Tracking Everything Everywhere All At Once –Supplementary Material–

Qianqian Wang[1,2] Yen-Yu Chang[1] Ruojin Cai[1] Zhengqi Li[2]
Bharath Hariharan[1] Aleksander Holynski[2,3] Noah Snavely[1,2]

[1]Cornell University [2]Google Research [3]UC Berkeley

In Sec. 1, we introduce our approach for generating pairwise correspondences, which serves as a noisy supervision signal for optimizing OmniMotion. In Sec. 2, we present additional ablation experiments, and in Sec. 3, we provide supplementary implementation details.

1. Preparing pairwise correspondences

Our method uses pairwise correspondences from existing methods, such as RAFT [8] and TAP-Net [3], and consolidates them into dense, globally consistent, and accurate correspondences that span an entire video. As a preprocessing stage, we exhaustively compute all pairwise correspondences (i.e., between every pair of frames i and j) and filter them using cycle consistency and appearance consistency checks.

When computing the flow field between a base frame i and a target frame j as $i \to j$, we always use the flow prediction for the previous target frame ($i \to j-1$) as initialization for the optical flow model (when possible). We find this improves flow predictions between distant frames. Still, the flow predictions between distant frames can contain significant errors, and therefore we filter out flow vector estimates with cycle consistency errors (i.e., forward-backward flow consistency error) greater than 3 pixels.

Despite this filtering process, we still frequently observe a persistent type of error that remains undetected by cycle consistency checks. This type of spurious correspondence, illustrated in Fig. 5, occurs because flow networks can struggle to estimate motion for regions that undergo significant deformation between the two input frames, and instead opt to interpolate motion from the surrounding areas. In the example in Fig. 5, this leads to flow on the foreground person "locking on" to the background layer instead. This behavior results in incorrect flows that survive the cycle consistency check, since they are consistent with a secondary layer's motion (e.g., background motion). To address this issue, we additionally use an appearance check: we extract dense features for each pixel using DINO [2] and filter out correspondences whose features' cosine similarity is < 0.5. In practice, we apply the cycle consistency check for all pairwise flows and supplement it with an appearance check

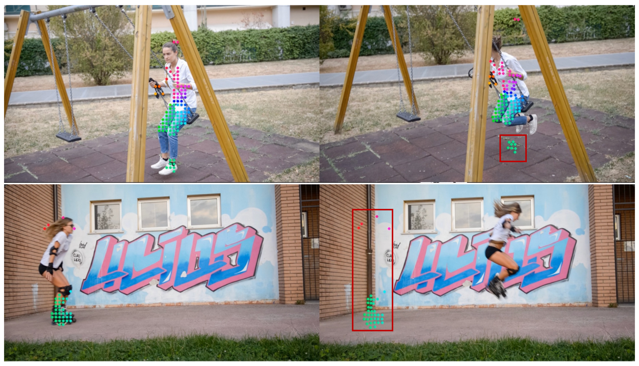

Figure 5: *Erroneous correspondences after cycle consistency check.* The red bounding box highlights a common type of incorrect correspondences from flow networks like RAFT [8] that remains undetected by cycle consistency check. The left images are query frames with query points and the right images are target frames with the corresponding predictions. Only correspondences on the foreground object are shown for better clarity.

when the two frames are more than 3 frames apart. We found this filtering process consistently eliminates major errors in flow fields across different sequences without per-sequence tuning. The results of our filtering approach, after both cycle and appearance consistency checks, are illustrated in Fig. 6.

One drawback of such a filtering process is that it will also remove correct flows for regions that become occluded in the target frame. For certain correspondence methods (such as RAFT), including these motion signals during occlusion events can result in better final motion estimates. Therefore, we devise a simple strategy for detecting reliable flow in occluded regions. For each pixel, we compute its forward flow to a target frame (a), cycle flow (flow back to the source frame from the target pixel) (b), and a second forward flow (c). This process effectively amounts to a 2-pass cycle consistency check: the consistency between (a) and (b) forms a standard cycle consistency check, and the consistency between (b) and (c) forms a secondary, supplementary one. We identify pixels where (a) and (b) are inconsistent but (b) and (c) are consistent and deem these to be occluded

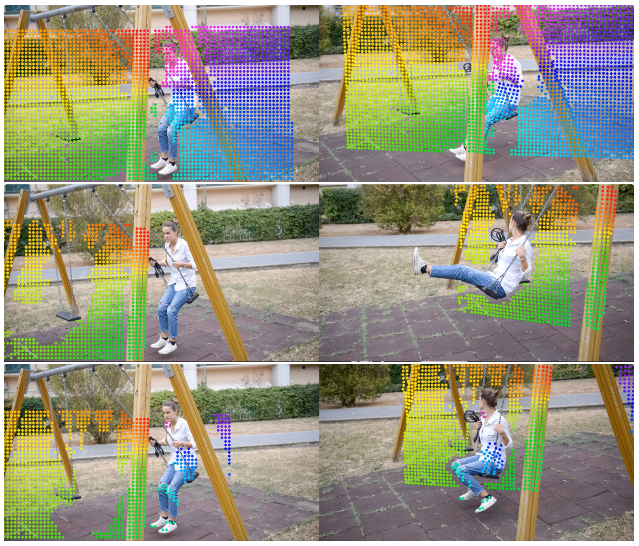

Figure 6: *Correspondences from RAFT [8] after both cycle and appearance checks*. The left column shows a single query frame, and the right column displays target frames with increasing frame distances to the query frame from top to bottom. The filtered correspondences are reliable without significant errors.

