



Decomposition Betters Tracking Everything Everywhere

Rui Li 

Dong Liu 

University of Science and Technology of China, Hefei, China
liruid@mail.ustc.edu.cn, dongeliu@ustc.edu.cn
<https://github.com/qianduoduolr/DecoMotion>

Abstract. Recent studies on motion estimation have advocated an optimized motion representation that is globally consistent across the entire video, preferably for every pixel. This is challenging as a uniform representation may not account for the complex and diverse motion and appearance of natural videos. We address this problem and propose a new test-time optimization method, named DecoMotion, for estimating per-pixel and long-range motion. DecoMotion explicitly decomposes video content into static scenes and dynamic objects, either of which uses a quasi-3D canonical volume to represent. DecoMotion separately coordinates the transformations between local and canonical spaces, facilitating an affine transformation for the static scene that corresponds to camera motion. For the dynamic volume, DecoMotion leverages discriminative and temporally consistent features to rectify the non-rigid transformation. The two volumes are finally fused to fully represent motion and appearance. This divide-and-conquer strategy leads to more robust tracking through occlusions and deformations and meanwhile obtains decomposed appearances. We conduct evaluations on the TAP-Vid benchmark. The results demonstrate our method boosts the point-tracking accuracy by a large margin and performs on par with some state-of-the-art dedicated point-tracking solutions.

Keywords: Decomposition · Motion-estimation · Point-tracking

1 Introduction

Video content comprises two primary components: dynamic and static scenes. In dynamic scenes (also denoted as dynamic objects), objects exhibit movement or change over time, and the motion between video frames results from a combination of camera movement and the object’s inherent motion. Conversely, in static scenes, objects remain unchanged, and inter-frame motion is solely influenced by the camera’s rigid movement. Compared to static scenes, dynamic scenes are more intricate in terms of both motion and appearance.

However, the advanced studies for establishing point correspondences in the last few years have always dealt with different motions in two scenes with a unified framework. For example, feature matching methods [17, 37, 40] try to design

learning methods for the dense features. Despite achieving good performance on tasks involving dynamic object tracking, these methods struggle to model consistent motion in static scenes. Dense optical flow estimation methods regard pixel correspondences as a regression problem, and usually train a convolutional neural network on a rendered video dataset [11], which are further improved by constructing a pyramid cost volume [32], iterative inference [33], and multi-frame prediction [14, 30]. However, these methods still show unsatisfactory robustness in real-world videos, especially on dynamic objects.

In a recent study, OmniMotion [36] proposes a novel test-time optimization method, which utilizes reliable correspondences and RGB frames as supervision to learn a globally consistent neural radiance field [26] with learned transformation. The representation stores entire color and quasi-3D geometry information in canonical space, along with transformation can establish long-range pixel trajectories even facing occlusions. While promising, the method is optimized with a unified neural field and transformation, which results in sub-optimal performance of estimating the highly non-rigid motion of moving objects.

In this work, intending to learn accurate point correspondences, we propose a decoupled representation that divides static scenes and dynamic objects in terms of motion and appearance. Leveraging the significant difference between them, we explicitly build individual 3D neural radiance fields for representation, and we carefully design the transformation and representation functions for each scene. More specifically, for the dynamic objects, taking into account the complex inter-frame motion and dramatic appearance changes, we utilize more non-linear layers to approximate the non-rigid motion. Besides, we additionally encode the features in the field for better representation, along with a feature rendering constraint to alleviate the problem of imprecise correspondences on dynamic objects. For the static scenes, we approach the rigid-motion by a simpler network with affine transformation and model the confidence of whether the 3D points are changing or moving. With this confidence, we further fuse two neural radiance fields and transformations to obtain the final appearance and motion representations. Such a design enables DecoMotion to establish the trajectory of any pixel within the whole video more accurately, especially on dynamic objects. Additionally, the observed appearance decomposition enables its application to video tasks like video inpainting, and object removal.

In summary, the main contribution of this work lies in: (i) We propose an optimized 3D video representation that decomposes video into dynamic objects and static scenes for better tracking any pixel; (ii) For dynamic objects, we propose to encode and render discriminative and temporal consistent features to rectify the non-rigid transformation; (iii) We demonstrate our method quantitatively on the TAP-Vid benchmark [9]. Our method significantly enhances point-tracking accuracy and shows competitive performance.

2 Related work

Optimization-based motion estimation. In the early days, motion estimation results are mostly obtained by optimizing the pairwise non-parametric constraint equations [1, 5, 15], which are improved by optimizing motion globally over an entire video. Particle video [29] generates a collection of semi-dense long-range trajectories based on initial optical flows, while ParticleSfM [43] optimizes long-range rigid-motion derived from pairwise optical flows within a Structure-from-Motion (SfM) framework. By optimizing a globally consistent canonical volume and coordinate transformation with reliable optical flows, OmniMotion [36] tries to deal with the non-rigid motion of dynamic objects. However, the results obtained by this method are not yet satisfactory.

Matching-based motion estimation. By matching between video frames, there are two dominant approaches: feature matching [17, 19, 22, 34, 37] and dense optical flow [11, 16, 32, 33, 39]. In feature matching, for example, by designing the pretext tasks like frame reconstruction [18, 34], cycle-consistent tracking [17, 37], and contrastive learning [40], dense representations are learned for matching in a self-supervised manner. On the other hand, optical flow estimators regard it as a regression problem, and employ the cost volumes to find pixel matching. FlowNet [11] first leverages a rendered video dataset to train a regression model, which further encourages the following methods to improve it by building pyramid cost volume [32], iterative inference [33], and multi-frame prediction [3, 14, 30, 44].

Dynamic neural representation. Our method shares similar spirits with recent studies about dynamic neural representation using coordinate-based neural networks [23, 24, 28]. For instance, Nerfies [28] augments NeRF [26] by optimizing a time-dependent continuous volumetric deformation field that warps each observed point into a canonical 5D NeRF. NSFF [23] proposes to represent a dynamic scene by combining static NeRF with scene flow-based dynamic NeRF, and is further improved by adopting a volumetric image-based rendering framework [24]. However, the studies in this direction are mostly designed for problems such as space-time novel view synthesis, while our method aims to establish long-range accurate motion trajectories.

