

LONG-TERM 3D POINT TRACKING BY COST VOLUME FUSION

SUPPLEMENTARY MATERIAL

Anonymous authors

Paper under double-blind review

The supplementary material consists of this document and a video demo. In the document, we provide additional ablation results and result analyses. In the video demo, we show comparisons between our approach and CoTracker, the strongest 2D point tracking baseline used in our experiments.

1 RUNNING TIME BREAKDOWN

In this section, we show the running time breakdown for our model. We benchmark our model on the test split of the PointOdyssey dataset. The resolution of each frame in the dataset is 540×960 . Therefore, the maximum points on each frame is up to 518400. Our average FPS on this dataset is around 2.8.

As shown in Fig. 1, our model takes roughly half of the total running to extract the point cloud and query's features (i.e., **encode & decode**). The Cost Volume Fusion module, which predicts the query motion, takes roughly one third of the total running time. To further cut down the running time, we can replace our backbone network with a faster network.

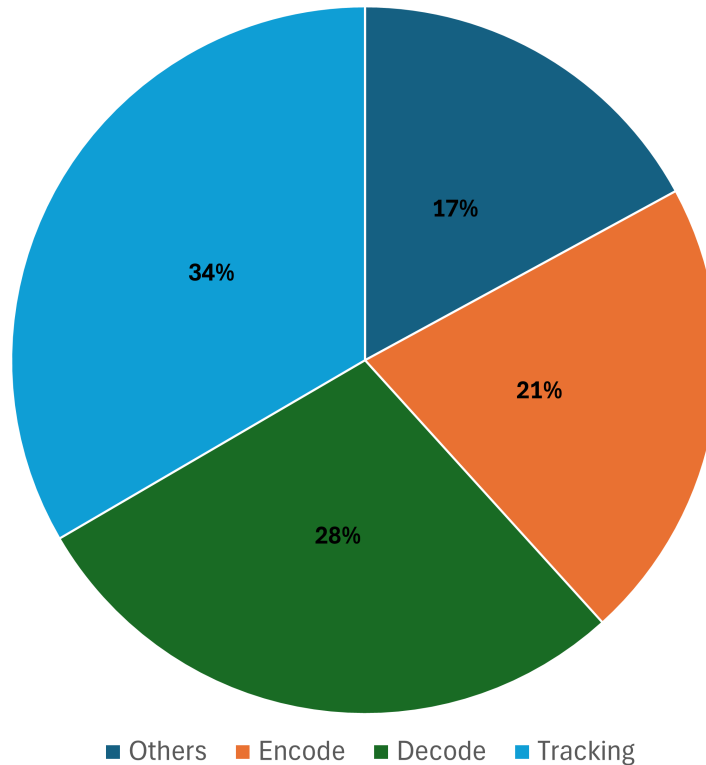


Figure 1: Running Time Percentage by Components

Table 1: Ablation on the Effect of Dense Point Cloud in Long-term Tracking

| Method | Sparse PC | Dense PC |
|-------------------------------------|-----------|-------------|
| 2D - $\delta^{avg} \uparrow$ | 80.5 | 87.8 |

Table 2: Compression Rates (CR) on 2 densest point cloud levels

| | Level 1 | Level 2 |
|-----------|---------|---------|
| CR | 13.99 | 9.29 |

2 ABLATION - EFFECT OF DENSE POINT CLOUD ON LONG TERM TRACKING

As mentioned in the main paper, the selective decoding module enables the use of a dense point cloud by reducing memory consumption significantly. In this ablation, we show the importance of using dense point clouds. As shown in Table. 1, the overall performance on long-term tracking is improved by switching from sparse point cloud input into dense point cloud input.

3 ABLATION - EFFECT OF SELECTIVE DECODING

Table 2 shows the compression rate we achieve by reducing the number of points to be decoded in the 2 most densest levels. Here, we define the compression rate to be the ratio between the number of points before and after the pruning process of the selective decoding module.

Besides, we can significantly reduce the total memory consumption when training the model - **Table 3**. Here, we show memory consumption during the pre-training process on the scene flow dataset with batch size 8. Besides, it also helps to speed up the training process by 1.7 times.

4 IN-DEPTH 3D PERFORMANCE COMPARISON TO CoTracker

While working well in 2D space, CoTracker (Karaev et al., 2023)’s performance significantly drops in 3D. We hypothesize that the main reason is due to noisy depth values used to convert 2D points into 3D. To validate our hypothesis, we visualize 3D tracks using new camera views by rotating the camera around the z-axis as follows. First, we back-project the predicted 2D point tracks of CoTracker into the 3D scene using the original camera pose and the provided depth maps. The depth of each predicted query point is obtained by first selecting the four nearest neighbors in the depth map (i.e., 4 depth pixels) and calculating the depth through bilinear interpolation. We also show the results for nearest interpolation in Sec. 5. Then, these points are projected to 2D again using new viewing angle. In contrast to CoTracker, we can directly project our predicted 3D points into the new viewing angles. The 2D results evaluated in the new camera views are shown in Table. 4. Although CoTracker performs well in the original view, its performance rapidly declines when these points are projected into new camera views. In contrast, our model exhibits more robustness to changes in viewpoint.

