Real-World Point Tracking with Verifier-Guided Pseudo-Labeling

Supplementary Material

In this supplementary material, we provide additional implementation details, analyses, and visualizations that complement the main paper. Sec. 7 introduces the verifier model in full, including its architecture, training setup, and trajectory perturbation strategy. Sec. 8 details the real-world adaptation pipeline and fine-tuning procedure. Sec. 9 presents additional ablation studies. Finally, qualitative examples and visualizations of the verifier are provided in Sec. 10.

7. Verifier Details

This section provides the complete specification of the verifier. We describe its architecture, training procedure, and the perturbation strategy used to synthesize diverse candidate trajectories during training.

7.1. Training Setup

Training schedule. The verifier is trained for 100 epochs on K-EPIC [9], corresponding to approximately 36K iterations with batch size 32. Training uses full 24-frame clips from K-EPIC, each providing up to 384 tracks.

Optimization. Optimization uses AdamW [30] on 32×A100 (64 GB) GPUs with mixed-precision training. The learning rate follows a cosine decay schedule with 1% warmup and peaks at 5×10^{-4}. Weight decay is set to 1×10^{-3}, and gradient norms are clipped to 1.0.

Data augmentation. We apply random resized cropping with scale range $[0.6, 1.0]$, horizontal flipping, and color jittering in brightness, contrast, saturation, and hue. Additional augmentations include Gaussian blur, low-probability solarization, and JPEG compression with a randomly sampled quality factor. Small geometric perturbations such as rotations up to 10° and mild perspective distortions are also applied. All augmentations are applied consistently across frames of a clip.

7.2. Architecture

Visual encoder. A stride-4 CNN encoder is initialized from the CoTracker3 video variant [21] and projected to the model width D=256 using a 1×1 convolution. A four-level feature hierarchy is constructed by average pooling the stride-4 feature map with factors $2, 4, 8$, producing features at strides $4, 8, 16, 32$. Feature sampling (as defined in the main paper) is performed at the highest resolution. These multi-scale features provide the input to a three-layer deformable attention decoder that extracts localized descriptors around query and candidate positions.

Localized feature extraction. As illustrated in Fig. 5, we first bilinearly sample a reference feature from the query-frame feature map $\mathbf{F}_{t_0}$ at the query location. Conditioned on this reference, a three-layer deformable attention decoder extracts localized descriptors by attending to regions centered at the query (at t_0) and at all candidate locations (at frame t) over the corresponding feature maps. Spatial displacements are encoded using a 2D sinusoidal embedding $\eta(\cdot)$ of size 32, concatenated with raw (x, y) coordinates and scaled by a learnable factor initialized to 16. A binary identity embedding distinguishes query tokens from candidate tokens. The resulting representations are concatenated, normalized, and projected to the model width. A learnable temporal embedding, initialized for 24-frame clips, is interpolated when clip lengths vary.

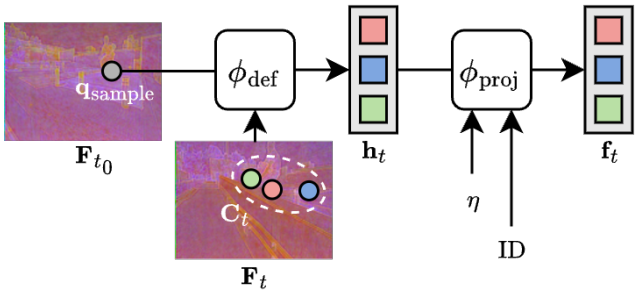

Figure 5. **Localized Feature Extraction.** Given frame-wise features of the query frame t_0 and target frame t, denoted by $\mathbf{F}_{t_0}$ and $\mathbf{F}_t$, we first bilinearly sample the reference feature $\mathbf{q}_{\text{sample}}$ at the query location. A deformable attention module ϕ_{def} then aggregates localized context around each candidate location ($\mathbf{C}_t$), producing descriptors $\mathbf{h}_t$. We concatenate displacement embeddings $\eta(\cdot)$ with an identity embedding (query vs. candidate) and project via ϕ_{proj} to obtain the final query and candidate features consumed by the candidate transformer.

Candidate transformer. Each multi-head attention block uses four attention heads followed by residual connections and normalization. The feed-forward network expands the hidden dimension by a factor of four, applies a GELU activation, and uses a dropout rate of 0.1 before a final normalization step. The module follows a post-normalization transformer design.

Verifier output. A ranking head linearly projects both query and candidate features, applies L2 normalization, and computes their dot-product similarity. A learnable temperature parameter, initialized to 0.1, scales the resulting logits to form the frame-wise reliability distribution.