3 Method

This work presents a decoupled motion representation, we name it as DecoMotion (Decomposing Motion). As follows, we will first revisit the OmniMotion [36], and then further introduce the proposed decoupled representation. The specific optimization process will also be presented in the next sections.

3.1 Representation in OmniMotion

OminiMotion [36] proposes a quasi-3D global motion representation, i.e., a data structure that can encode not only the appearance but also the inter-frame motion of all points in the scene. Such a representation can provide accurate and consistent motion trajectories even when the query pixels are occluded.

Canonical 3D volume. OmniMotion represents the scenes with a canonical 3D volume G . Following NeRF [26], OmniMotion defines a coordinate-based network F_θ that maps any 3D point $u \in G$ to the corresponding color c and density value σ . The density indicates the quasi-3D scene geometry of the video.

Transformation. Meanwhile, OmniMotion defines a continuous invertible transformation \mathcal{T}_i that maps the local 3D points x_i in the local volume L_i to the canonical space: $u = \mathcal{T}_i(x_i)$. The u is a time-independent 3D coordinate and can be considered as a globally consistent index of a particular point in the scene. By combining these invertible mappings, a 3D point can be mapped from one local volume frame L_i to another local volume L_j , i.e:

$$x_j = \mathcal{T}_j^{-1} \circ \mathcal{T}_i(x_i). \quad (1)$$

Motion rendering. Given a global representation G of the video and a transformation \mathcal{T} , OmniMotion predicts the motion by volume rendering. Specifically, for the query pixel p_i , a set of local 3D points $\{x_i^k\}_{k=1}^K$ is sampled along the ray by a fixed, orthographic camera [36]. These points are further mapped to the local volume L_j using the invertible projections \mathcal{T}_i and \mathcal{T}_j . These mapped 3D points $\{x_j^k\}_{k=1}^K$ are finally aggregated with alpha compositing:

$$\hat{x}_j = \sum_{k=1}^K T_k \alpha(\sigma_k) x_j^k, \text{ where } T_k = \prod_{l=1}^{k-1} (1 - \alpha(\sigma_l)). \quad (2)$$

The α value for k -th sample as $\alpha(\sigma_k) = 1 - \exp(-\sigma_k)$. The density and color are queried as: $(\sigma_k, c_k) = F_\theta(\mathcal{T}_i(x_i^k))$, we omit i when there is no ambiguity. To get the final motion estimation result \hat{p}_j , \hat{x}_j is directly projected back to the 2D image plane. In the same way, the image space color \hat{C}_i for p_i can be obtained by compositing c_k . The representation uses reliable correspondences as well as the original RGB frames as labels for optimization. Given any novel coordinate in the frame, the model can render consistent full-length motion in the video.

3.2 Decoupled representation

As analyzed above, dynamic objects and static scenes are different in terms of movement and appearance. Therefore, we explicitly decouple the video into two parts, and optimize representations for static scenes and dynamic objects separately. The final representation is obtained by volumetric fusion. The overview of our method is shown in Figure 1.

3D volume of static scenes. In contrast to a single global 3D volume, we maintain canonical volume $G_{\text{st}}, G_{\text{dy}}$ for static scenes and dynamic objects respectively. For static scenes, G_{st} refers to the design of [36], i.e., accessing the color c^{st} and the density σ^{st} . At the same time, G_{st} also stores the confidence β^{st}

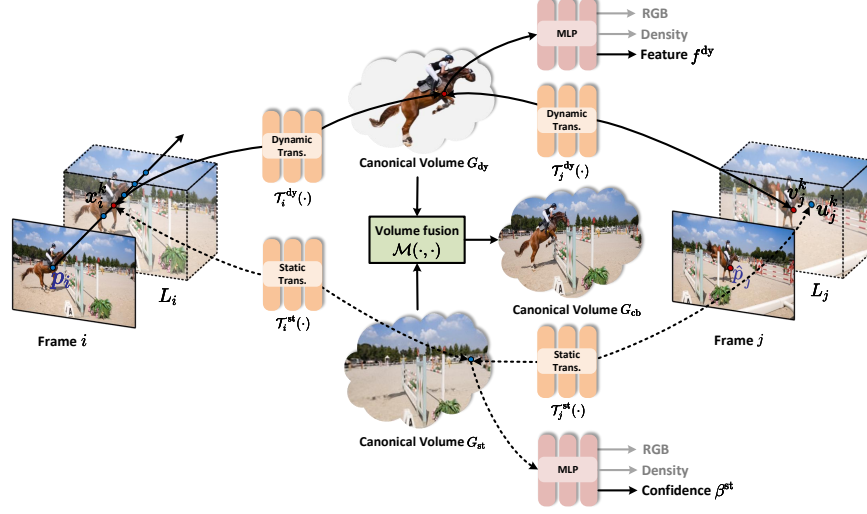


Fig. 1: Overview of the proposed decoupled representation. We explicitly define two separate 3D canonical volumes G_{dy} and G_{st} to respectively characterize dynamic objects and static scenes in videos. In each representation, in addition to color and density, the G_{dy} encodes the feature f^{dy} to better represent dynamic objects, and feature rendering loss is further proposed to rectify the non-rigid transformation. The G_{st} stores the β^{st} determining the confidence of being a static point. In each canonical volume, we carefully design transformation functions \mathcal{T}^{dy} (solid line) and \mathcal{T}^{st} (dash line) to map each local 3D point x_i^k along the ray of the point p_i to u_j^k, v_j^k in another local 3D volume L_j . In order to render the 2D correspondence \hat{p}_j for p_i , we get the canonical volume G_{cb} of final representation by volume fusion with β^{st} . Set of 3D points $\{u_j^k\}_{k=1}^K, \{v_j^k\}_{k=1}^K$ mapped from $\{x_i^k\}_{k=1}^K$ are aggregated by alpha compositing in the G_{cb} (see Eq.(5)), and are projected to the image plane.

of whether each 3D point in the canonical volume is a static point or not through network F_θ^{st} . The pixel color \hat{C}_i^{st} , motion mask \hat{B}_i^{st} , and 2D correspondence \hat{p}_j^{st} for p_i can be obtained by volume rendering similar to Eq.(2).

