Segment Any Motion in Videos

Supplementary Material

7. Pseudo-code for SAM2 Iterative Prompting

We present the pseudo-code for the first stage of SAM2 Iterative Prompting in Algorithm 1. The first stage focuses on grouping trajectories belonging to the same object and storing the trajectories of each distinct object in memory.

Algorithm 1 Process Invisible Trajectory with Memory

```
1:  Initialize iteration to 0
2:  Initialize memory_dict as an empty dictionary
3:  Set take_all to False
4:  if $traj.shape[1] \leq 5$ then
5:      Set take_all to True
6:  end if
7:  while iteration < max_iterations do
8:      Set $t$ to frame with maximum visible points
9:      Extract visible points at frame $t$
10:     Find densest point as $nearest\_point$
11:     Reset predictor state and add new point
12:     Reset predictor state
13:     Set $obj\_id$ to 1 and $labels$ to [1]
14:     Add new point using predictor to get mask
15:     Dilate the mask and determine points within the mask: $dilated\_mask$
16:     Determine points in prompt mask (visible + non-dilated): $prompt\_mask$
17:     if sufficient points in mask or take_all is True then
18:         Increment valid $obj\_id$
19:         Store object information in $memory\_dict$
20:     end if
21:     Remove points included in the mask from $traj$, $visiblemask$, and $confidences$
22:     Update $traj$, $visiblemask$, and $confidences$ with remaining points
23:     Increment iteration by 1
24:     if $traj.shape[1] < 6$ then
25:         Break the loop
26:     end if
27: end while
28: return memory_dict
```

8. Additional Experiment Details

Training Details. We train the model for 5 epochs, with each epoch comprising approximately 8000 steps, using the Adam optimizer with a learning rate of 1e-4 and a weight decay of 1e-4.

Model Architecture. As shown in the Fig 9, for the trajectory motion pattern encoder, we employ 4 heads for multi-head attention and a 64-dimensional feed-forward layer. For the tracks decoder, we use 8 heads for multi-head attention and a 512-dimensional feed-forward layer.

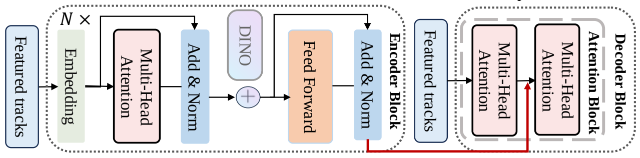

Figure 9. Architecture of tracks decoder.

Model efficiency and details. We parallelized the code during data processing. For a 50-frame video, processing takes 2 minutes, model inference 3 seconds, and object prompt generation requires 2 seconds per object only. For a dynamic object, 1-2 iterations are usually required. And the experimental settings are discussed in Sec 4.1 and Sec 7. Training time is about 60 hours, and the hardware used for training is an NVIDIA RTX A6000.

Point Trajectory. We utilize BootsTAP [15] to generate 2D tracks for query frames in video sequences. Specifically, query frames are selected at intervals defined by `step`, and 2D tracks are generated only for these frames. For training datasets, `grid_size` specifies the sampling grid resolution, determining the spacing between sampled points, while `step` controls the temporal interval between query frames. During training, we randomly select one query frame and load all its associated tracks to accelerate the process. For the Kubric dataset with a resolution of 512×512, we set `grid_size` to 8, generating 4096 points per frame, and `step` to 8, with the total number of tracks randomly sampled from $[512, 1024, 2048, 3000, 4096]$. For the HOI4D dataset with a resolution of 1920×1080, we set `grid_size` to 15, generating 9216 points per frame, and `step` to 15, with total tracks number sampled from $[1024, 1536, 2048, 4096, 5000, 6000]$. For the Stereo dataset with a resolution of 1280×720, we set `grid_size` to 32, generating 920 points per frame, and `step` to 8, with track counts sampled from $[256, 512, 768, 920]$. During inference, 2D tracks are also generated for each query frame. To ensure that dynamic objects appearing at different times are fully captured, tracks from all query frames are loaded, and 5000 tracks are randomly selected. For the FBMS-59 dataset [41], we set `grid_size` to 7 and `step` to 30, because some datasets contain relatively long sequences, we select a larger `step` to accelerate the loading process. For SegTrack V2 [33], `grid_size` is set to 5 and `step` to 8. For DAVIS-16 [45] and DAVIS-17 [46], `grid_size` is set to 10 and `step` to 8.

