DINO-Tracker: Taming DINO for Self-Supervised Point Tracking in a Single Video

Supplementary Material (SM)

A Implementation Details

A.1 Preprocessing

Optical flow. As discussed in Sec. 3.2, our method chains RAFT optical flow [8] between consecutive frames, forming short-term accurate tracks for supervision. Specifically, for a given point $\mathbf{x}^i$ in frame $\mathbf{I}^i$, we generate a tracklet $\{\mathbf{x}^j = \mathbf{x}^{j-1} + \mathbf{f}_{j-1\to j}(\mathbf{x}^{j-1}); j \in \{i+1, ..., t\}\}$, where $\mathbf{f}_{j-1\to j}$ is the optical flow between frames $\mathbf{I}^{j-1}$ and $\mathbf{I}^j$. We terminate the track at a frame t if $||\mathbf{x}^t - (\mathbf{x}^{t+1} + \mathbf{f}_{t+1\to t}(\mathbf{x}^{t+1}))|| \geq \gamma_{\texttt{of}}$, where $\gamma_{\texttt{of}} = 1.5$px is a cycle-consistency threshold. To avoid drift error, we apply cycle-consistency checks on optical flow between distant frames. That is, we filter-out correspondences $\mathbf{x}^j$ that are inconsistent with $\mathbf{f}_{i\to j}$, i.e. if $||\mathbf{x}^j - \mathbf{x}^{i\to j}||_2 \geq \gamma_{\texttt{of-lng}}$ and $||\mathbf{x}^i - (\mathbf{x}^{i\to j} + \mathbf{f}_{j\to i}(\mathbf{x}^{i\to j}))||_2 \leq \gamma_{\texttt{of}}$, where $\mathbf{x}^{i\to j} = \mathbf{x}^i + \mathbf{f}_{i\to j}(\mathbf{x}^i)$, $\gamma_{\texttt{of-lng}} = 2$px, and the second condition ensures that $\mathbf{x}^{i\to j}$ is reliable. For each frame $\mathbf{I}^i$, we initialize new tracklets for points that do not have correspondences. The set of all correspondences processed from the optical flow is denoted as $\Omega_{\texttt{flow}} = \{(\mathbf{x}^i, \mathbf{x}^j)\}$. In all our losses, continuous coordinates are being normalized to $[-1, 1]$.

DINO feature correspondences. Since the *coarse* feature correspondence supervision complements the *sub-pixel* optical flow supervision, we discard feature correspondences for which optical flow supervision is available: $\Omega_{\texttt{dino-bb}} = \{(\mathbf{p}^i, \mathbf{p}^j)$ DINO best-buddy : $(\mathbf{x}^i, \mathbf{x}^j) \notin \Omega_{\texttt{flow}}\}$. In Fig. A1, we visualize DINO best-buddy pairs extracted between distant frames. As seen, DINO best-buddies provides localized, semantic correspondences across multiple occlusions.

A.2 Training details

Minibatch sampling. For memory efficiency, we sample correspondences from a set of 8 frames in each training batch. We sample 512 pairs of optical flow correspondences, at most 1024 pairs of best-buddy features (for $\mathcal{L}_{\texttt{dino-bb}}$ and $\mathcal{L}_{\texttt{rfn-bb}}$ separately), and at most 1024 cycle-consistent correspondences. The best-buddy and cycle-consistent correspondences are sampled between 4 pairs of frames. For balanced training, we ensure that 50% of the optical flow correspondences and 70% of feature and cycle-consistent correspondences lie in the foreground. We use saliency maps of DINOv2 features [1] for detecting the foreground when ground-truth masks are not available.

Fig. A1: *DINO best-buddies.* We visualize best-buddy pairs between distant frames. DINO best-buddies provide localized semantic correspondences, allowing the model to recover the object past repeating occlusions.