3D volume of dynamic objects. For dynamic objects, there are often appearance changes and deformations in video. Simple color features can no longer guarantee temporal consistency, and are also very susceptible to similar distractors. Image features can represent objects more discriminatively and consistently [17, 37], which would also be beneficial in obtaining continuous and consistent motion trajectories. Therefore, we additionally encode the 3D feature f^{dy} in G_{dy} via F_θ^{dy} . Through volume rendering, apart from the pixel color \hat{C}_i^{dy} and the 2D correspondence \hat{p}_j^{dy} for p_i , the 2D feature \hat{F}_i^{dy} can also be obtained:

$$\hat{F}_i^{dy}(p_i) = \sum_{k=1}^K T_k^{dy} \alpha(\sigma_k^{dy}) f_k^{dy}, \text{ where } T_k^{dy} = \prod_{l=1}^{k-1} (1 - \alpha(\sigma_l^{dy})). \quad (3)$$

Transformations. We also carefully design the static transformation \mathcal{T}^{st} and the dynamic transformation \mathcal{T}^{dy} , respectively. For the static scenes, the movement caused by the camera’s rigid-motion is mainly considered, so we model the static mapping by optimizing the parameters of the 3D affine transformation between frames, i.e., $\mathcal{T}^{\text{st}}(\cdot) = A_\theta(\cdot)$. The motion trajectories of the object are the result of the superposition of the camera and the object’s own motion, which often corresponds to a more complex non-linear transformation. Therefore, we refer to the method of [36], and additionally utilize the Real-NVP [8] to model the dynamic mapping, i.e., $\mathcal{T}^{\text{dy}}(\cdot) = M_\theta(A_\theta(\cdot); \phi_i)$, where ϕ_i stands for the individually optimized latent code for each frame. Due to the ambiguity of motion decomposition, we find that for static transformation, it is also beneficial to appropriately add some non-linear transformation layers, which will be discussed in the subsequent experimental analysis.

Volume fusion. After obtaining the static and dynamic representations, we obtain the final 3D volume G_{cb} by volume fusion, i.e., $\mathcal{M}(G_{\text{st}}, G_{\text{dy}}) = G_{\text{cb}}$, in order to get the final representation of the whole video scene, as well as the pixels’ motion trajectories. We leverage the orthogonal approximation technique described in previous work [25] to approximate the process:

$$\sigma_k^{\text{cb}} x_j^k = \beta_k^{\text{st}} u_j^k \sigma_k^{\text{st}} + (1 - \beta_k^{\text{st}}) v_j^k \sigma_k^{\text{dy}}, \quad (4)$$

where u_j^k/v_j^k represents the local 3D points at L_j obtained by static/dynamic transformation for x_i^k . The motion rendering in Eq.(2) can be rewritten as:

$$\begin{aligned} \hat{\mathbf{x}}_j^{\text{cb}} &= \sum_{k=1}^K T_k^{\text{cb}} \left(\beta_k^{\text{st}} \alpha(\sigma_k^{\text{st}}) u_j^k + (1 - \beta_k^{\text{st}}) \alpha(\sigma_k^{\text{dy}}) v_j^k \right) \\ \text{where } T_k^{\text{cb}} &= \exp \left(- \sum_{l=1}^{k-1} \beta_l^{\text{st}} \sigma_l^{\text{st}} + (1 - \beta_l^{\text{st}}) \sigma_l^{\text{dy}} \right) \end{aligned} \quad (5)$$

We obtain final 2D correspondence $\hat{\mathbf{x}}_j^{\text{cb}}$ by projecting \hat{p}_j^{cb} to the image plane. Similarly, we can get the rendered color \hat{C}_i^{cb} by replacing 3D coordinate with color in Eq.(5).

3.3 Optimization

In order to obtain a decoupled 3D representation of the video, we utilize a variety of loss functions, including feature rendering loss, color rendering loss, motion rendering loss, etc. This section will describe them in detail.

Data preprocessing. We refer to the previous method [35, 36] to get the optical flow between every two frames of the video by the advanced optical flow estimation method RAFT [33] and filter out the optical flow results with high confidence as the training pseudo-labels. In addition to this, to realize feature

rendering loss, we also extract the feature F by video correspondence learning methods [6, 37, 40]. In order to achieve better decoupling, we also refer to the previous methods [13, 24] to get the mask M of the moving objects by semantic segmentation [7, 31] or motion segmentation [20, 38, 42], which serves as a guidance for the network in the initial period of training.

Optimization for dynamic objects. For dynamic objects, the following loss functions are used for optimization. After obtaining the collected reliable optical flow $\mathbf{f}_{i \rightarrow j} = p_j - p_i$, predict optical flow $\hat{\mathbf{f}}_{i \rightarrow j}^{\text{dy}} = \hat{p}_j^{\text{dy}} - p_i$ and the motion segmentation mask M_i , the L_1 distance is used as the motion rendering loss, and the motion segmentation mask forces the transformation network to focus only on modeling the motion of the dynamic objects during the initial phase of optimization, and the Ω_f indicates the set of filtered flows:

$$\mathcal{L}_{\text{flo}}^{\text{dy}} = \sum_{\mathbf{f}_{i \rightarrow j} \in \Omega_f} M_i(p_i) \left\| \hat{\mathbf{f}}_{i \rightarrow j}^{\text{dy}} - \mathbf{f}_{i \rightarrow j} \right\|_1 \quad (6)$$