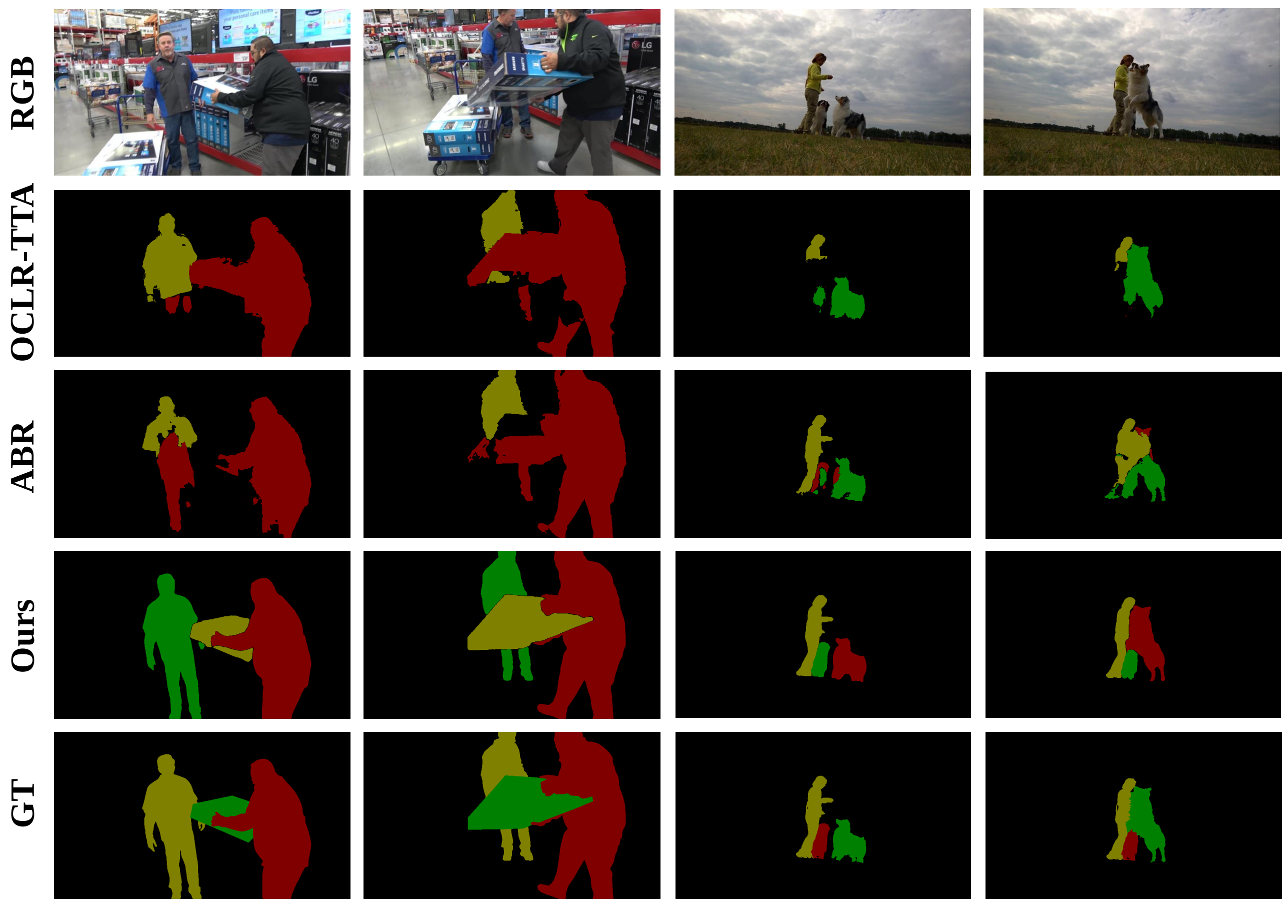

Figure 10. Our method demonstrates exceptional capability in generating fine-grained masks. Most previous approaches rely on the common fate assumption, where objects moving at the same speed are considered part of the same entity. Moreover, many methods lack the ability to produce fine-grained masks altogether. In contrast, our method can accurately distinguish and segment individual objects, even when they are closely positioned, moving simultaneously, or traveling at the same speed.

Failure case. We perform well on most sequences, but struggle on cases like the "penguin" in SegTrackv2 (Fig 11), where 90% of the content has similar motion. The lack of contrast and uniform motion patterns can cause the model to misinterpret object motion as camera motion, leading to a J metric of 0.014. Since SAM2 requires prompts, any failure in this process results in near-zero scores, whereas the baseline still achieves the 30-50 range even when it fails.

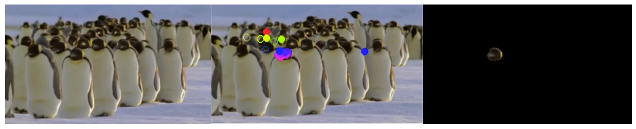

Figure 11. From left to right: input, dynamic tracks and mask.

9. Additional Experiments

Comparison with FlowSAM [60]. We further included a new baseline experiment (see Tab. 4) to demonstrate the superior performance of our model. It is worth noting that among all the baselines, only our method and FlowSAM require human annotation—our method needs human annotation during training, but not during inference.