pixels. We found this approach effective in identifying reliable flows for occluded regions—particularly when the two frames are close to each other. Therefore, we allow these correspondences to bypass cycle consistency checks if they span a temporal distance of less than 3 frames. Our experiments use this added signal for the variant of our method that uses RAFT flow, but not for the TAP-Net variant, as we found the predicted correspondences from the latter were less reliable near occlusion events.

We can also optionally augment the supervising input flow by chaining sequences of correspondences that are deemed reliable (i.e., those that satisfy the cycle consistency and appearance consistency checks). This helps densify the set of correspondences, creating supervision between distant frames where the direct flow estimates were deemed unreliable and therefore discarded during filtering. We found this process to be beneficial especially for challenging sequences with rapid motion or large displacements, where optical flow estimates between non-adjacent frames are less reliable.

2. Additional ablations

In addition to the ablations in the main paper, we provide the following ablations and report the results in Table 1: 1) *Plain 2D*: Rather than using a quasi-3D representation with bijections to model motion, we utilized a simple 8-layer MLP with 256 neurons that takes the query pixel location, query time, and target time as input and outputs the corresponding location in the target frame. Although we applied positional encoding with 8 frequencies to the input to enable better fitting, this ablation failed to capture the holistic motion of the video, instead only capturing simpler motion patterns for the rigid background. 2) *No flow loss*: we remove the flow loss and only rely on photometric information for training. We find this approach is effective only for sequences with small motion, where a photometric loss can provide useful signals to adjust motion locally. For sequences with relatively large motion, this method fails to provide correct results. 3) We also vary the number of samples K for each ray from 32 to 16 and 8. The resulting ablations, named *#Samples* K*=8* and *#Samples* K*=16*, demonstrate that using a denser sampling strategy tends to produce better results.

Method	AJ ↑	$< \delta^x_{\text{avg}}$ ↑	OA ↑	TC ↓
Plain 2D	11.6	19.8	76.7	1.25
No invertible	12.5	21.4	76.5	0.97
No flow loss	23.9	37.3	70.8	1.75
No photometric	42.3	58.3	84.1	0.83
Uniform sampling	47.8	61.8	83.6	0.88
#Samples K = 8	48.1	63.5	84.6	0.75
#Samples K = 16	49.7	65.0	**85.6**	0.84
Full	**51.7**	**67.5**	85.3	**0.74**

Table 1: Ablation study on DAVIS [6].

3. Additional implementation details

We provide additional implementation details below and will release our code upon acceptance.

Error map sampling. We cache the flow predictions generated by our model every 20k steps and use them to mine hard examples for effective training. Specifically, for each frame in the video sequence, we compute the optical flow between that frame and its subsequent frame, except for the final frame where we compute the flow between it and the previous frame. We then compute the L_2 distance between the predicted flow and supervising input flow, where each pixel in the video is now associated with a flow error. In each training batch, we randomly sampled half of the query pixels using weights proportional to the flow errors and the other half using uniform sampling weights.

Training details. In addition to the photometric loss $\mathcal{L}_{\text{pho}}$ introduced in the main paper, we include an auxiliary loss term that supervises the *relative* color between a pair of pixels in a frame:

$$\mathcal{L}_{\text{pgrad}} = \sum_{\Omega_p} ||(\hat{\boldsymbol{C}}_i(\boldsymbol{p}_1) - \hat{\boldsymbol{C}}_i(\boldsymbol{p}_2)) - (\boldsymbol{C}_i(\boldsymbol{p}_1) - \boldsymbol{C}_i(\boldsymbol{p}_2))||_1 \tag{1}$$

Here, $(\hat{\boldsymbol{C}}_i(\boldsymbol{p}_1) - \hat{\boldsymbol{C}}_i(\boldsymbol{p}_2))$ is the difference in predicted color between a pair of pixels, and $(\boldsymbol{C}_i(\boldsymbol{p}_1) - \boldsymbol{C}_i(\boldsymbol{p}_2))$ is the corresponding difference between ground-truth observations. This loss is akin to spatial smoothness regularizations or gradient losses that supplement pixel reconstruction losses in prior work [4, 7], but instead computed between pairs of randomly sampled, potentially distant pixels $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$, rather than between adjacent pixels. We apply the same gradient loss to the flow prediction as well. We found that including these gradient losses helps improve the spatial consistency of estimates, and more generally improves the training process. We also use distortion loss introduced in mip-NeRF 360 [1] to suppress floaters.