In addition to this, we also utilize the color rendering loss. The loss decreases only if corresponding 2D points p_i and p_j map to 3D points in the canonical space with the same color. We believe through this way, the representation implicitly captures the correspondences by matching the appearance, which shares similar spirits to previous matching-based motion estimation methods [32, 34]. The loss is defined as:

$$\mathcal{L}_{\text{pho}}^{\text{dy}} = \sum_{p_i \in \Omega_p} M_i(p_i) \left\| \hat{C}_i^{\text{dy}}(p_i) - C_i(p_i) \right\|_2^2 \quad (7)$$

The Ω_p indicates the set of filtered points. However, as introduced above, the photometric error reflects only the similarity in color space and is very susceptible to distractors. For example, in Figure 2(a), the pixels $\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3$ are very similar in color space, leading to a reduction of the photometric error even if \mathbf{s}_1 in the right frame is incorrectly mapped to \mathbf{s}_3 . Whereas in (b), \mathbf{s}_3 is significantly different from the rest of the points in the feature space, which indicates this problem can be mitigated by rendering discriminative features. Moreover, inter-frame appearance changes and deformations make pixels inconsistent between frames, especially on dynamic objects. Therefore, this paper proposes a feature-based rendering loss:

$$\mathcal{L}_{\text{feat}} = \sum_{p_i \in \Omega_p} M_i(p_i) \left\| \hat{F}_i^{\text{dy}}(p_i) - F_i(p_i) \right\|_1 \quad (8)$$

Using the pre-trained temporally-consistent and spatially-discriminative features [6] as labels, we rectify the dynamic transformation and force corresponding pixels to map to the same 3D point in canonical space. The loss function used for dynamic objects is defined as $\mathcal{L}^{\text{dy}} = \mathcal{L}_{\text{flow}}^{\text{dy}} + \lambda_1^{\text{dy}} \mathcal{L}_{\text{pho}}^{\text{dy}} + \lambda_2^{\text{dy}} \mathcal{L}_{\text{feat}}^{\text{dy}}$.

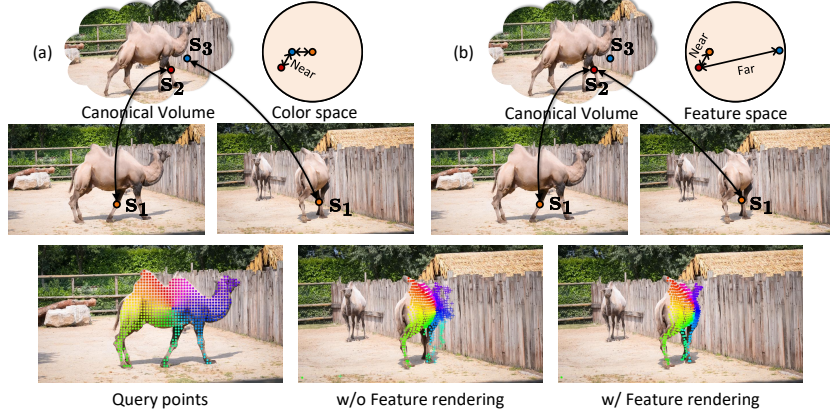


Fig. 2: The illustration of feature rendering loss. We also show the qualitative results in the last row where the query points are marked with different colors. Please refer to the corresponding video in the supplemental materials. **(Zoom in for best view)**

Optimization for static scenes. For static scenes, we also adopt motion and color rendering loss for optimization:

$$\mathcal{L}_{\text{flo}}^{\text{st}} = \sum_{\mathbf{f}_{i \rightarrow j} \in \Omega_f} (1 - M_i(p_i)) \left\| \hat{\mathbf{f}}_{i \rightarrow j}^{\text{st}} - \mathbf{f}_{i \rightarrow j} \right\|_1 \quad (9)$$

$$\mathcal{L}_{\text{pho}}^{\text{st}} = \sum_{p_i \in \Omega_p} (1 - M_i(p_i)) \left\| \hat{C}_i^{\text{st}}(p_i) - C_i(p_i) \right\|_2^2 \quad (10)$$

Besides, the motion segmentation masks are utilized to supervise the rendered motion mask $\hat{B}_i^{\text{st}}(p_i)$, giving guidance to static network in the early stages:

$$\mathcal{L}_{\text{seg}} = \sum_{p_i \in \Omega_p} -(1 - M_i(p_i)) \log(\hat{B}_i^{\text{st}}(p_i)) - M_i(p_i) \log(1 - \hat{B}_i^{\text{st}}(p_i)), \quad (11)$$

the optimized motion confidence will be further used in volume fusion, and the loss function used for static scenes is defined as $\mathcal{L}^{\text{st}} = \mathcal{L}_{\text{flow}}^{\text{st}} + \lambda_1^{\text{st}} \mathcal{L}_{\text{pho}}^{\text{st}} + \lambda_2^{\text{st}} \mathcal{L}_{\text{seg}}$.

Optimization for final representation. Based on obtaining the respective representation and transformation of the static scenes and dynamic objects, we attempt to obtain the final representation by volume fusion. We use the motion and color rendering loss for optimization, denoted as $\mathcal{L}^{\text{cb}} = \mathcal{L}_{\text{flo}}^{\text{cb}} + \lambda_1^{\text{cb}} \mathcal{L}_{\text{pho}}^{\text{cb}}$:

$$\mathcal{L}_{\text{flo}}^{\text{cb}} = \sum_{\mathbf{f}_{i \rightarrow j} \in \Omega_f} \left\| \hat{\mathbf{f}}_{i \rightarrow j}^{\text{cb}} - \mathbf{f}_{i \rightarrow j} \right\|_1 \quad (12)$$

$$\mathcal{L}_{\text{pho}}^{\text{cb}} = \sum_{p_i \in \Omega_p} \left\| \hat{C}_i^{\text{cb}}(p_i) - C_i(p_i) \right\|_2^2 \quad (13)$$

In addition to the above losses, we also adopt regularization schemes used in prior work, e.g., the scene flow smoothness loss [23], the distortion loss [2], which are labeled as \mathcal{L}^{reg} . The final overall training loss is defined as follows:

$$\mathcal{L} = \mathcal{L}^{\text{dy}} + \mathcal{L}^{\text{st}} + \mathcal{L}^{\text{cb}} + \mathcal{L}^{\text{reg}} \quad (14)$$

4 Experiment

To validate the proposed method, we evaluate our method on the TAP-Vid benchmark [9], which is designed to evaluate the performance of long-range motion estimation (point tracking) in the real world. This section first describes the experimental setup including dataset, network, training, and evaluation details. After that, detailed ablation studies and performance comparisons with the baseline methods will be presented to prove the effectiveness of the proposed method for motion estimation.