Contrastive loss weighting. As discussed in Sec. 3.3 of the paper, each best-buddy term in $\mathcal{L}_{\texttt{dino-bb}}$ is weighted with a confidence score. For a given pair $\{\boldsymbol{\varphi}^i_{\texttt{DINO}}, \boldsymbol{\varphi}^j_{\texttt{DINO}}\}$, we measure the confidence score based on 2 metrics: (i) the unimodality of the correlations $\{\mathbf{S}(\mathbf{p}) = \text{cos-sim}(\boldsymbol{\varphi}^i_{\texttt{DINO}}, \boldsymbol{\Phi}^j(\mathbf{p})) : \mathbf{p} \in H' \times W'\}$, (ii) the correlation of the pair $s^{ij} = \mathbf{S}(\mathbf{p}^j)$. To measure (i), we compute the ratio $r_{ij} = s_2/s_1$, where $s_1 > s_2$ are the 2 highest correlations in $\mathbf{S}$. To detect them, we apply non-maximum suppression (NMS) [4] on the similarity map $\mathbf{S}$ with an IoU threshold of 0.2, where we use a box size of 60px for each position. s_1 and s_2 are, therefore, the top 2 similarities proposed by NMS. Thus, our confidence score is given by $w^{ij}_{\texttt{dino-bb}} = \sigma(a \cdot (1 - \max(r_{ij}, r_{ji})) - b) \cdot 2(s^{ij})^3$, where $\sigma(\cdot)$ is the sigmoid function. We fix $a = 27, b = -5.7$ in all our experiments.

For each best-buddy pair $\{\mathbf{p}^i, \mathbf{p}^j\}$ in $\mathcal{L}_{\texttt{rfn-bb}}$, we weight the term based on the correlation between the features: $w^{ij}_{\texttt{rfn-bb}} = 2(s^{ij})^3$, where $s^{ij} = \text{cos-sim}(\boldsymbol{\varphi}_i, \boldsymbol{\varphi}_j)$.

Cycle-consistency loss. We noticed that $\mathcal{L}_{\texttt{rfn-cc}}$ in the main paper misses a coefficient $w^{ij}_{\texttt{rfn-cc}}$ for each pair $\{\mathbf{x}^i, \mathbf{x}^j\}$, which weights the loss according to the cycle error of the pair: $w^{ij}_{\texttt{rfn-cc}} = 0.8^{e_{\texttt{cyc}}}$, where $e_{\texttt{cyc}} = ||\mathbf{x}^i - \Pi(\mathbf{x}^j, i)||_2$. Thus, the final form of our cycle-consistency loss is:

$$\mathcal{L}_{\texttt{rfn-cc}} = \frac{1}{|\Omega_{\texttt{rfn-cc}}|} \sum_{(\mathbf{x}^i, \mathbf{x}^j) \in \Omega_{\texttt{rfn-cc}}} \frac{1}{2} w^{ij}_{\texttt{rfn-cc}} \left(L_H(\Pi(\mathbf{x}^i, j), \mathbf{x}^j) + L_H(\Pi(\mathbf{x}^j, i), \mathbf{x}^i) \right)$$

Hyperparameters. We train our model using Adam optimizer [5], with a learning rate of 0.01 for all parameters. We decrease the learning rate of the CNN-refiner (Fig. 2) by a factor of 0.999 every 40 step. For videos of up to 100 frames, the

model is trained for 10K iterations. On Kinetics, which contains longer videos (250 frames), we train for 20k iterations. We apply the losses $\mathcal{L}_{\texttt{rfn-bb}}$ and $\mathcal{L}_{\texttt{rfn-cc}}$ after 5k training iterations. The radius R in Eq. 2 is set to 35px. In $\mathcal{L}_{\texttt{rfn-bb}}$ and $\mathcal{L}_{\texttt{dino-bb}}$, we set the temperature $\tau = 0.1$. In $\mathcal{L}_{\texttt{rfn-cc}}$, we use an error threshold $\gamma = 4$. In all our experiments, we use the following weighting in Eq. 4: $\lambda_1 = 25 \times 10^{-5}, \lambda_2 = 5 \times 10^{-5}, \lambda_3 = 0.5, \lambda_4 = 1 \times 10^{-4}$.

A.3 Architecture

Delta-DINO is a fully convolutional neural network. It comprises 4 layers with channel dimensions of $[3 \rightarrow 64 \rightarrow 128 \rightarrow 256 \rightarrow 1024]$. All layers comprise `Conv2d` $\rightarrow$ `BatchNorm2d` $\rightarrow$ `ReLU` $\rightarrow$ `BlurPool`, except for the last layer, which comprises `Conv2d` $\rightarrow$ `BatchNorm2d`. For `BlurPool`, we use the antialiased downsampling layers from [9]. All convolutional layers have kernel size 5, stride of 1, and reflection padding of 2, except the last layer has reflection padding of 4 and a dilation of 2. To align the residual features with DINO features, we grid-sample from the output of Delta-DINO at the DINO patch-center positions.