We train our network with the Adam optimizer with base learning rates of 3×10^{-4}, 1×10^{-4}, and 1×10^{-3} for the density/color network $F_{\boldsymbol{\theta}}$, the mapping network $M_{\boldsymbol{\theta}}$, and the MLP that computes the latent code, respectively. We decrease the learning rate by a factor of 0.5 every 20k step. To select correspondences during training, we begin by sampling correspondences from pairs of frames with a maximum interval of 20, and gradually increase the window size during training. Specifically, we expand the window by one every 2k steps.

In our loss formulation, we compute the flow loss $\mathcal{L}_{\text{flo}}$ as a weighted sum of the mean absolute error (MAE) between each pair of correspondences in a training batch. The weight is determined by the frame interval, and is given by $w = 1/\cos(\Delta/N' \cdot \pi/2)$, where Δ is the frame interval, and N' is the current window size. The coefficient λ_{pho} for the photometric loss initially starts at 0 and linearly increases to 10 over the first 50k steps of training. After 50k steps, λ_{pho} stays fixed at 10. This design is motivated by our observation that the photometric loss is not effective in fixing large motion errors early on in the training process, but is effective in refining the motion. The coefficient λ_{reg} for smoothness regularization is set to 20. We use the same set of network architecture and training hyperparameters when evaluating different datasets in the TAP-Net benchmark.

When sampling on each ray, we use a stratified sampling strategy and sample $K = 32$ points on each ray between the near and far depth range. Additionally, when mapping a 3D location from one local volume to another, we encourage it to be mapped within our predefined depth range to avoid degenerate solutions.

During training, we use alpha compositing to propagate the training signal to all samples along a ray. However, at inference time, we instead compute the corresponding location using the single sample with the largest alpha value, which we found to produce quantitatively similar but visually better results.

Network architecture for $M_{\boldsymbol{\theta}}$. We illustrate the architecture for our invertible network $M_{\boldsymbol{\theta}}$ that maps between local and canonical coordinate frames in Fig. 7. $M_{\boldsymbol{\theta}}$ is comprised of six affine coupling layers with alternating split patterns (only the first layer is highlighted in Fig. 7). The learnable component in each affine coupling layer is an MLP that computes a scale and a translation from a frame latent

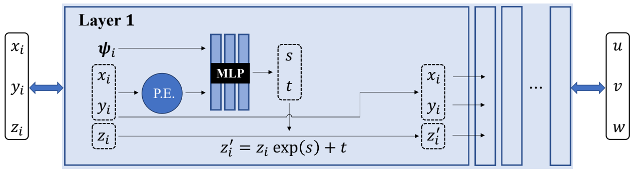

Figure 7: *Network architecture for the mapping network $M_{\boldsymbol{\theta}}$.* We show the first affine coupling layer, which is representative of the subsequent layers, except for the different splitting patterns used. As mentioned in the main paper, this architecture is fully invertible, i.e., it can be queried in either direction, from (u, v, w) to (x, y, z) and vice-versa.

code $\boldsymbol{\psi}_i$ and the first part of the input coordinates. This scale and translation is then applied to the second part of the input coordinate. This process subsequently is repeated for each of the other coordinates. The MLP network in each affine coupling layer has 3 layers with 256 channels. We found that applying positional encoding [5] to the MLP's input coordinates improved its fitting ability, and we set the number of frequencies to 4.

Deformable sprites evaluation. Because the Deformable Sprites method defines directional mappings from image space to atlas space, we must approximate the inverses of these mappings in order to establish corresponding point estimates between pairs of frames. We do this by performing a nearest neighbor search: all points in the target frame are mapped to the atlas, and the closest atlas coordinate to the source point's mapping is chosen as the corresponding pixel. Furthermore, occlusion estimates are extracted using the following process: (1) initialize the layer assignment of source point tracks to the layer which has the higher opacity at the source frame, (2) at a given target frame index, denote the point as occluded if its originally assigned layer has lower opacity than the other layer.

References

[1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 5470–5479, 2022. 3

[2] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 9650–9660, 2021. 1

[3] Carl Doersch, Ankush Gupta, Larisa Markeeva, Adria Recasens Continente, Kucas Smaira, Yusuf Aytar, Joao Carreira, Andrew Zisserman, and Yi Yang. Tap-vid: A benchmark for tracking any point in a video. In *NeurIPS Datasets Track*, 2022. 1

[4] Yoni Kasten, Dolev Ofri, Oliver Wang, and Tali Dekel. Layered neural atlases for consistent video editing. In *ACM Trans. Graphics (SIGGRAPH Asia)*, 2021. 3

[5] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 3

[6] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 DAVIS challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017. 2

[7] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*, 44(3):1623–1637, 2020. 3

[8] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Proc. European Conf. on Computer Vision (ECCV)*, pages 402–419, 2020. 1, 2