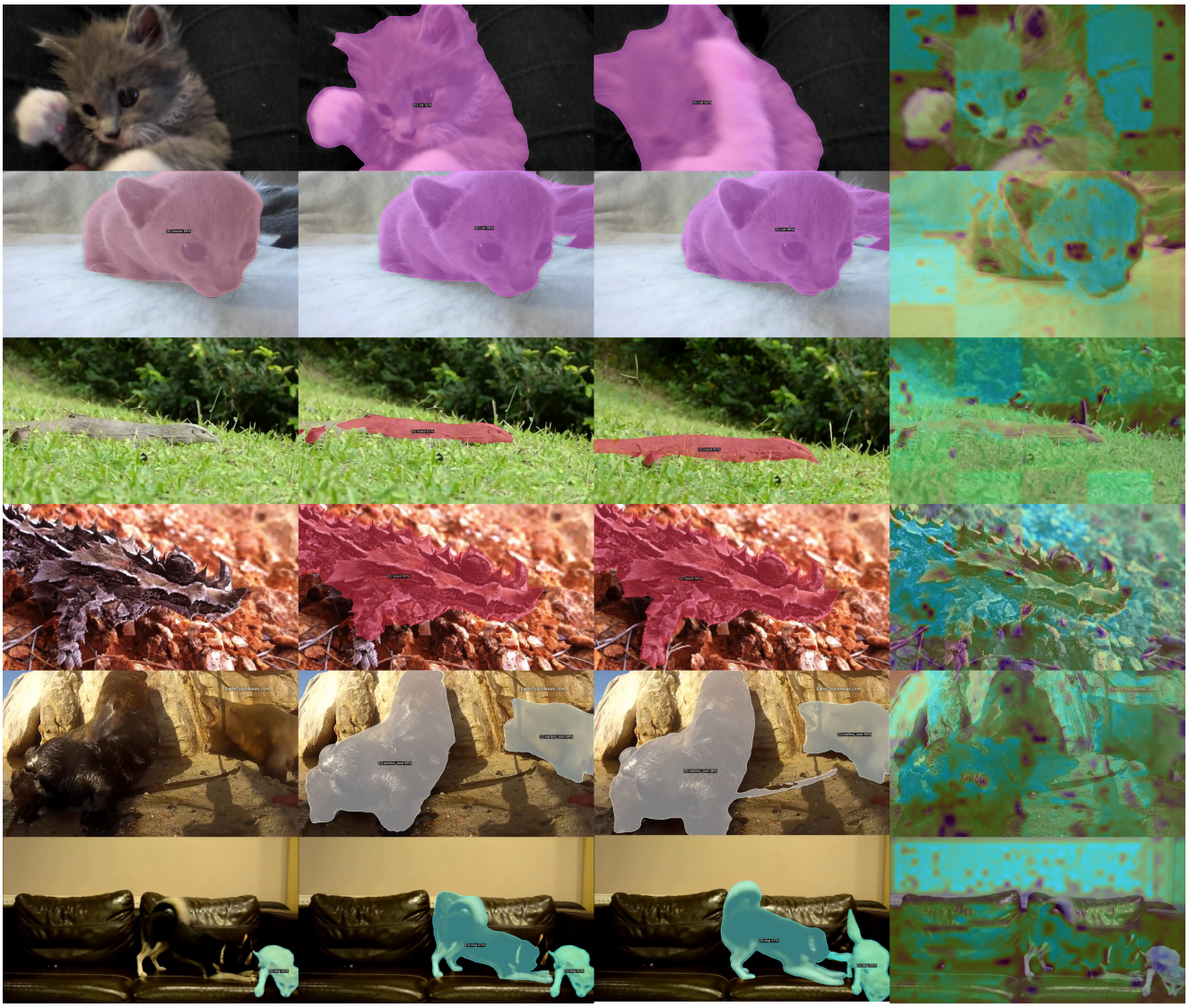

Figure 3. Further examples of comparisons and heatmaps from YTVIS19

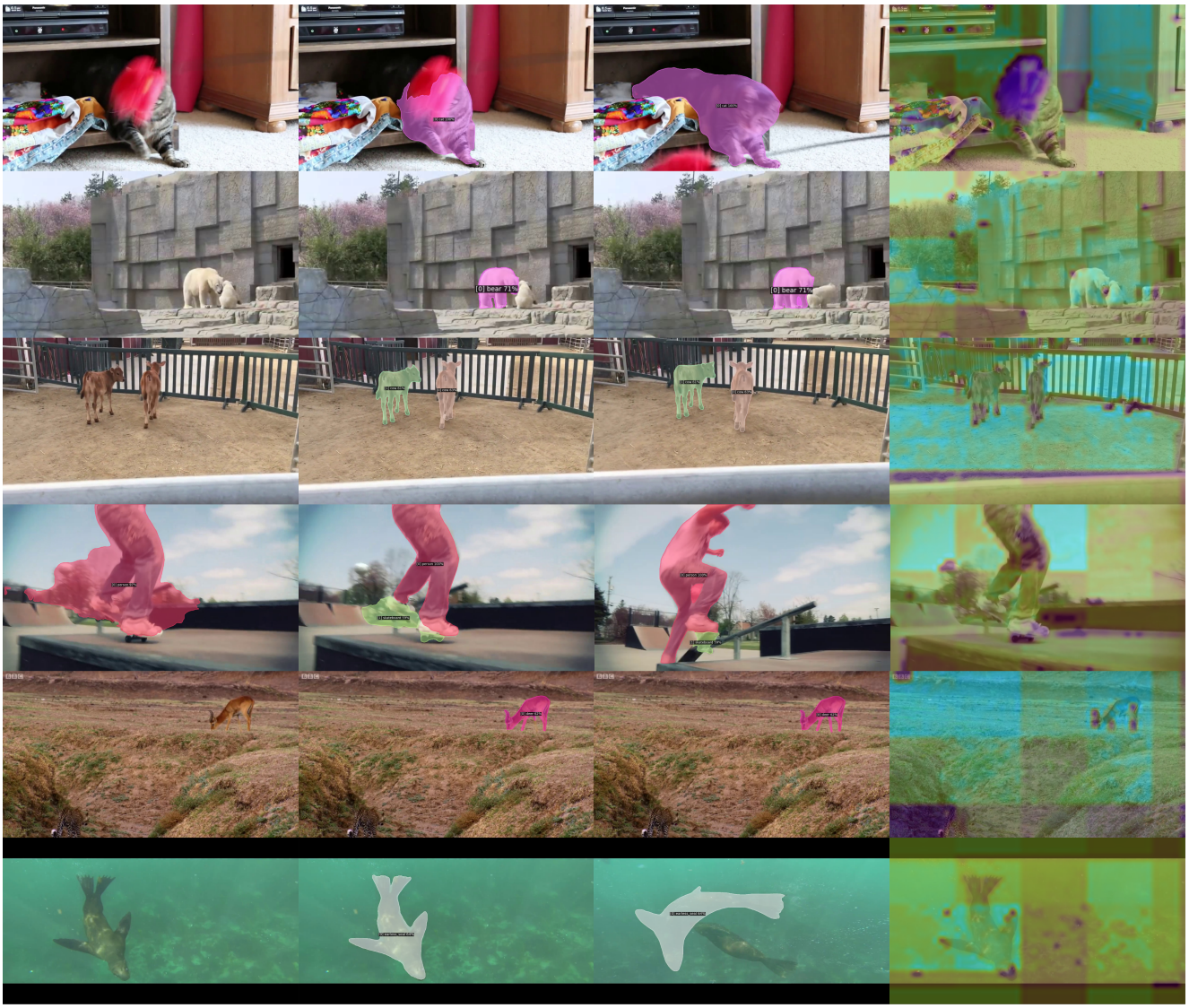

Figure 4. Further examples of comparisons and heatmaps from YTVIS21

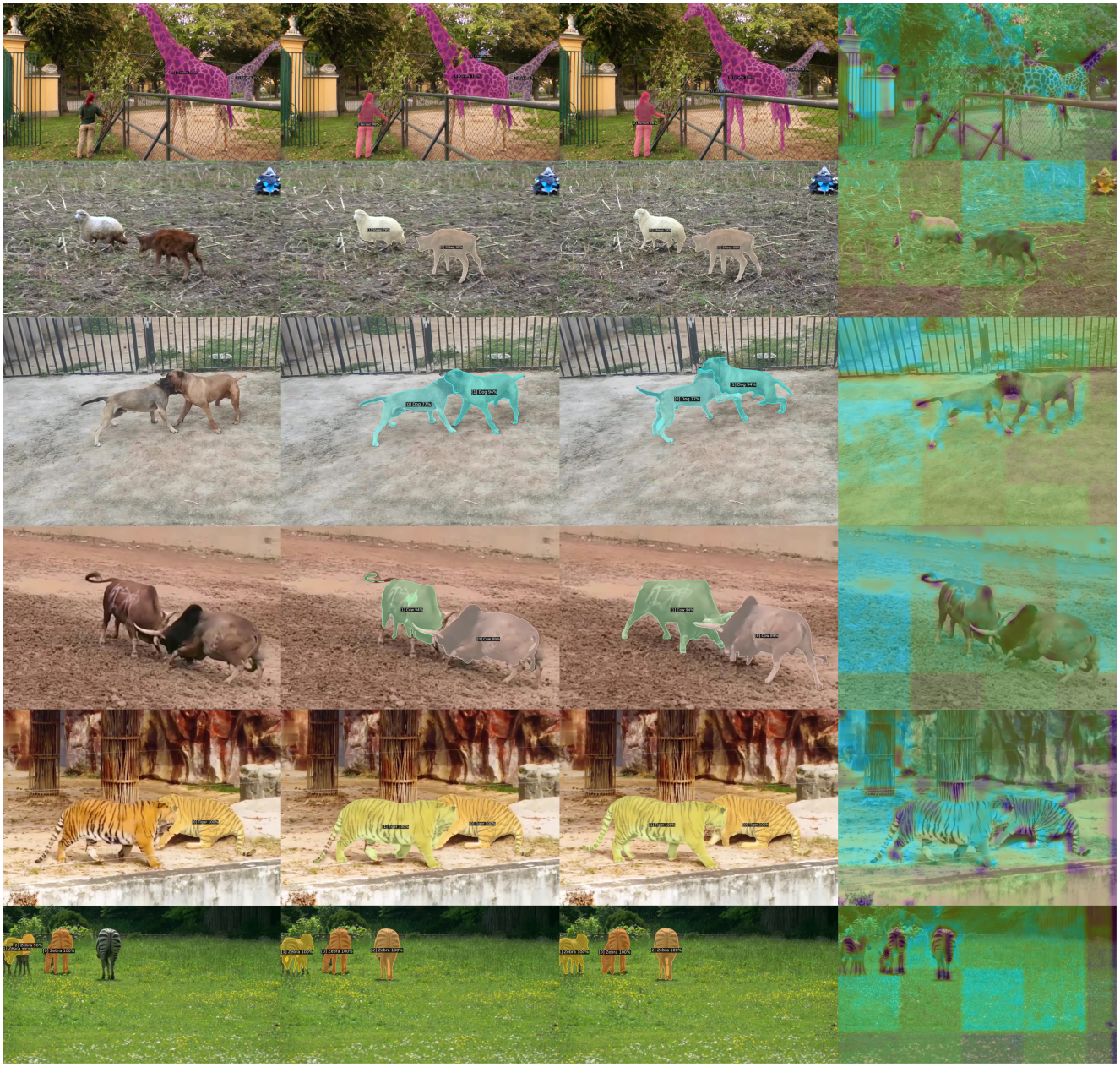

Figure 5. Further examples of comparisons and heatmaps from OVIS

References

[1] Ivana Balažević, Yuge Shi, Pinelopi Papalampidi, Rahma Chaabouni, Skanda Koppula, and Olivier J Hénaff. Memory consolidation enables long-context video understanding. *arXiv preprint arXiv:2402.05861*, 2024. 2

[2] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012. 1

[3] Yule Li, Jianping Shi, and Dahua Lin. Low-latency video semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5997–6005, 2018. 2

[4] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12179–12188, 2021. 1

[5] Yu-Syuan Xu, Tsu-Jui Fu, Hsuan-Kung Yang, and Chun-Yi Lee. Dynamic video segmentation network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6556–6565, 2018. 2

[6] Xizhou Zhu, Jifeng Dai, Lu Yuan, and Yichen Wei. Towards high performance video object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7210–7218, 2018. 2