ProTracker: Probabilistic Integration for Robust and Accurate Point Tracking

Supplementary Material

1. Video Results

Please refer to our Supplementary Webpage for the corresponding videos of images illustrated in the paper and more results on different data.

2. Implementation Details

2.1. Flow Preparation

We utilize the RAFT [5] model, as adopted by MFT [4], as our flow estimation model. It takes two images, $\boldsymbol{I}_j$ and $\boldsymbol{I}_i$, captured at different times as input and outputs the flow map $\boldsymbol{\mathcal{F}}_{ji}$, occlusion map $\boldsymbol{\mathcal{O}}_{ji}$, and uncertainty map $\boldsymbol{\mathcal{U}}_{ji}$. Since the uncertainty of an estimation can also be interpreted as its variance, the initial flow prediction from frame j to frame i at location $\boldsymbol{p}$ is computed as follows:

$$(\boldsymbol{f}_{ji}, \mu_{ji}) = \begin{cases} (S(\boldsymbol{\mathcal{F}}_{ji}, \boldsymbol{p}), S(\boldsymbol{\mathcal{U}}_{ji}, \boldsymbol{p})) & \text{if}(\boldsymbol{\mathcal{O}}_{ji}, \boldsymbol{p})) > \rho \\ None & \text{otherwise} \end{cases} \tag{1}$$

where $\rho = 0.1$, and $S(Target, \boldsymbol{p})$ indicates sampling the target at location $\boldsymbol{p}$. The initial flow predictions are then set as input to flow integration. Following MFT, we adopt $\{\infty, 1, 2, 4, 8, 16, 32\}$ as the time intervals, meaning that each frame's prediction is computed by combining results from these previous frames.

2.2. Filter Thresholds

During the dual filtering stage, we apply different thresholds to predictions from flow and long-term keypoints. For long-term keypoints, a prediction is marked as invalid if the cosine similarity to the query point on the geometry-aware feature falls below 0.5. For predictions from flow, however, we use a threshold of 0.3 instead. These distinct thresholds allow flow to track featureless areas while ensuring that long-term keypoints do not drift into visually similar regions.

2.3. Outlier Removal in Integration

Even after dual filtering, occasional erroneous predictions may persist. To ensure a more stable integration process and minimize the impact of these incorrect predictions, we discard rough predictions that deviate significantly from others. Denoting the input predictions as $\{(\boldsymbol{f}_1, \mu_1), (\boldsymbol{f}_2, \mu_2), \ldots, (\boldsymbol{f}_N, \mu_N)\}$, we remove outliers using the following criterion:

$$(\boldsymbol{f}_i, \mu_i) = \begin{cases} (\boldsymbol{f}_i, \mu_i) & \text{if } Norm(\boldsymbol{f}_i - \boldsymbol{f}_T) < \rho_{dist} \\ None & \text{otherwise} \end{cases} \tag{2}$$

where $\rho_{dist} = 10$ and $Norm(\boldsymbol{x})$ measures the magnitude of a vector.

3. Dense Inference

As discussed in Sec.3.1 in the main paper, we utilize a geometry-aware feature extractor and a video mask generator for the dual-filter stage. While optical flow and geometry-aware features can be computed densely, generating masks for each pixel is both time-intensive and memory-intensive. To address this, we adopt an iterative approach to efficiently generate a set of masks that collectively cover all pixels, as described below:

Algorithm 1 Dense Mask Generation Algorithm

```
Initialize: Set all pixels as unassigned.
while there exist unassigned pixels do
    Select the first unassigned pixel p.
    Generate a new mask M starting from pixel p.
    for each pixel q in M do
        if q is unassigned then
            Assign q to mask M.
        end if
    end for
end while
Output: All pixels assigned to corresponding masks.
```

Subsequently, the dual filter can be applied to each pixel based on its corresponding mask.