7.3. Track Augmentations

To approximate realistic tracker behavior, we perturb ground-truth trajectories with stochastic transformations to generate candidate predictions during training. Each trajectory starts with light one-pixel Gaussian noise to model localization uncertainty. A random subset of the following perturbations is then applied, with each triggered by a fixed probability p:

1. **Stable noise ($p = 0.5$):** Small Gaussian displacements (2–4 pixels) smoothed across 3–5 frames.
2. **Gradual drift ($p = 0.4$):** Progressive displacement or soft identity blending with nearby visible points (within 16–32 pixels).
3. **Long-term drift ($p = 0.3$):** A steadily increasing offset up to 64 pixels.
4. **Spiky noise ($p = 0.3$):** Short, high-magnitude spikes of about 8 pixels that recover quickly.
5. **Abrupt jump ($p = 0.1$):** Sudden jumps or identity switches up to 128 pixels.
6. **Complete switch ($p = 0.1$):** Replacement of an entire trajectory with another visible one.

Multiple perturbations may co-occur, producing diverse deviations ranging from 1 to 128 pixels. This strategy exposes the verifier to the full range of errors observed in real trackers and encourages robust reliability estimation.

8. Real-world Fine-tuning Pipeline

This section describes the full configuration of our real-world fine-tuning pipeline, including training details and the teacher model configurations.

8.1. Training Setup

Training schedule. We fine-tune the pretrained Track-On2 [2] model for 24 epochs (approximately 3.7K iterations) with batch size 32 using a mixture of TAP-Vid Kubric [21] and the collected real-world dataset described in Sec. 4.5. During training, both datasets initially contribute equally to the loss. The loss weight of real-world videos is then linearly increased from 1.0 to 2.0, while the synthetic weight is reduced to 0. This schedule gradually shifts supervision toward real-world data.

Supervision. For synthetic videos, ground-truth trajectories and visibility annotations are available, and the model is supervised using all training losses. For real-world videos, we instead use verifier-guided pseudo-labels. Occluded frames are masked for localization losses based on majority-voted visibility across teacher predictions. The visibility and uncertainty heads are not explicitly supervised for real-world data. All remaining hyperparameters, including memory size and top-k candidate selection, follow the original Track-On2 training configuration.

Optimization. Training uses the AdamW optimizer [30] on 32 A100 GPUs (64 GB) with mixed-precision training. The learning rate follows a cosine decay schedule without warmup, with a peak value of 3×10^{-5}. Weight decay is set to 1×10^{-5}, and gradient norms are clipped to 1.0.

Query sampling and data preparation. Training uses clips of length 48 with 256 query points per video. For real-world videos, two thirds of the queries are sampled from SIFT [31] detections, while the remaining queries are selected from motion-salient regions obtained via grayscale frame differencing with mild spatial smoothing. This lightweight detector reliably identifies moving objects and reduces the proportion of static or low-texture points. Queries are extracted from the first 24 frames with temporal stride 4 (frames 1, 5, . . . , 25). If fewer than 256 keypoints are detected, the remaining locations are filled with randomly sampled points. We apply the same video-level augmentations used for verifier training (see Sec. 7.1) consistently across all frames of a clip. Teacher predictions are computed on the clean video (without augmentation) to obtain pseudo-label supervision.

8.2. Teacher Models and Reproduction

We evaluate all teacher methods using the official checkpoints released by their respective authors. Unless stated otherwise, each model is executed with its recommended inference configuration. This ensures that our reproduced results reflect the intended operating conditions of each method and provide a fair comparison. Specific implementation details are summarized below:

Track-On2 [2]. We use the official DINOv3 [37]-based checkpoint with memory size 24. A global grid of size 10 is employed when generating pseudo-labels.

BootsTAPIR [13] & **BootsTAPNext-B** [48]. Evaluated directly using the released checkpoints in `PyTorch` [34] with their default inference settings.

CoTracker3 [22]. The publicly available checkpoints correspond to real-world fine-tuned models. We use the window-input variant with a global grid of size 10 and joint multi-point tracking enabled.

Anthro-LocoTrack [8]. Base model evaluated at 256×256 resolution.

AllTracker [19]. Model trained on Kubric + Mix, evaluated at 384×512 resolution. As it is not a sparse point tracker, we run the model independently for each frame containing query points.

9. Additional Experiments

Comparison on synthetic benchmarks. In Table 4, we compare our real-world fine-tuned model with the previous work on the synthetic benchmarks Dynamic Replica [20] and PointOdyssey [47]. Dynamic Replica consists of 300-frame videos, while PointOdyssey contains extremely long sequences with thousands of frames, making it particularly

Table 4. **Quantitative results on synthetic benchmarks.** We compare the synthetic baseline Track-On2 with our real-world fine-tuned variant (Track-On-R) on Dynamic Replica (DR) and PointOdyssey.