When analyzing the results of CoTracker in the new view - Fig. 2, we observe many cases where the points move in zig-zag patterns or have abnormally large motions at some specific time steps. We examine the depth values around these points in the original view and find that they typically reside on or near the object boundary - Fig. 3. Consequently, if the predicted positions of CoTracker for the query points are slightly off by a few pixels, it can lead to a significant discrepancy in the extracted depth. This discrepancy causes the queries to oscillate when they are projected into different views.

Table 3: Ablation on the Effect of Using Selective Decoding

| Training Memory | GBs |
|----------------------------------|-----------|
| Decode all points | 37 |
| Selectively decode points | 17 |

Table 4: Rotate the camera around the z-axis. The camera always looks at (0, 0, 0).

| | δ^{avg} | $\delta_{15^\circ}^{avg}$ | $\delta_{30^\circ}^{avg}$ | $\delta_{45^\circ}^{avg}$ | $\delta_{60^\circ}^{avg}$ | $\delta_{75^\circ}^{avg}$ | $\delta_{90^\circ}^{avg}$ |
|-----------|----------------|---------------------------|---------------------------|---------------------------|---------------------------|---------------------------|---------------------------|
| CoTracker | 93.8 | 87.1 | 81.2 | 74.4 | 65.4 | 52.4 | 43.8 |
| Ours | 87.8 | 87.5 | 86.3 | 84.0 | 79.2 | 68.6 | 62.5 |

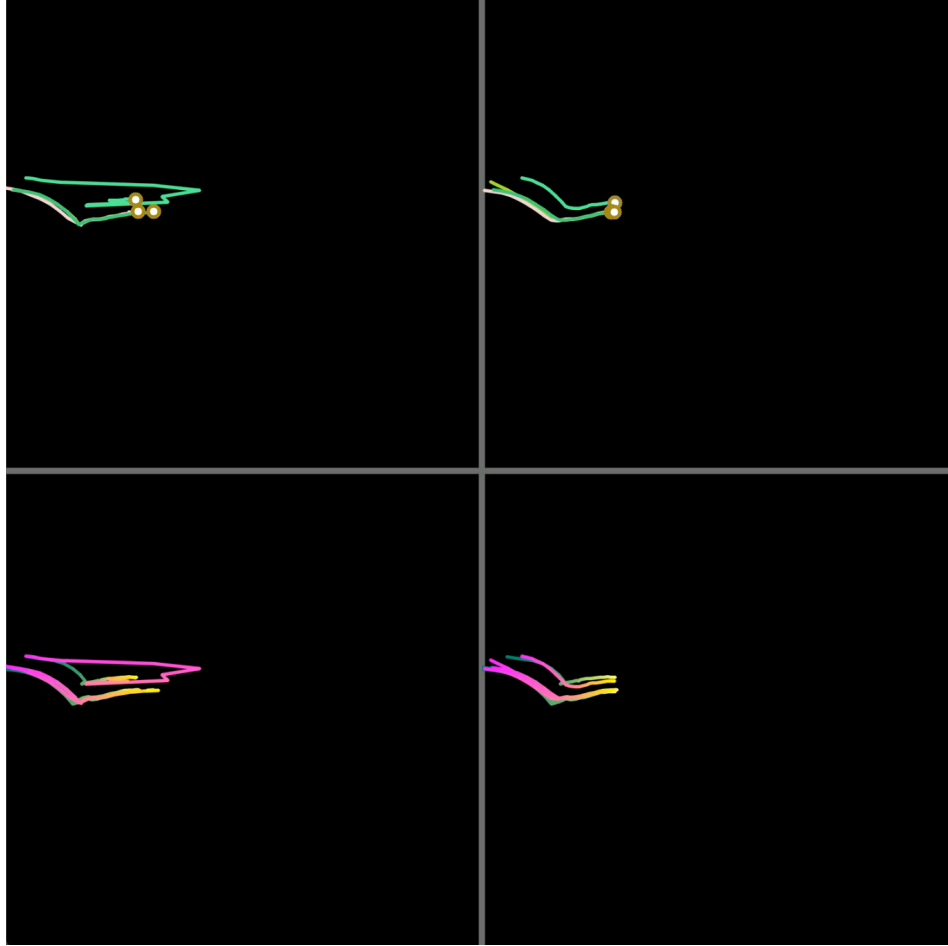


Figure 2: Comparison between CoTracker and our method’s predictions in the new view (i.e. 45°). **Upper-left:** CoTracker’s results. **Upper-right:** Our results. **Lower-left:** Cotracker’s predictions overlapped with GT trajectories. **Lower-right:** our predictions overlapped with GT trajectories. We show more results in our video.