4.1 Implementation details

Dataset. Since the proposed method is a test-time optimization method, considering the tractable, a more representative TAP-Vid-DAVIS [9] dataset is selected for training and testing. This dataset, contains 30 videos in the real world, with videos ranging from 34-104 frames and an average of 21.7 points annotations per video. The videos in TAP-Vid [9] are downsampled to 256×256 , which ensures that no more information than expected is used.

Network. The parameters corresponding to the affine transformation A_θ are predicted by the 2-layer MLP. Meanwhile, as discussed in Figure 4, we also incorporate non-linear invertible transformation layers [8] in the static transformation network and set the number of layers n_{st} to 3. The M_θ in the dynamic transformation network consists of 6 invertible layers. The F_θ^{dy} and F_θ^{st} use the GaborNet [12] with 3 layers and 512 channels per layer to encode the color, motion confidence, and density. Considering the complexity of the features, we leverage another GaborNet alone to do the feature rendering. Following OmniMotion [36], a 2-layer fully-connected layer with 256 channels is included to compute the implicit representation ϕ_i for each video frame, and the normalized time t_i is used as the input to obtain the time embedding.

Training details. DecoMotion uses the Adam as an optimizer to train each video sequence for 100k iterations. Warm-up strategy is used to slowly increase the learning rate, and we decay the learning rate every 20k steps thereafter. In each training batch, we extract 256 pairs of optical flows from 8 pairs of images. For each pixel corresponding to a ray, we uniformly sample $K = 32$ local 3D points in the z axis. In feature rendering, we use DINO [6] as an image feature extractor in the data preprocessing stage to extract 768-d features for each point.

Table 1: Ablation study for decomposition.

\mathcal{L}^{st}	\mathcal{L}^{dy}	\mathcal{L}^{cb}	TAP-Vid-DAVIS		
			AJ \uparrow	$< \delta_{avg}^x \uparrow$	OA \uparrow
✓	✓		52.6	69.9	85.3
			27.6	31.0	77.6
			54.3	73.3	85.2
		✓	57.9	76.4	86.0
✓	✓	✓	59.9	79.0	86.1

The rest of the settings including reliable sample filtering, hard sample mining strategy, and far and near depth for volume rendering refer to the work in [36]. Further implementation details are presented in the supplemental materials.

Evaluation metrics. Following the TAP-Vid benchmark [9] and OmniMotion [36], we report the tracking position accuracy ($< \delta_{avg}^x$), Occlusion Accuracy (OA), Average Jaccard (AJ), and Temporal Coherence (TC) for predict motion trajectories. Please refer to the benchmark for more information.

4.2 Ablation study

We perform detailed ablation studies on point tracking. Considering DecoMotion is a test-time optimization method, to ensure the executability of the experiments, a subset of TAP-Vid-DAVIS [9] (480p) is randomly sampled.

The effectiveness of decomposition. We first study how each design in our decoupled representation affects the performance of point tracking. The original OmniMotion is set as the baseline. The results are shown in Table 1. The \mathcal{L}^{st} , \mathcal{L}^{dy} and \mathcal{L}^{cb} represents the loss functions used in each representation optimization. Solely optimizing representation for static scenes substantially harms the performance, the learned static transformation can not deal with the complex motion patterns of dynamic objects. Whereas the proprietary transformation for dynamic objects improves the tracking performance from 69.9% to 73.3%. With the help of the motion segmentation mask, the dynamic transformation can focus on learning the difficult parts, achieving the globally optimal capability of estimating the object’s motion. However, the dynamic transformation still fails to capture the camera motion. By combining the static part with \mathcal{L}^{cb} in volume fusion, the tracking accuracy is further improved to 79.0%. The results demonstrate the effectiveness of the divide-and-conquer strategy in motion estimation.

The motion segmentation models [42] are complex and sometimes could be brittle. A dynamic/static point could be incorrectly assigned to a static/dynamic point, which misleads the optimization. Without the guidance of motion segmentation masks, we find optimizing the final representation solely with \mathcal{L}^{cb}



Fig. 3: The rendering results for static scenes and dynamic objects. Only optimized with \mathcal{L}^{cb} , we still observe the decomposition in terms of motion and appearance.

still shows competitive results. In Figure 3, we show rendering results of static appearance \hat{C}_i^{st} , dynamic appearance \hat{C}_i^{dy} , and motion segmentation mask \hat{B}_i^{st} , we can still observe the decomposition in terms of appearance and motion. Moreover, thanks to the insightful feedback of reviewers, we are seeking more elegant ways to provide reasonable results without a segmenter, e.g. first optimizing the globally-rigid component alone, and then adding the non-rigid component to explain the remainder. We regard it as our future work.

Feature rendering. We analyze the effect of the proposed feature rendering. The results are shown in Table 2. In (a), we regard the DecoMotion without using any appearance rendering loss as the baseline. With the implicit appearance matching in canonical space, both color and feature rendering enhance the performance of motion estimation. Moreover, the proposed feature rendering further boots up the performance from 76.1% to 79.0%. As shown in the last row of Figure 2, the given query points are misled by the distractors in the background. By using feature rendering loss, the query points are accurately matched to the object in another frame, even facing large deformations. Nevertheless, the feature rendering on static scenes does not help much. We believe this is because of the relatively poor representation ability of the pre-trained features [6] and the slight inter-frame changes in the background. Besides, in (b), we use different pre-trained features as supervision, compared with FGVC [21] that more specializes in extracting local features, the VFS [40] and DINO [6], which are better at representing objects, show better performance.