Model	DR	PointOdyssey		
	δ^x_{avg} ↑	δ^x_{avg} ↑	MTE ↓	Survival ↑
BootsTAPNext	46.2	9.9	88.2	12.8
CoTracker3	72.3	44.5	20.7	56.3
Track-On2	74.5	45.1	22.0	57.7
Track-On-R (Ours)	**75.1**	**53.4**	**13.4**	**63.1**

suitable for evaluating long-term temporal robustness. Although our model is fine-tuned using only real-world videos, it still improves performance on synthetic benchmarks. On Dynamic Replica, the gain is modest (+0.6 δ^x_{avg}). In contrast, the improvement on PointOdyssey is substantial, with +8.3 in δ^x_{avg} and +5.4% in survival rate. These results indicate that our real-world adaptation does not degrade synthetic-domain performance, and in fact improves long-term tracking robustness, particularly on very long sequences.

Non-learning baselines. In Table 5, we compare the verifier against representative non-learning ensemble heuristics: (i) geometric median (Weiszfeld's algorithm); (ii) agreement-based selection, which selects the candidate with minimum average distance to others; (iii) a Kalman-style constant-velocity selector favoring temporal smoothness; and (iv) a minimum-acceleration selector that chooses the candidate closest to constant-velocity extrapolation from the previous two frames. We use four teacher models in the ensemble (Track-On2, BootsTAPNext, BootsTAPIR, and CoTracker3), whose predictions often disagree on challenging frames, making selection non-trivial. While spatial consensus methods outperform temporal heuristics, the learned verifier consistently surpasses all fixed strategies, with the largest margins observed in challenging settings such as EgoPoints. This indicates that rigid heuristics fail to capture diverse failure modes, whereas the verifier adapts to varying error patterns.

The $\mathcal{D}_{\text{real}}$ collection. To analyze the impact of the real-world dataset $\mathcal{D}_{\text{real}}$ used for fine-tuning, we compare models trained on TAO only (2921 videos) and on the full collection of TAO, OVIS, and VSPW described in Sec. 4.5 (4864 videos), as summarized in Table 6. For this experiment, we use only real-world data without additional synthetic supervision. We observe that adding OVIS and VSPW on top of TAO yields only marginal improvements, with most gains below 0.2 points. In some cases, the TAO-only model performs slightly better (*e.g.*, OA on EgoPoints). These results suggest that most of the adaptation benefit can already be achieved with fewer than 3K real videos and short fine-tuning, highlighting the efficiency of our approach for domain adaptation.

Table 5. **Non-learning ensemble baselines vs. verifier.** Comparison of fixed ensemble heuristics and the learned verifier on four benchmarks measured by δ^x_{avg} .

Ensemble	EgoPoints	RoboTAP	Kinetics	DAVIS
Geometric median	60.1	81.7	70.3	80.6
Agreement-based	61.8	81.8	70.4	80.5
Kalman constant-vel.	52.1	76.5	65.8	72.8
Minimum acceleration	54.0	78.3	67.2	77.8
Verifier	64.8	82.8	71.2	80.8

Table 6. **Effect of the $\mathcal{D}_{\text{real}}$ dataset composition.** We compare fine-tuning using TAO only (2921 videos) versus the full collection of TAO, OVIS, and VSPW (4864 videos).

$\mathcal{D}_{\text{real}}$	Size	EgoPoints		RoboTAP		Kinetics		DAVIS	
		δ^x_{avg}	OA	δ^x_{avg}	OA	δ^x_{avg}	OA	δ^x_{avg}	OA
TAO	2.9K	66.1	**91.1**	82.6	93.6	70.8	90.2	**80.3**	92.3
Mix	4.9K	**66.9**	90.2	**82.7**	**93.8**	**70.9**	**90.4**	**80.3**	**92.4**

10. Visualizations

We visualize examples of the verifier's per-frame selection behavior on videos from the TAP-Vid Kinetics dataset in Fig. 6. Each row corresponds to a different video and illustrates how the verifier dynamically assigns reliability scores to candidate predictions over time. For visualization, we uniformly sample frames among the visible ones and show a 50×50 crop centered at the ground-truth point. Predictions from teacher models are shown as colored dots, the ground-truth location is marked with a star, and the verifier scores are listed in the legend, where the selected candidate is highlighted in bold.