Methods	Supervision	Davis16-m		Davis16	
		$\mathcal{J}\uparrow$	$\mathcal{F}\uparrow$	$\mathcal{J}\uparrow$	$\mathcal{F}\uparrow$
FlowSAM	YES	85.7	83.8	87.1	84.9
Ours	YES	**89.2**	**89.7**	**90.6**	**91.0**

Table 4. Comparison with FlowSAM on MOS task.

10. Additional Visualizations

We present additional visualizations on the three main datasets that we benchmark our method on [34, 36, 58, 59, 67]. We visualize our methods on DAVIS2016 in Fig. 12, Fig. 13 on the task of moving object segmentation. And Fig. 10 shows the result of fine-grained moving object segmentation on DAVIS2017.

Additionally, we provide a video demonstration featuring featuring non-cherry-picked examples from DAVIS16-Moving, showcasing both long-range trajectory label predictions and the final mask results.

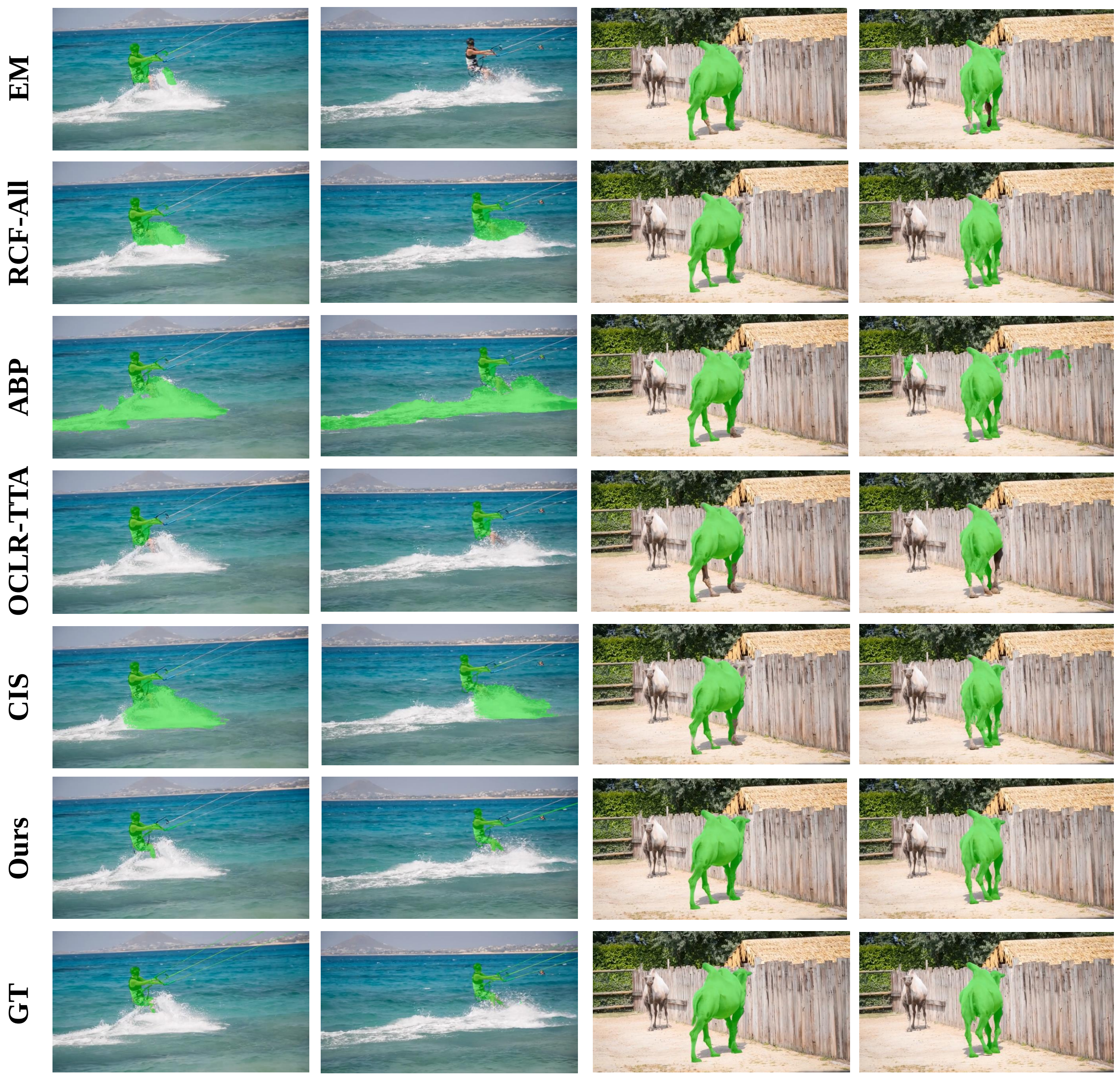

Figure 12. Our method effectively preserves the geometric integrity of articulated objects, such as human legs or camel limbs. At the same time, it can distinguish between dynamic backgrounds and foregrounds, focusing specifically on the object level. Additionally, it accurately identifies camouflage-like textures, such as a camel's head blending with the wooden fence in the background.