CNN-Refiner [3] comprises of `Conv2d` $\rightarrow$ `ReLU` $\rightarrow$ `Conv2d` with channels $[1 \rightarrow 16 \rightarrow 1]$, kernel size 3, and padding 1.

Our model has $\sim$7.6M trainable parameters: $\sim$7.59M for Delta-DINO, $\sim$300 for CNN-Refiner. We use DINOv2-ViTL/14 [7] as the DINO backbone in all our experiments. To increase the resolution of DINO features, we modify the stride of the embedding projection layer from 14 to 7 [1].

A.4 Occlusion Prediction

We select the anchor frames $\{k_i\}$ based on high cos-similarity between query and tracked features: $\{k_i : \text{cos-sim}(\boldsymbol{\varphi}^{k_i}, \boldsymbol{\varphi}_{\mathbf{q}}) \geq 0.7\}$. To predict occlusion from trajectory agreement, we calculate an agreement threshold for the trajectory $\mathcal{T}_{\mathbf{q}}$: for each anchor frame k, we sample the median disagreement w.r.t. other anchor frames: $e_k = \text{med}_{k_i}(||\Pi(\hat{\mathbf{x}}^k, k_i) - \hat{\mathbf{x}}^{k_i}||_2)$, and take the maximum of the median errors as the threshold for $\mathcal{T}_{\mathbf{q}}$: $e_{\mathbf{q}} = \max_k(e_k)$. A tracked point $\hat{\mathbf{x}}^t$ is predicted as visible if $\text{med}(d_k) \leq e_{\mathbf{q}} \ \wedge \ \text{cos-sim}(\boldsymbol{\varphi}^t, \boldsymbol{\varphi}_{\mathbf{q}}) \geq \gamma_{\texttt{occ}}$, where $d_k = ||\Pi(\mathbf{x}_{\mathbf{q}}, k) - \Pi(\hat{\mathbf{x}}^t, k)||_2$, and $\gamma_{\texttt{occ}} = 0.6$ in all experiments.

A.5 Ablation Details

LoRA tuning. We use the PEFT implementation [6] for LoRA. We fine-tune the queries, keys, and values of layers-$\{15, 16\}$ of DINOv2 since we use layer-16 in our tracker (see Tab. 3). We set `lora_alpha=0.5, lora_dropout=0.1, rank=8` when fine-tuning with PEFT.

Raw DINOv2 tracking. To track with raw DINOv2 features (see Sec. 4 of the paper), we use the tracking algorithm described in Sec. 3.1 and Eq. 2, while setting $\boldsymbol{\Phi}_{\Delta}(\mathbf{I}) = \mathbf{0}$ and $\mathbf{H} = \mathbf{S}$ (i.e. without Delta-DINO and CNN-Refiner).

A.6 Benchmarks Evaluation

On the TAP-Vid benchmark we evaluate all methods using "query-strided" sampling, where points on the annotated tracks are sampled as query every five frames [3]. All metrics on the TAP-Vid benchmark are computed in 256x256 resolution. BADJA [2] provides key-point position and visibility labels every 3-5 frames. For evaluation, points are sampled once, at their first visible frame. For the dotted visualizations shown in Fig. 4 and the supplementary website, we track a dense grid of points on the query frame, and visualize only tracks that lie on the foreground.

We follow PIPs++ and Co-Tracker's evaluation protocol, and resize frames to their training resolution of 384x512 and 512x896 respectively, before inference. We provide Co-Tracker's query points a support of 6 global and 6 local grid points. TAP-IR and TAP-Net are evaluated at the provided input resolution. For the RAFT baseline, we found that upsamling frames from 256 $\times$ 256 improves performance on the TAP-Vid-DAVIS-256 and TAP-Vid-Kinetics-256 benchmarks, and we resize downsampled frames to 480x854 before inference. We used Omnimotion's published code to train models for the TAP-Vid-DAVIS-480 and BADJA benchmarks, pre-trained weights were provided for TAP-Vid-DAVIS-256 and TAP-Vid-Kinetics-256.

B Additional ablations

Table A1: *Ablating optical flow* reduces the performance of our framework in all metrics.