Table 2: Ablation study for feature rendering.

(a) Ablation for feature rendering in different scenes.

\mathcal{L}_{pho}		\mathcal{L}_{feat}		TAP-Vid-DAVIS		
Dynamic	Static	Dynamic	Static	AJ \uparrow	$< \delta_{avg}^x \uparrow$	OA \uparrow
				51.2	69.7	85.6
✓				55.6	74.5	85.8
✓	✓			57.3	76.1	85.5
✓	✓	✓		59.9	79.0	86.1
✓	✓	✓	✓	60.2	79.2	85.9

(b) Ablation for different features.

Method	$< \delta_{avg}^x \uparrow$
w/o \mathcal{L}_{feat}	76.1
\mathcal{L}_{feat} w/ FGVC [21]	76.7
\mathcal{L}_{feat} w/ VFS [40]	77.3
\mathcal{L}_{feat} w/ DINO [6]	79.0

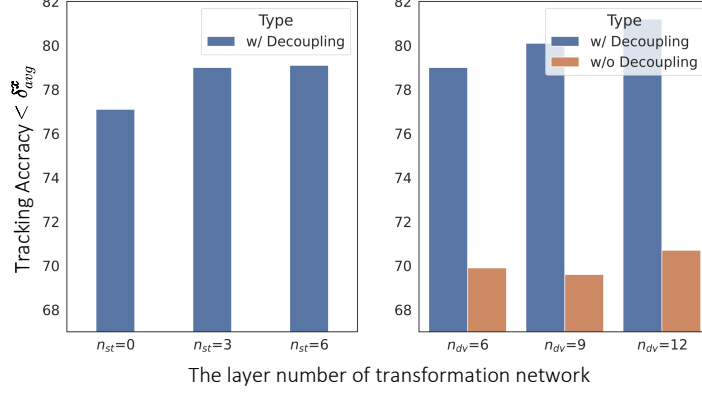


Fig. 4: The ablation study for the transformations. The n_{st}, n_{dy} represent the number of non-linear layers [8] used in static and dynamic transformations.

The design of transformation. In Figure 4, the n_{st}, n_{dy} represent the number of non-linear invertible layers [8] used in static and dynamic transformations. For static transformation, purely utilizing affine transformation, i.e., $n_{st} = 0$, gives the worst performance. We find the motion confidence prediction is not always perfect. If a dynamic point is incorrectly predicted as a static point, the affine transformation alone can not deal with it well. Adding some non-linear layers in the static transformation improves the performance. However, including more layers does not bring benefits. Besides, increasing the n_{dy} in dynamic transformation shows a more significant improvement compared with the method without decoupling.

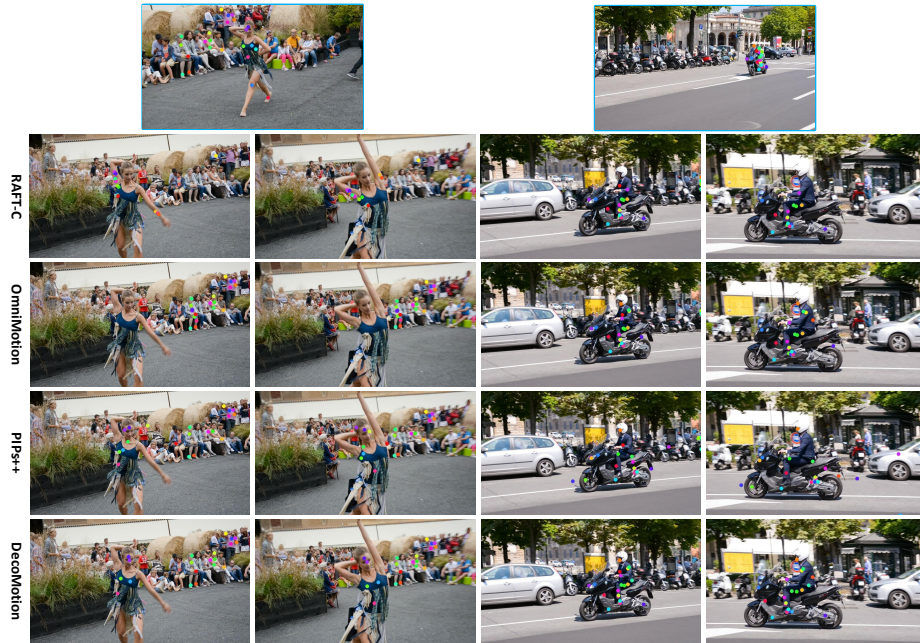
4.3 Comparisons on point tracking

We compare our method with the state-of-the-art methods of point tracking. The quantitative results are shown in Table 3. Apart from optimization-based methods Deformable-Sprites and OmniMotion, we add MFT that utilizes input pairwise correspondences from RAFT as our competitors. Compared with the most direct competitor OmniMotion, DecoMotion leads the tracking position accuracy $< \delta_{avg}^x$ by 6.9%/7.2% on DAVIS-Strided and DAVIS-First. Also, DecoMotion is superior to Deformable-Sprites and MFT, surpassing MFT by 3.6%/3.1%. We also include some dedicated point-tracking methods that track through occlusions by estimating multi-frame point trajectories for comparisons. Note DecoMotion is only supervised by RAFT which has limited capability of providing accurate point correspondences, DecoMotion still shows on-par performance, leading the tracking position accuracy by 0.7% on DAVIS-Strided.

We select several representative videos for inference. DecoMotion produces more accurate and stable pixel correspondences over frames. For example, in the

Table 3: Quantitative results for point tracking on TAP-Vid [9]. The “(RAFT)” indicates using pairwise correspondences from RAFT [33] as input or supervision.