First, we observe that the verifier selects different trackers across frames rather than consistently following a single globally strong model, which aligns with its intended design. Second, the verifier assigns higher scores to spatially accurate predictions while suppressing incorrect ones, *e.g.* BootsTAPNext, CoTracker3, and Anthro-LocoTrack in the first example (top row).

Even when the selected prediction is not strictly the best among the available candidates, the verifier tends to identify the group of reliable predictions and selects among them. For instance, in the second example (middle row), the verifier chooses between BootsTAPNext, BootsTAPIR, and AllTracker, which all remain close to the ground-truth. Although the exact ranking between these models may differ by a few pixels, the verifier consistently favors this reliable group while avoiding large tracking errors.

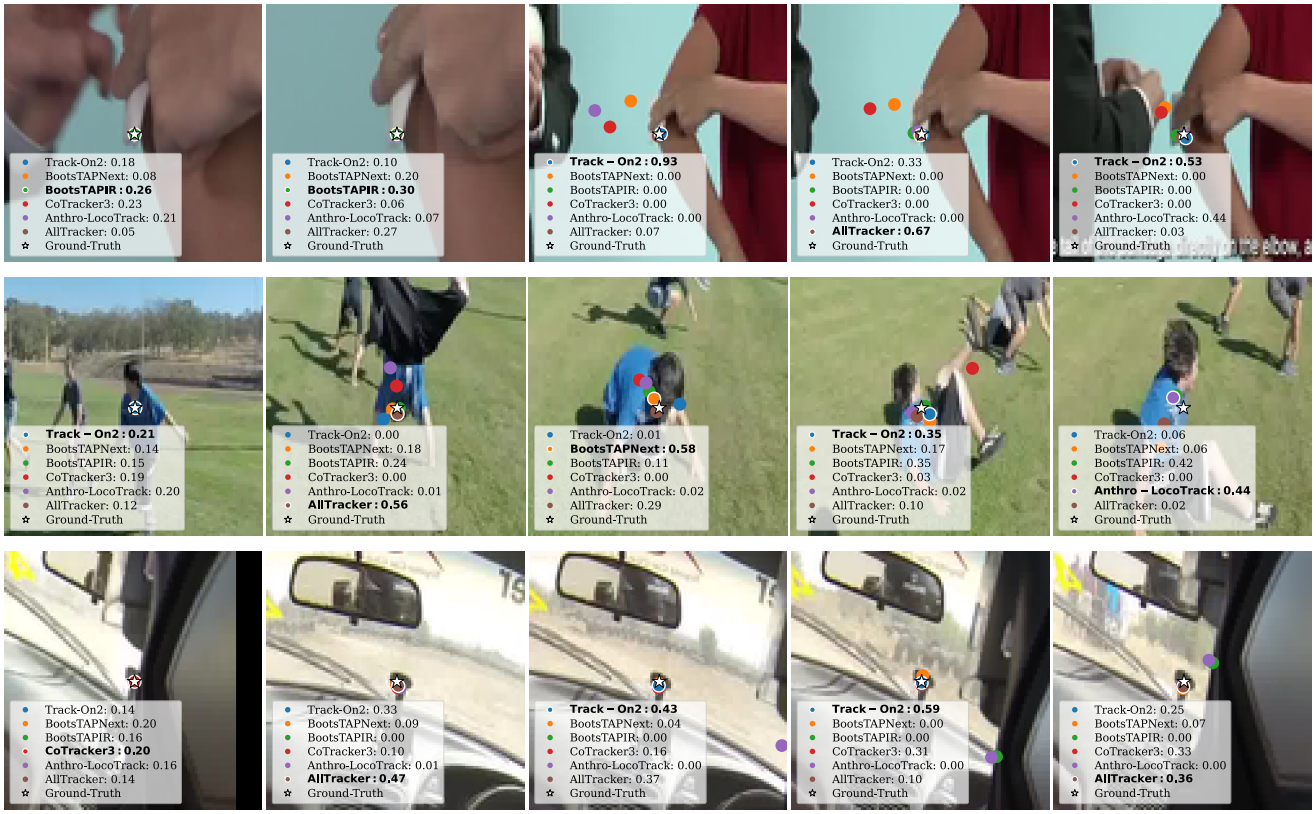

Figure 6. **Verifier selection behavior across videos.** Each row corresponds to a different video from TAP-Vid Kinetics. Frames are uniformly sampled among visible ones, and a 50×50 crop centered at the ground-truth point is shown. Colored dots indicate predictions from teacher trackers, the star marks the ground-truth location, and the legend lists the verifier reliability scores, with the selected candidate highlighted in bold. The verifier adaptively switches between trackers across frames, assigning higher scores to spatially accurate predictions while suppressing unreliable ones.