A More discussions.

A.1 Different attention weight distribution requirements.

We measured the distributions of the attention weights for content and position update in our cross attention, as visualized in Fig. 5, the distributions of these two groups of attention weights show a significant difference, indicating that the attention weights required by content and position update are different. This can also verify the importance of our weight disentangle design in APU.
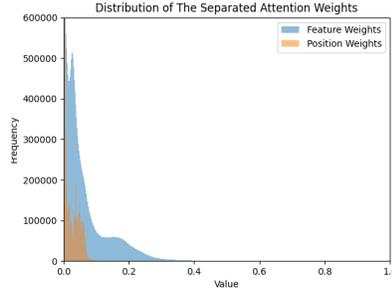
Figure 5: Attention weight distributions for feature and position updating in our cross attention.

A.2 Removing of cost-volume makes framework lightweight.

As shown in Table. 5, TAPTRv2 exhibits a faster speed and lower resource requirements compared to TAPTR. More importantly, it's a common case that in the downstream tasks, we need to track all pixels in a region (e.g. tracking a text written on the back of a horse) rather than just a few scattered points. In this case, the number of points to be tracked will reach tens of thousands. However, since the computation of the cost-volume and also the cost-volume aggregation operation in TAPTR increases sharply with the number of tracking points, with the number of tracking points increased, the advantage of TAPTRv2 will become more and more pronounced. As shown in the right table, when the number of tracking points reaches 5000 (which is only 1.9% of the pixels in a 512x512 image), the advantage of TAPTRv2 in speed and resource consumption becomes much more significant (about 24% faster and 20% fewer computational resource requirements).

800 Points	FPS	GFLOPS	#Param	5000 Points	FPS	GFLOPS	#Param
TAPTR	65.9	147.2	39.2M	TAPTR	11.8	426.8	39.2M
TAPTRv2	69.1	143.4	38.2M	TAPTRv2	14.6	354.2	38.2M

Table 5: Comparison of resource requirements between TAPTR and TAPTRv2. We evaluate TAPTR and TAPTRv2 on A100 GPU (80G), and the computational cost (GFLOPS) is calculated following detectron2.

B More Visualizations

B.1 Application of TAPTRv2 in Video Editing

Here we show the results of the video editing using TAPTRv2. After the users plot on one frame to specify the region to be edited, we sample points in the editing area and track these points across the whole video. For more details please refer to the videos in our supplementary material.

Fig. 6 (a) not only shows the ability of video editing but also the potential of TAPTRv2 in applying in 3D reconstruction.

Fig. 6 (b) shows that TAPTRv2 can handle the color change during the tracking. More importantly, although the editing area is cluttered in the middle of the video TAPTRv2 can robustly continue tracking the editing area when it reappears again.

Fig. 6 (c) shows that TAPTRv2 has the ability to handle the changing of scale.

B.2 Application of TAPTRv2 in Trajectory Estimation

In Fig. 7 we show the results of the trajectory estimation using TAPTRv2. After the users click on one frame to specify the points to be tracked, TAPTRv2 will keep tracking these points across the whole video to construct their trajectory.

14

A user write "**tiger**" on image, and track "tiger" throughout the video.

(a)

A user write "**o**" on chameleon, and track "o" throughout the video.

(b)

A user plot circle regions on **cars**, and track the regions throughout the video.

(c)
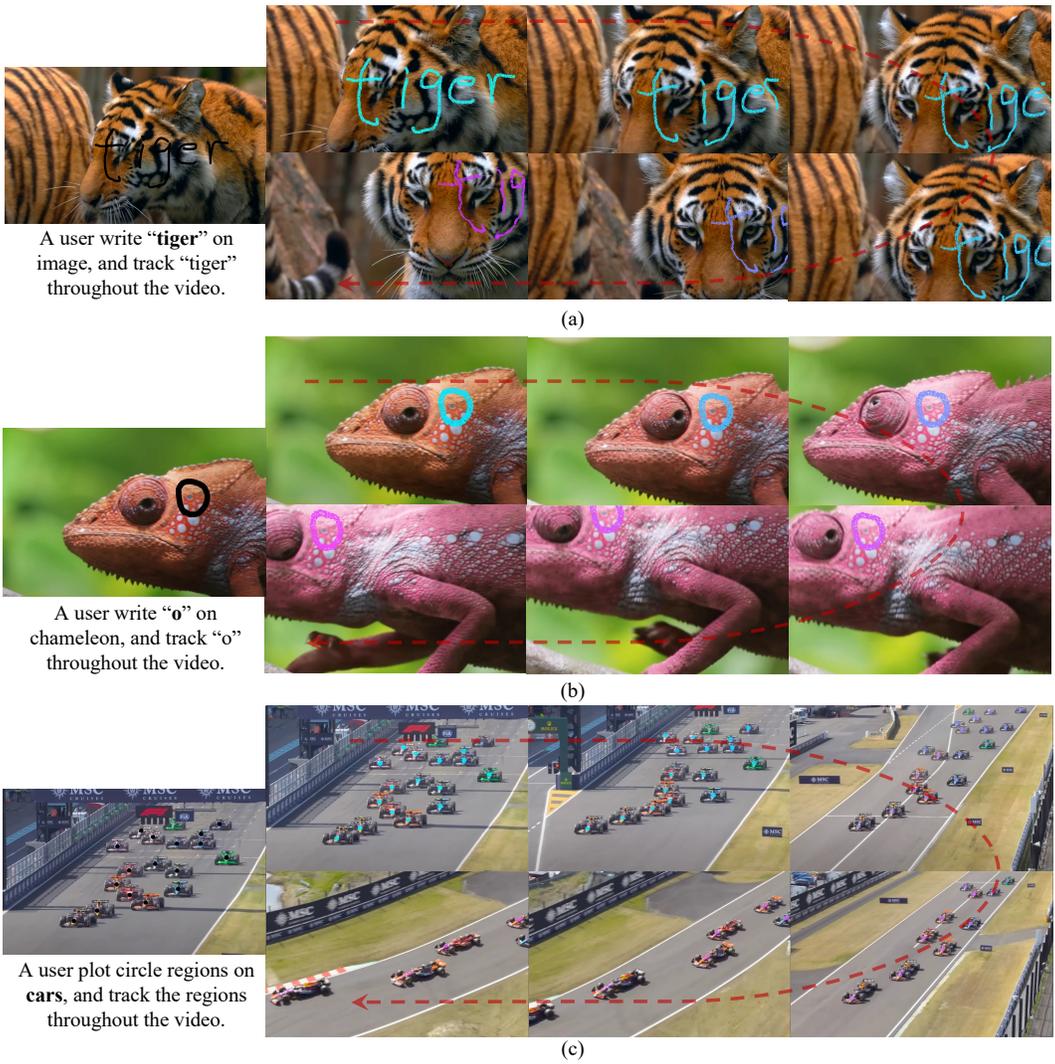
Figure 6: Apply TAPTRv2 in Video Editing. The color of the editing area changes over time.