Method	DAVIS Strided				DAVIS First			
	AJ \uparrow	$< \delta_{avg}^x \uparrow$	OA \uparrow	TC \downarrow	AJ \uparrow	$< \delta_{avg}^x \uparrow$	OA \uparrow	TC \downarrow
Flow-Walk-C [4]	35.2	51.4	80.6	0.90	-	-	-	-
Flow-Walk-D [4]	24.4	40.9	76.5	10.41	-	-	-	-
RAFT-C [33]	30.7	46.6	80.2	0.93	27.1	42.1	77.6	0.95
RAFT-D [33]	34.1	48.9	76.1	9.83	31.4	44.5	74.1	10.3
TAP-Net [9]	38.4	53.4	81.4	10.82	33.0	48.6	78.8	-
PIPs [14]	39.9	56.0	81.3	1.78	36.4	53.8	76.1	-
Context-TAP [3]	48.9	64.0	-	-	42.7	60.3	-	-
FGVC [21]	-	66.4	-	-	-	62.8	-	-
PIPs++ [44]	-	73.7	-	-	-	69.1	-	-
TAPIR [10]	61.3	73.6	88.8	-	56.2	70.0	86.5	-
Deformable-Sprites [41]	20.6	32.9	69.7	2.07	-	-	-	-
OmniMotion [36] (RAFT)	51.7	67.5	85.3	0.74	44.9	62.7	80.7	0.73
MFT [27] (RAFT)	56.1	70.8	86.9	-	47.3	66.8	77.8	-
DecoMotion (RAFT)	60.2	74.4	87.2	0.69	53.0	69.9	84.2	0.69

**Fig. 5:** Qualitative results for point tracking. Given the query points marked with different colors in the first frame (blue border), we visualize the visible correspondences in randomly sampled frames. Please refer to more videos in the supplemental materials. **(Zoom in for best view)**

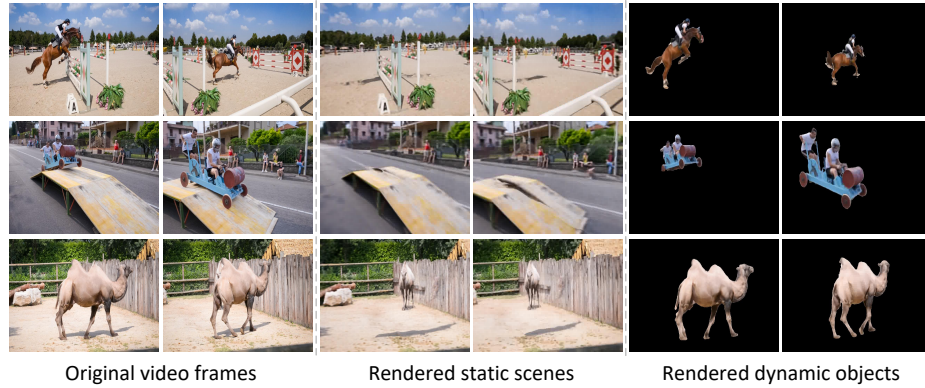


Fig. 6: The rendering results for decoupled static scenes and dynamic objects. Please refer to the corresponding videos in the supplemental materials.

left case of Figure 5, the dancing girl presents a very complex movement with large deformations, the baseline methods either lose track or get imprecise results, while DecoMotion can consistently track points accurately, which validates the superior ability of DecoMotion to handle the motion of dynamic objects.

4.4 Visualize the appearance decomposition

We present the visualization of the decoupled static scenes and dynamic objects in Figure 6. Using the individual 3D canonical volume and transformation, we generate each frame in the given video by volume rendering. We can clearly observe the static part focuses on learning the appearance of static scenes, while the dynamic part pays more attention to the dynamic objects. With such a decoupled representation, our method can “remove” the dynamic objects while maintaining reasonable structure.

5 Conclusion

In this work, we introduce an innovative test-time optimization approach for establishing accurate pixel correspondences. We separately establish 3D representation for static scenes and dynamic objects. For dynamic objects, we address their complex pattern by incorporating more non-linear layers with a feature rendering loss. For static scenes, we employ a simpler network to handle rigid motion, while also modeling the confidence of whether 3D points are static or not. Then we fuse these two canonical volumes to better represent motion and appearance. Quantitative results on point tracking and qualitative results of appearance decomposition validate the proposed method.

Acknowledgements. We thank anonymous reviewers for their valuable feedback. This work was supported by the Natural Science Foundation of China under Grant 61931014, and by the Fundamental Research Funds for the Central Universities under Grant WK3490000006.