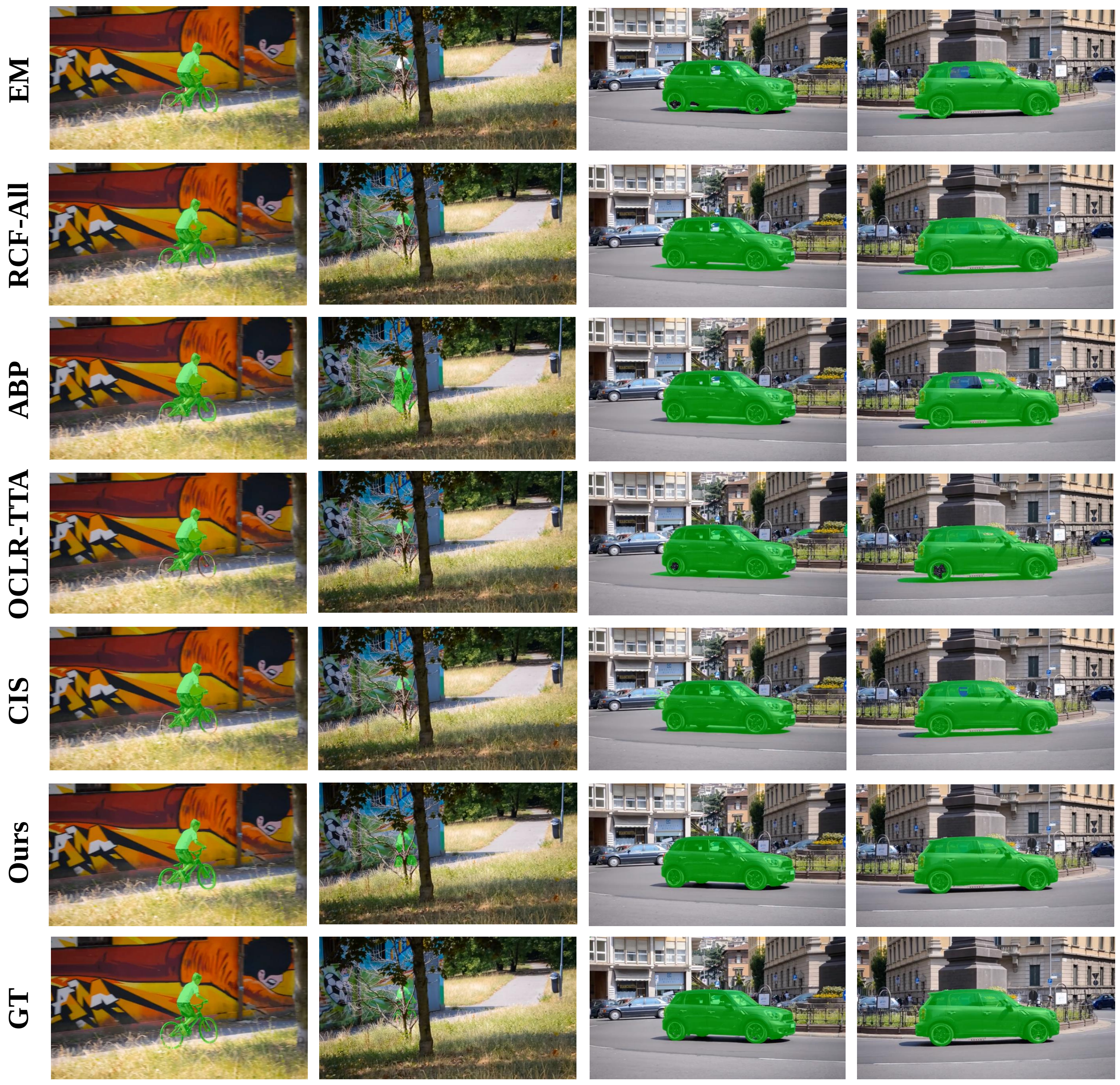

Figure 13. Our method handles occlusion scenarios more effectively. Thanks to long-range tracks, we can accurately follow a boy temporarily obscured by trees. Furthermore, our approach addresses complex situations, such as transparent glass, by including it in the mask to ensure the completeness of the moving object mask. Additionally, for highly intricate reflections, such as vehicle shadows, our method can accurately exclude them.