Supplementary

A CaDeX++ and Canonical Space

We implemented the temporal feature grid $\Psi_l(i)$ in three resolutions: $T/20$, $T/4$, and $13T/20$, where T is the number of frames. Each resolution has a feature dimension of 16. For the spatial feature grid $\Phi_l(x, y)$, we implemented 2 resolutions 12 and 96, with feature dimensions 32 for each resolution. 2 hidden layers are set for the tiny MLP. We perform ablation studies on DAVIS [26] scenes: breakdance, bmx-trees, libby, parkour, and blackswan.

The tiny MLP predicts the positive incremental bias of the control points as $[(\Delta\alpha^1, \Delta\beta^1)...(\alpha^B, \Delta\beta^B)]$ together with the positive outlier slope k_l, k_r. We divide the incremental bias into two sets $\{(\Delta\alpha^i_N, \Delta\beta^i_N)\}_{i=1}^{B/2}$ and $\{(\Delta\alpha^i_P, \Delta\beta^i_P)\}_{i=1}^{B/2}$ to generate the control points with negative and positive α values. For the control points with negative α values, their coordinates are computed as:

$$(\alpha^k_N, \beta^k_N) = -(\sum_{i=1}^{k} \Delta\alpha^i_N, \sum_{i=1}^{k} \Delta\beta^i_N) \tag{13}$$

While the control points with positive α values are aggregated as:

$$(\alpha^k_P, \beta^k_P) = (\sum_{i=1}^{k} \Delta\alpha^i_P, \sum_{i=1}^{k} \Delta\beta^i_P) \tag{14}$$

For the input that lies outside the left-most or right-most control point (α_m, β_m), we compute the output as:

$$z' = k_m(z - \alpha_m) + \beta_m \tag{15}$$

where k_m is the outlier slope.

Although introducing CaDeX++ will not improve the accuracy substantially, it significantly accelerates the fitting as shown in the main paper Fig. 6 (2 times faster). Due to the efficient design, the average memory consumption while using CaDeX++ is **2752 MB**, compared to **9240 MB** for the version without it. We agree that the multiplication of spatial-temporal features is a feasible option and we further study it. However, the ablation study in Tab. 5 shows a slight performance decrease.

While OmniMotion adopts an implicit NeRF-like representation of the shared canonical space over the scene, our design replaces it with an explicit set of points that are consistently aligned via supervision. We warp and merge the points from different frames to the canonical space and observe a relatively consistent reconstruction as shown in Fig. 9. Such a design significantly accelerates the fitting by avoiding expensive volume rendering and the ambiguities inherent in alpha blending. Furthermore, the canonical space representation can also be coupled with other feature-render methods like 3D Gaussian splatting, which may be explored in future work.

Fig. 9: The panorama-like canonical space points of the soapbox scene.

B Preparing Long-term Correspondence

During training, we sample the flow for each query frame among a neighborhood of 12 frames and search for long-term correspondence outside a neighborhood of 10 frames. Coarse correspondences are computed on the low-resolution feature maps of DINOv2 [25]. We applied three strong filters to remove noise and keep representative matches.

– **Mutual Maximum**. For a matched pair (p_i, p_j) of two frames F_i, F_j, the best matching of p_i in frame F_j should be p_j and vice versa:

$$\underset{p_i \in F_i}{\operatorname{argmax}}\, S\langle \underset{p_j \in F_j}{\operatorname{argmax}}\, S\langle p_k, p_j\rangle, p_i\rangle = p_k,\ p_k \in F_i \tag{16}$$

where $S\langle p_i, p_j\rangle$ denotes the cosine similarity between the feature of points p_i, p_j. We only choose the pairs that have similarity over $\theta_m = 0.75$.