References

1. Baker, S., Matthews, I.: Lucas-kanade 20 years on: A unifying framework. *IJCV* **56**, 221–255 (2004)
2. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: *CVPR*. pp. 5470–5479 (2022)
3. Bian, W., Huang, Z., Shi, X., Dong, Y., Li, Y., Li, H.: Context-tap: Tracking any point demands spatial context features. In: *NeurIPS* (2023)
4. Bian, Z., Jabri, A., Efros, A.A., Owens, A.: Learning pixel trajectories with multi-scale contrastive random walks. In: *CVPR*. pp. 6508–6519 (2022)
5. Bouguet, J.Y., et al.: Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. Intel corporation **5**(1-10), 4 (2001)
6. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: *ICCV*. pp. 9650–9660 (2021)
7. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. on PAMI* **40**(4), 834–848 (2017)
8. Dinh, L., Sohl-Dickstein, J., Bengio, S.: Density estimation using real nvp. *arXiv preprint arXiv:1605.08803* (2016)
9. Doersch, C., Gupta, A., Markeeva, L., Recasens, A., Smaira, L., Aytar, Y., Carreira, J., Zisserman, A., Yang, Y.: Tap-vid: A benchmark for tracking any point in a video. In: *NeurIPS*. vol. 35, pp. 13610–13626 (2022)
10. Doersch, C., Yang, Y., Vecerik, M., Gokay, D., Gupta, A., Aytar, Y., Carreira, J., Zisserman, A.: Tapir: Tracking any point with per-frame initialization and temporal refinement. In: *ICCV* (2023)
11. Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., Brox, T.: FlowNet: Learning optical flow with convolutional networks. In: *ICCV*. pp. 2758–2766 (2015)
12. Fathony, R., Sahu, A.K., Willmott, D., Kolter, J.Z.: Multiplicative filter networks. In: *ICLR* (2020)
13. Gao, C., Saraf, A., Kopf, J., Huang, J.B.: Dynamic view synthesis from dynamic monocular video. In: *ICCV*. pp. 5712–5721 (2021)
14. Harley, A.W., Fang, Z., Fragkiadaki, K.: Particle video revisited: Tracking through occlusions using point trajectories. In: *ECCV*. pp. 59–75 (2022)
15. Horn, B.K., Schunck, B.G.: Determining optical flow. *Artificial intelligence* **17**(1-3), 185–203 (1981)
16. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: FlowNet 2.0: Evolution of optical flow estimation with deep networks. In: *CVPR*. pp. 2462–2470 (2017)
17. Jabri, A., Owens, A., Efros, A.: Space-time correspondence as a contrastive random walk. In: *NeurIPS*. vol. 33, pp. 19545–19560 (2020)
18. Lai, Z., Lu, E., Xie, W.: Mast: A memory-augmented self-supervised tracker. In: *CVPR*. pp. 6479–6488 (2020)
19. Li, R., Liu, D.: Spatial-then-temporal self-supervised learning for video correspondence. In: *CVPR*. pp. 2279–2288 (2023)
20. Li, R., Zhang, Y., Qiu, Z., Yao, T., Liu, D., Mei, T.: Motion-focused contrastive learning of video representations. In: *ICCV*. pp. 2105–2114 (2021)

21. Li, R., Zhou, S., Liu, D.: Learning fine-grained features for pixel-wise video correspondences. In: ICCV. pp. 9632–9641 (2023)
22. Li, X., Liu, S., De Mello, S., Wang, X., Kautz, J., Yang, M.H.: Joint-task self-supervised learning for temporal correspondence. In: NeurIPS. vol. 32 (2019)
23. Li, Z., Niklaus, S., Snavely, N., Wang, O.: Neural scene flow fields for space-time view synthesis of dynamic scenes. In: CVPR. pp. 6498–6508 (2021)
24. Li, Z., Wang, Q., Cole, F., Tucker, R., Snavely, N.: Dynibar: Neural dynamic image-based rendering. In: CVPR. pp. 4273–4284 (2023)
25. Martin-Brualla, R., Radwan, N., Sajjadi, M.S., Barron, J.T., Dosovitskiy, A., Duckworth, D.: Nerf in the wild: Neural radiance fields for unconstrained photo collections. In: CVPR. pp. 7210–7219 (2021)
26. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM* **65**(1), 99–106 (2021)
27. Neoral, M., Šerých, J., Matas, J.: Mft: Long-term tracking of every pixel. In: WACV. pp. 6837–6847 (2024)
28. Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Nerfies: Deformable neural radiance fields. In: ICCV. pp. 5865–5874 (2021)
29. Sand, P., Teller, S.: Particle video: Long-range motion estimation using point trajectories. *IJCV* **80**, 72–91 (2008)
30. Shi, X., Huang, Z., Bian, W., Li, D., Zhang, M., Cheung, K.C., See, S., Qin, H., Dai, J., Li, H.: Videoflow: Exploiting temporal cues for multi-frame optical flow estimation. In: ICCV (2023)
31. Strudel, R., Garcia, R., Laptev, I., Schmid, C.: Segmenter: Transformer for semantic segmentation. In: ICCV. pp. 7262–7272 (2021)
32. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In: CVPR. pp. 8934–8943 (2018)
33. Teed, Z., Deng, J.: Raft: Recurrent all-pairs field transforms for optical flow. In: ECCV. pp. 402–419 (2020)
34. Vondrick, C., Shrivastava, A., Fathi, A., Guadarrama, S., Murphy, K.: Tracking emerges by colorizing videos. In: ECCV. pp. 391–408 (2018)
35. Wang, C., MacDonald, L.E., Jeni, L.A., Lucey, S.: Flow supervision for deformable nerf. In: CVPR. pp. 21128–21137 (2023)
36. Wang, Q., Chang, Y.Y., Cai, R., Li, Z., Hariharan, B., Holynski, A., Snavely, N.: Tracking everything everywhere all at once. In: ICCV (2023)
37. Wang, X., Jabri, A., Efros, A.A.: Learning correspondence from the cycle-consistency of time. In: CVPR. pp. 2566–2576 (2019)
38. Xie, J., Xie, W., Zisserman, A.: Segmenting moving objects via an object-centric layered representation. In: NeurIPS. vol. 35, pp. 28023–28036 (2022)
39. Xu, H., Zhang, J., Cai, J., Rezatofighi, H., Tao, D.: Gmflow: Learning optical flow via global matching. In: CVPR. pp. 8121–8130 (2022)
40. Xu, J., Wang, X.: Rethinking self-supervised correspondence learning: A video frame-level similarity perspective. In: ICCV. pp. 10075–10085 (2021)
41. Ye, V., Li, Z., Tucker, R., Kanazawa, A., Snavely, N.: Deformable sprites for unsupervised video decomposition. In: CVPR. pp. 2657–2666 (2022)
42. Yuan, Y., Wang, Y., Wang, L., Zhao, X., Lu, H., Wang, Y., Su, W., Zhang, L.: Isomer: Isomeric transformer for zero-shot video object segmentation. In: ICCV. pp. 966–976 (2023)

43. Zhao, W., Liu, S., Guo, H., Wang, W., Liu, Y.J.: Particlesfm: Exploiting dense point trajectories for localizing moving cameras in the wild. In: ECCV. pp. 523–542 (2022)
44. Zheng, Y., Harley, A.W., Shen, B., Wetzstein, G., Guibas, L.J.: Pointodyssey: A large-scale synthetic dataset for long-term point tracking. In: ICCV. pp. 19855–19865 (2023)