15

A user click points on "**fighers**", and track points throughout the video.

(a)

A user click points on the "**horse**", and track points throughout the video.

(b)

A user click points on the "**car**", and track points throughout the video.

(c)
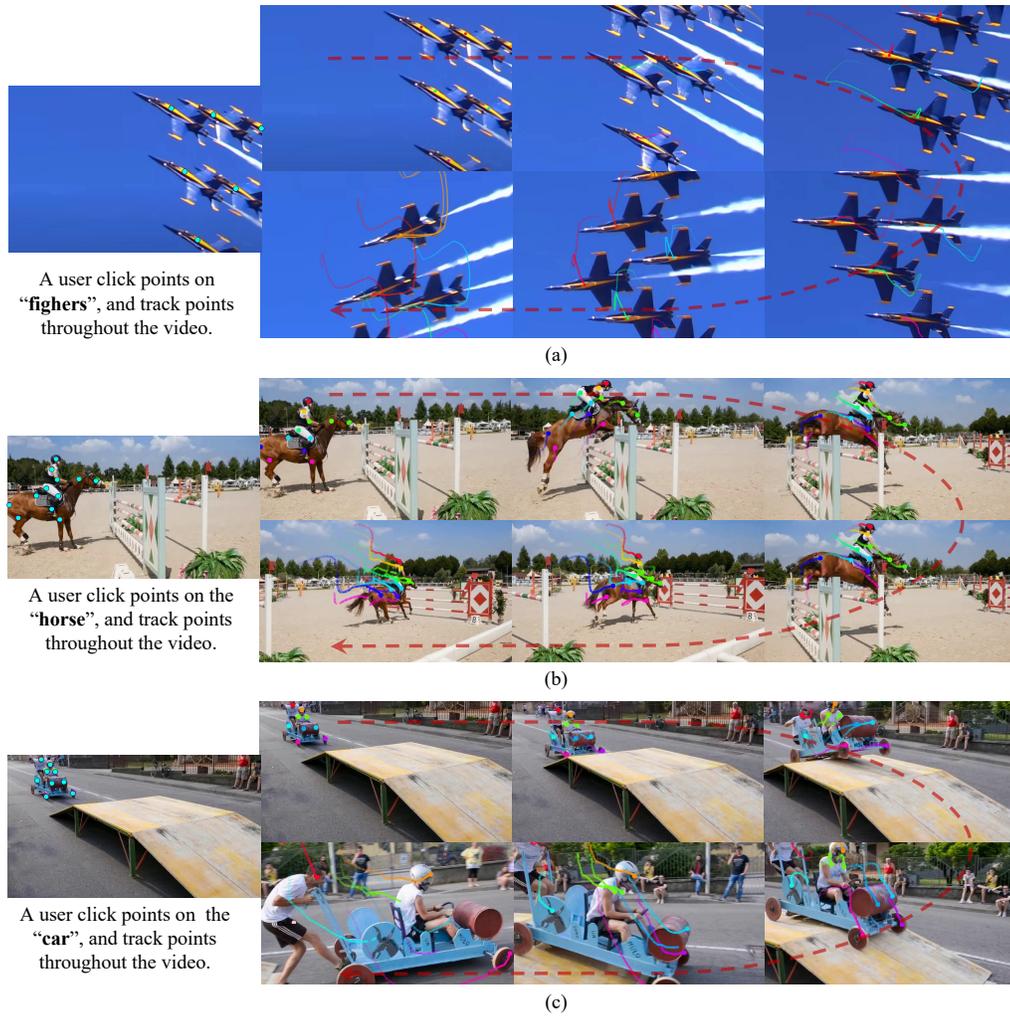
Figure 7: Apply TAPTRv2 in Trajectory Estimation.

16