– **Background Filter**. For a point p_k in a matched pair, we compute the similarity between p_k with all other points in its feature map. Then we count the number of similar points beyond a threshold of θ_s. We keep the points that have less than N_s similar points. We set $\theta_s = 0.55$ and $N_s = 100$.

$$\sum_{p_i \in F_i} \mathbf{1}(S\langle p_k, p_k\rangle > \theta_s) < N_s \tag{17}$$

– **Local Noise Filter**. For a point p_k in a matched pair, we compute the similarity among its 11×11 neighbor points $M(p_k)$ and sum up all the similarity. We choose the points with total local similarity larger than $\theta_l = 30$.

$$\sum_{p_i \in M(p_k)} S\langle p_i, p_k\rangle > \theta_l \tag{18}$$

Fig. 10: Qualitative comparison of ablation configurations.

Table 5: Further ablation experiment results.

Configuration	AJ ↑	δ^x_{avg} ↑	OA ↑	TC ↓
Fixed Depth	42.5	58.4	76.8	3.71
Random Init Depth	-	-	-	-
Noisy Depth±0.2	46.6	64.0	76.8	2.91
Noisy Depth±0.5	-	-	-	-
UniDepth	48.0	64.4	**80.9**	1.42
GMflow	44.2	58.9	77.6	1.432
Noisy Flow±5pix	46.7	62.3	78.0	1.44
Noisy Flow±10pix	42.8	57.7	77.5	1.655
Feature Multiplication	46.5	63.3	77.9	1.296
Full (reported in main paper)	**48.6**	**65.7**	80.1	**1.14**

C Further Experiments

Qualitative results demonstrated in Fig. 10 prove that the introduction of the depth prior makes the tracking of points within the same instance more concentrated and less prone to dispersion. Besides, without long-term supervision, our method fails to handle large and frequent occlusions across time.

We conducted ablation experiments on different depth map configurations: (1) Fixed Depth: Disabling optimizable depth. (2) Random Init Depth: Initializing the depth maps from random noise. (3) Noisier Depth: Adding varying magnitudes of uniform noise to the initial depth maps. As shown in Tab. 5, fixing the depth maps limits the performance. The optimization diverges(rows 2 and 4 in Tab. 5) on randomly initialized or extremely noisy depth maps. Although slight noise in depth initialization hardly affects tracking precision, it is the prime factor affecting the robustness of tracking.

We adopt other SOTA depth and optical flow priors (UniDepth, GMflow) to explore the essence of our method as shown in Tab. 5 rows 5 and 6. Based on the result shown in Tab. 5, we find that the quality of optical flow is the primary determinant of tracking precision, while the depth prior significantly impacts the robustness.

We further measure the optimization magnitude on the DAVIS dataset. Tab. 6 shows the average mean and max pixel optimization magnitude, along with their ratio over the depth range. The visualization of the magnitude is illustrated in Fig. 11.

Table 6: Depth Optimization Magnitude

	mean	max
mag	0.053	2.844
ratio%	0.93	50.22

Fig. 11: Depth optimization visualization

D Optimization-based vs feed-forward

While the SOTA feed-forward methods pre-trained on large datasets with plenty of computing are still relatively expensive for inference, we admit that their inference time is faster than our fitting time. However, feed-forward methods rely on the assumption that the training prior distribution generalizes to the testing situation. As shown in Tab. 3 and Fig. 7 of our main paper, this assumption does not always hold, leading to failures in methods like CoTracker. In such cases, our optimization-based method proves to be more reliable. In other words, we believe an efficient optimization-based approach remains valuable for the community to build more robust systems when the inference runs out of distribution. We also would like to highlight that once trained, inference of the track of any pixel at any time with our method reaches near-instantaneous speed.

E Optimization Based on CoTracker

We utilizes the output of CoTracker as part of our training supervision for each scene. The optimization result on DAVIS dataset is shown in Tab. 7.

Table 7: Result of optimization on DAVIS with CoTracker output.

Method	DAVIS [26]			
	AJ↑	δ^x_{avg} ↑	OA↑	TC ↓
CoTracker [14]	**65.1**	79.0	**89.4**	0.93
Ours	62.2	**80.0**	86.8	**0.69**