5 ABLATION - NEAREST AND BILINEAR INTERPOLATION FOR LIFTING INTO 3D

In this section, we show the results of CoTracker (Karaev et al., 2023) and TAPIR (Doersch et al., 2023) when we use the nearest neighbor to extract the depth value for each query point from the GT depth map - Table 5. From these results, it can be shown that the nearest neighbor can extract the depth value for the query more precisely and improve the overall results of CoTracker and TAPIR consistently. However, despite the improvement, CoTracker and Tapir still underperform compared to ours in terms of the 3D metrics.

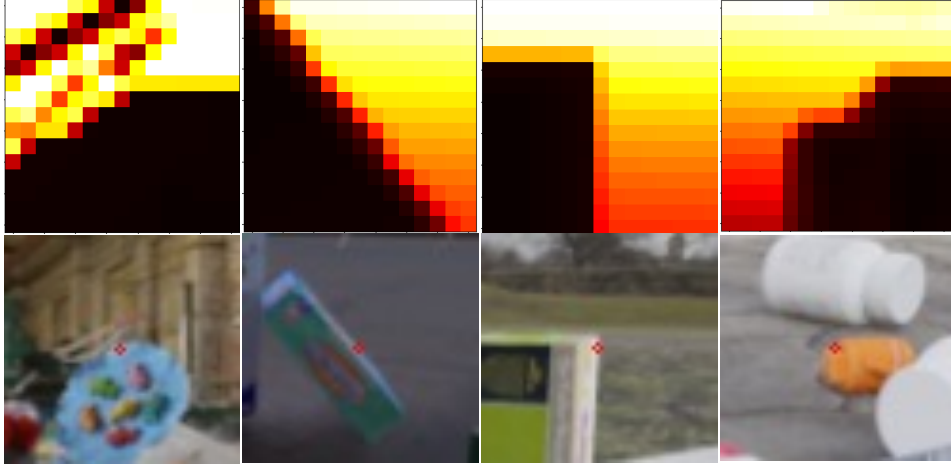


Figure 3: Local patches of the GT depth maps used in the original view. These patches are cropped around points whose projections in the new view move abruptly.

Table 5: Ablation on interpolation types for lifting 2D into 3D. BI: Bilinear Interpolation. NI: Nearest Neighbor

| | OA | 3D - δ^{avg} | $\delta_{occluded}^{avg}$ |
|----------------|-------------|---------------------|---------------------------|
| CoTracker (BI) | 92.5 | 57.8 | 8.7 |
| CoTracker (NI) | 92.5 | 58.7 | 9.7 |
| TAPIR (BI) | 96.5 | 46.9 | 3.8 |
| TAPIR (NI) | 96.5 | 47.8 | 4.4 |
| Ours | 93.4 | 73.1 | 44.0 |

6 ABLATION - 2D LOSS FINE TUNING

To compare with other 2D baselines, we have to project our 3D results into 2D. To reduce the effect of the numerical error when projecting, we tried to tune our model directly in 2D with the projection loss:

$$L_{track}^{2D} = \frac{1}{Tn_q} \sum_{l=1}^L \sum_{t=1}^T \sum_{i=1}^{n_q} \alpha^{l-1} |q_{t,i}^{2D} - \hat{q}_{t,i}^{2D}|_1$$

where $q_{t,i}^{2D}$ is the projection of the predicted 3D position of the query point, and $\hat{q}_{t,i}^{2D}$ is the 2D ground truth. Thanks to this, the accuracy in 2D is slightly improved. The ablations in the main paper were done without this projection loss hence the numbers were a little lower than tables 5 & 6 with the 2D results.

7 MORE QUALITATIVE RESULTS - REBUTTAL

We provided **2 more videos** (kubric_comparison.mp4 & odyssey_comparison.mp4) to compare our method with the scene flow chaining baseline on Tapvid Kubric and Point Odyssey dataset.

We show in the videos that with the help of past motions and appearances, our tracking framework can overcome occlusion - Fig. 4.