4. Training and Inference Speed

The total time consumed for our method includes the time for keypoint extraction, mask generation, geometry-aware feature extraction and probabilistic integration. During keypoint extraction, we follow DINO-Tracker [6] to train a delta-DINO model and a heatmap refiner, which takes about 1 hour for an 80-frame video on a single RTX 4090 GPU. We refer to DINO-Tracker [6] for more details. However, our method skips the time-consuming occlusion prediction and directly uses points with high cosine similarity as keypoints, which saves much time. The mask generation and geometry-aware feature extraction together takes about 2 minutes and the probabilistic integration takes about 1 minute for the same video.

In total, during the inference stage, the time spent tracking 3,000 points on a single object in an 80-frame video is about 3 minutes, which is about 20x faster than DINO-Tracker [6]. For dense inference, an additional 4 minutes

may be required due to the increased number of masks generated, but our method remains more than 30x faster than DINO-Tracker [6].

5. More Qualitative Results

To further illustrate our methods' robustness. We conduct experiments on more challenging cases and show the qualitative results.

Some of the previous methods rely on computing a heatmap between the query point and the target frame. However, the per-frame heatmap lacks temporal-awareness and may confuse different objects. We address this issue by leveraging the mask and combining the heatmap with optical flow. As illustrated in Fig. 1 and Fig. 2, by comparing the results of our method with DINO-Tracker [6] and TAPIR [1], we show that although our method also relies on per-frame heatmap to extract keypoints,our method has strong temporal-awareness and is able to tell between similar objects.

To further demonstrate the robustness of our method, we conduct experiments on extended videos from TAP-Vid-DAVIS, simulating high frame-rate videos by repeating each frame three times, as illustrated in Fig. 3 and Fig. 4. In contrast to typical sliding-window or flow-based trackers (such as TAPTR [3], SpatialTracker [7] and CoTracker [2]), which tend to accumulate errors and drift over time, our integration of long-term key points with short-term optical flow enables continuous, drift-free tracking of the same point through occlusions.

References

[1] Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, Joao Carreira, and Andrew Zisserman. Tapir: Tracking any point with per-frame initialization and temporal refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10061–10072, 2023. 2

[2] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together. *arXiv preprint arXiv:2307.07635*, 2023. 2

[3] Hongyang Li, Hao Zhang, Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, and Lei Zhang. Taptr: Tracking any point with transformers as detection. In *Proceedings of the IEEE/CVF European Conference on Computer Vision*, 2024. 2

[4] Michal Neoral, Jonáš Šerỳch, and Jiří Matas. Mft: Long-term tracking of every pixel. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6837–6847, 2024. 1

[5] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 402–419. Springer, 2020. 1

[6] Narek Tumanyan, Assaf Singer, Shai Bagon, and Tali Dekel. Dino-tracker: Taming dino for self-supervised point tracking in a single video, 2024. 1, 2

[7] Yuxi Xiao, Qianqian Wang, Shangzhan Zhang, Nan Xue, Sida Peng, Yujun Shen, and Xiaowei Zhou. Spatialtracker: Tracking any 2d pixels in 3d space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20406–20417, 2024. 2