	DAVIS-480		
	δ^x_{avg}	OA	AJ
w/o $\mathcal{L}_{\texttt{dino-bb}}$	78.2	87.0	61.9
w/o $\mathcal{L}_{\texttt{flow}}$	78.3	87.2	62.0
Ours	**80.4**	**88.0**	**64.6**

We additionally ablate the optical flow loss in our framework. As reported in Tab. A1, the performance decreases for all metrics in the TAP-Vid-DAVIS benchmark. The decrease in performance for w/o $\mathcal{L}_{\texttt{flow}}$ and w/o $\mathcal{L}_{\texttt{dino-bb}}$ baselines verifies that our framework benefits from both types of correspondences. Interestingly, w/o $\mathcal{L}_{\texttt{flow}}$ reduces positional accuracy only by 2%. This shows the effectiveness of combining DINO prior with our self-supervision and feature refinement for accurate tracking.

C Complexity.

C.1 Training time.

Fitting DINO-Tracker to a single video with 100 frames takes about 1.6 hours (less than a second per iteration) on a single A100 GPU. Our training time is $\times$10 faster than Omnimotion for the same video. Training LoRA-tune baseline (see Sec. 4.2) with the same settings takes almost 9 hours per video (about 1.5 sec/iteration). This is $\times$6 slower than our CNN-based refiner network.

To improve training efficiency, we show that DINO-Tracker can be trained only on a subset of the frames, while still evaluating on *all* frames during inference. Tab. A2 reports our performance when training only 50% or 25% of the frames, thus reducing the training time by the same factor. As seen, our method maintains its performance when trained on 50% and is competitive when trained on 25% of the frames. This demonstrates the ability of our tracker to generalize to *unseen* frames and suggest it can be extended efficiently to longer videos.

Table A2: *Generalization when training on every 2nd and every 4th frame in each video (TAP-Vid-DAVIS-480) on a single A100.*

	δ^x_{avg}	AJ	OA	train time
Ours	80.7	65.3	88.5	90 min.
Ours, every 2nd frame	80.6	65.7	88.5	45 min.
Ours, every 4th frame	79.1	64.0	87.2	22.5 min.

C.2 Runtime and Memory

We measure the required compute for *full* inference (both position and visibility) of DINO-Tracker and feed-forward competitors. Tab. A3 reports average runtime and allocated memory on TAP-Vid-DAVIS-480 on a single A100 for our tracker and feed-forward methods. Most of our runtime is used for visibility prediction, yet once trained, our total inference time is fastest (note that PIPS++ cannot predict visibility) and is less memory-consuming than TAPIR.

Table A3: *DAVIS-480 inference time and memory on a single A100.*

	Co-Tracker (full)	TAPIR (full)	PIPS++ (pos. only)	Ours (pos. only)	Ours (full)
Time (sec)	345.8	110.4	8.6	4.3	80.5
GPU-Mem (GB)	19.2	60.0	12.0	15.2	52.6

References

1. Amir, S., Gandelsman, Y., Bagon, S., Dekel, T.: Deep vit features as dense visual descriptors. ECCVW What is Motion For? (2022)
2. Biggs, B., Roddick, T., Fitzgibbon, A., Cipolla, R.: Creatures great and SMAL: Recovering the shape and motion of animals from video. In: ACCV (2018)
3. Doersch, C., Gupta, A., Markeeva, L., Continente, A.R., Smaira, K., Aytar, Y., Carreira, J., Zisserman, A., Yang, Y.: Tap-vid: A benchmark for tracking any point in a video. In: NeurIPS Datasets Track (2022)
4. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. IEEE Transactions on Pattern Analysis and Machine Intelligence **32**(9), 1627–1645 (2010)
5. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015)
6. Mangrulkar, S., Gugger, S., Debut, L., Belkada, Y., Paul, S., Bossan, B.: Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft` (2022)
7. Oquab, M., Darcet, T., Moutakanni, T., Vo, H.V., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Howes, R., Huang, P.Y., Xu, H., Sharma, V., Li, S.W., Galuba, W., Rabbat, M., Assran, M., Ballas, N., Synnaeve, G., Misra, I., Jegou, H., Mairal, J., Labatut, P., Joulin, A., Bojanowski, P.: Dinov2: Learning robust visual features without supervision (2023)
8. Teed, Z., Deng, J.: Raft: Recurrent all-pairs field transforms for optical flow. In: European Conference on Computer Vision (ECCV). pp. 402–419 (2020)
9. Zhang, R.: Making convolutional networks shift-invariant again. In: ICML (2019)