Table 6: Ablation on the Projection Loss to reduce projection error.

| | With Projection Loss | Without Projection Loss |
|----------------|----------------------|-------------------------|
| δ^{avg} | 87.8 | 87.0 |

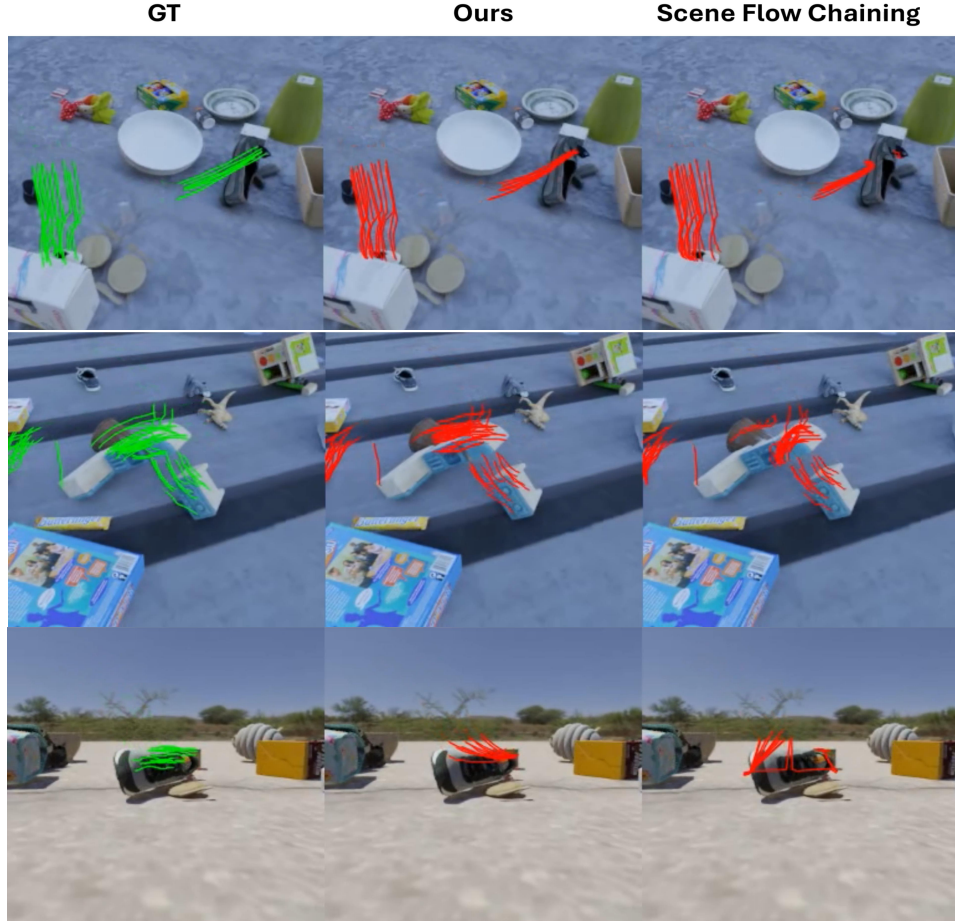


Figure 4: Past motions help the model to track the small object near the shoe through occlusion.

Table 7: Results on real-world Tapvid3D-ADT dataset.

| Method | 3D-AJ \uparrow | APD \uparrow | OA \uparrow |
|--------------------------------------|------------------|----------------|---------------|
| Ours (fine-tuned on one part of ADT) | 49.9 | 63.5 | 88.5 |
| Ours (trained only on Tapvid) | 11.7 | 25.0 | 56.1 |
| Spatial Tracker (Xiao et al., 2024) | 10.1 | 14.9 | 72.6 |

Besides, in many cases, multiple trajectories predicted by the scene flow chaining degenerate and are merged into fewer trajectories - Fig. 5.

8 ADDITIONAL RESULTS ON REAL-WORLD DATASET

Our results on the real-world Tapvid3D-ADT (Koppula et al.) dataset are presented in Table 7. Following the approach in Koppula et al., we evaluate our method using the 3D-AJ, APD, and OA metrics.

Our method, while being trained only on the synthetic Kubric dataset, achieves better performance than SpatialTracker. Fine-tuning on a small subset of the ADT dataset further boosts performance significantly. By tracking points directly in the world coordinate system rather than the camera coordinate system, we disentangle camera motion from actual point motion, making it easier to distinguish between dynamic and static points.



Figure 5: Multiple trajectories predicted by Scene Flow Chaining are merged into one.

REFERENCES

- 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. *arXiv preprint arXiv:2306.08637*, 2023. 3
- Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker: It is better to track together. 2023. 2, 3
- Skanda Koppula, Ignacio Rocco, Yi Yang, Joe Heyward, Joao Carreira, Andrew Zisserman, Gabriel Brostow, and Carl Doersch. Tapvid-3d: A benchmark for tracking any point in 3d, 2024. [URL https://arxiv.org/abs/2407.05921](https://arxiv.org/abs/2407.05921), 2(8):17. 5
- 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*, pp. 20406–20417, 2024. 5