CVPR
#271

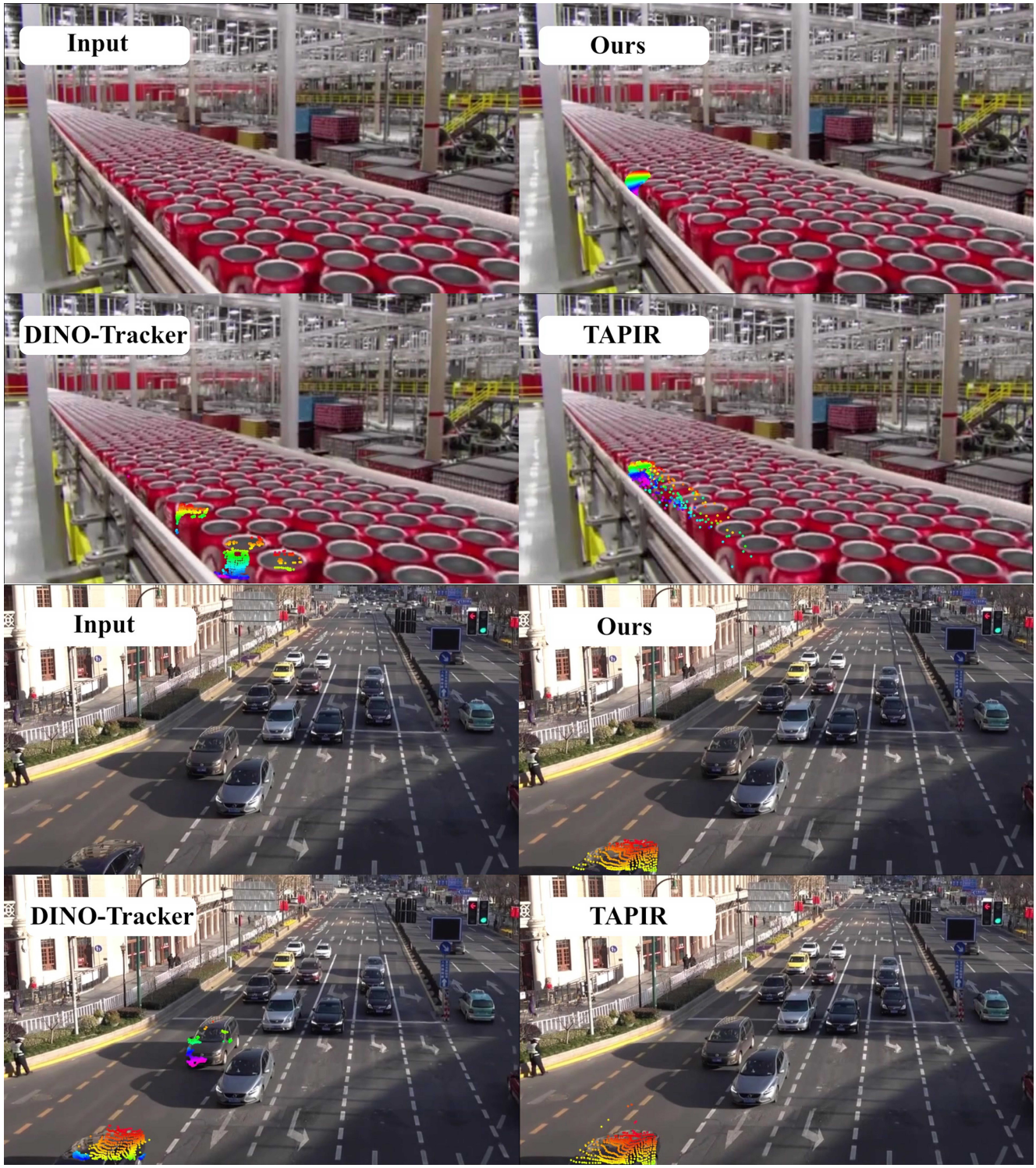

Figure 1. Results of tracking a single object. While DINO-Tracker may mispredict parts onto similar objects and TAPIR can be disrupted by similar patterns, our method avoids these errors.

CVPR
#271

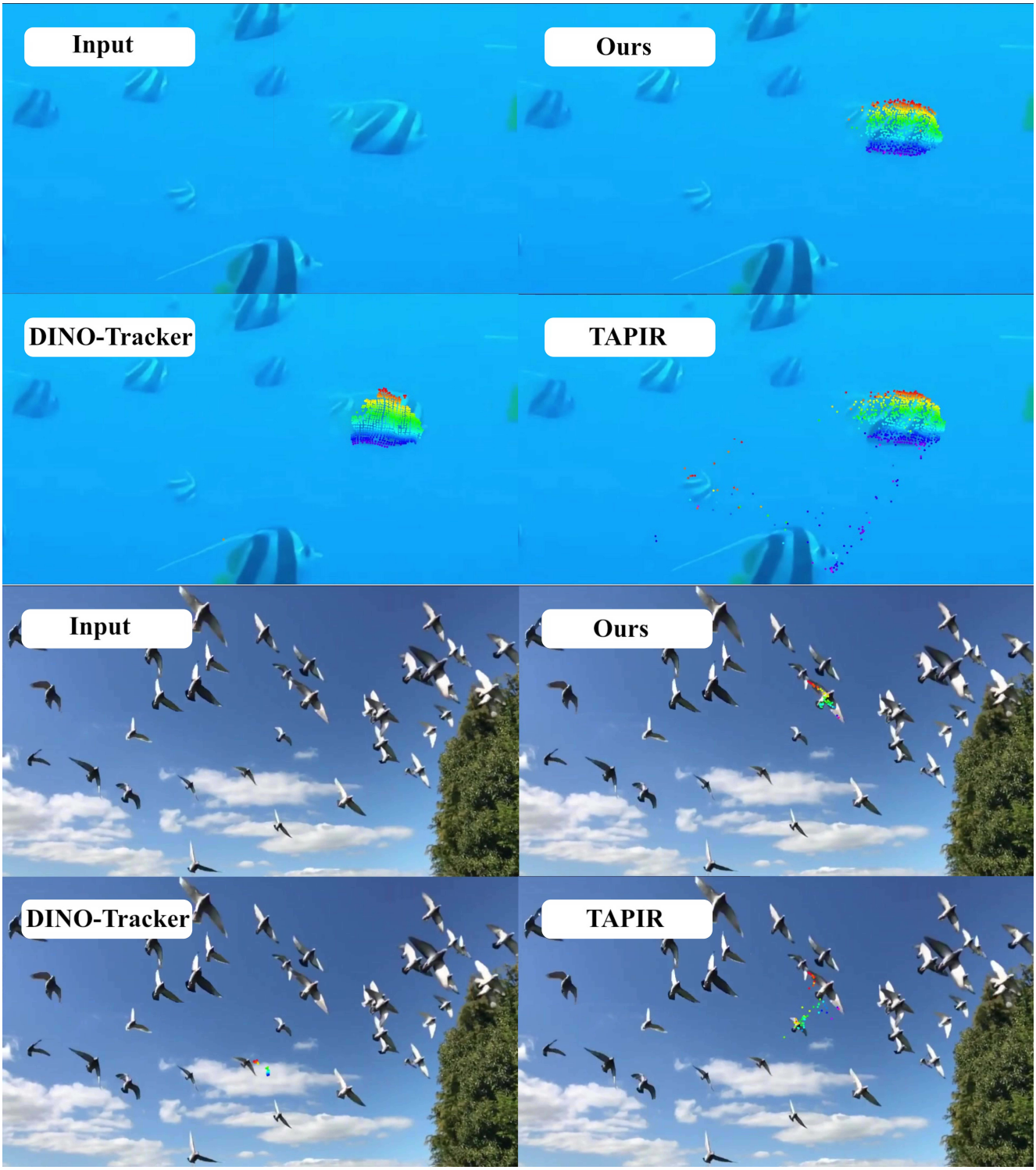

Figure 2. Results of tracking a single object. While DINO-Tracker may lose some parts and TAPIR can be disrupted by multiple similar patterns, our method avoids these errors.

Figure 3. Results of tracking at a higher frame rate. Sliding window based methods can easily lose track after occlusion and drift due to accumulating errors, while ours exhibit robustness.

Figure 4. Results of tracking at a higher frame rate. Sliding window based methods can mispredict points to other regions during occlusion (e.g. the gun and rope in *shooting* and the wrong person in *india*), while ours exhibit robustness.