Learning Pixel Trajectories with Multiscale Contrastive Random Walks

Zhangxing Bian\textsuperscript{1,3} Allan Jabri\textsuperscript{2} Alexei A. Efros\textsuperscript{2} Andrew Owens\textsuperscript{1}
University of Michigan\textsuperscript{1} UC Berkeley\textsuperscript{2} Johns Hopkins University\textsuperscript{3}

Abstract

A range of video modeling tasks, from optical flow to multiple object tracking, share the same fundamental challenge: establishing space-time correspondence. Yet, approaches that dominate each space differ. We take a step towards bridging this gap by extending the recent contrastive random walk formulation to much denser, pixel-level space-time graphs. The main contribution is introducing hierarchy into the search problem by computing the transition matrix between two frames in a coarse-to-fine manner, forming a multiscale contrastive random walk when extended in time. This establishes a unified technique for self-supervised learning of optical flow, keypoint tracking, and video object segmentation. Experiments demonstrate that, for each of these tasks, the unified model achieves performance competitive with strong self-supervised approaches specific to that task.\textsuperscript{1}

1. Introduction

Temporal correspondence underlies a range of video understanding tasks, from optical flow to object tracking. At the core, the challenge is to estimate the motion of some entity as it persists in the world, by searching in space and time. For historical reasons, the practicalities differ substantially across tasks: optical flow aims for dense correspondences but only between neighboring pairs of frames, whereas tracking cares about longer-range correspondences but is spatially sparse. We argue that the time might be right to try and re-unify these different takes on temporal correspondence.

An emerging line of work in self-supervised learning has shown that generic representations pretrained on unlabeled images and video can lead to strong performance across a range of tracking tasks [20, 26, 35, 75, 77]. The key idea is that if tracking can be formulated as label propagation [86] on a space-time graph, all that is needed is a good measure of similarity between nodes. Indeed, the recent contrastive random walk (CRW) formulation [26] shows how such a similarity measure can be learned for temporal correspondence problems, suggesting a path towards a unified solution. However, scaling this perspective to pixel-level space-time graphs holds challenges. Since computing similarity between frames is quadratic in the number of nodes, estimating dense motion is prohibitively expensive. Moreover, there is no way of explicitly estimating the motion in ambiguous cases, like occlusion. In parallel, the unsupervised optical flow community has adopted highly effective methods for dense matching [83], which use multiscale representations [9, 45, 61, 84] to reduce the large search space, and smoothness priors to deal with ambiguity and occlusion. But, in contrast to the self-supervised tracking methods, they rely on hand-crafted distance functions, such as the Census Transform [32, 47]. Furthermore, because they focus on producing point estimates of motion, they may be less robust under long-term dynamics.

In this work, we take a step toward bridging the gap between tracking and optical flow by extending the contrastive random walk formulation [26] to much denser, pixel-level space-time graphs. The main contribution is introducing hierarchy into the search problem, i.e. the multiscale contrastive random walk. By integrating local attention in a coarse-to-fine manner, the model can efficiently consider a distribution over pixel-level trajectories. Through experiments across optical flow and video label propagation benchmarks, we show:

- This provides a unified technique for self-supervised optical flow, pose tracking, and video object segmentation.
- For optical flow, the model is competitive with many recent unsupervised optical flow methods, despite using a novel loss function (without hand-crafted features).
- For tracking, the model outperforms existing self-supervised approaches on pose tracking, and is competitive on video object segmentation.
- Cycle-consistency provides a complementary learning signal to photo-consistency.
- Multi-frame training improves two-frame performance.

2. Related work

Space-time representation learning. Recent work has proposed methods for tracking objects in video through
Figure 1. **Multiscale contrastive random walks.** We learn representations for dense, fine-grained matching using *multiscale contrastive random walks*. At each scale, we create a space-time graph in which each pixel is a node, and where nodes close in space-time are connected. Transition probabilities are based on similarity in a learned representation. We train the network to maximize the probability of obtaining a cycle-consistent random walk. Here, we illustrate a pixel’s walk over 3 spatial scales and 3 frames. The walker is initialized using its position at the previous, coarser scale.

self-supervised representation learning. Vondrick et al. [71] posed tracking as a colorization problem, by training a network to match pixels in grayscale images that have the same held-out colors. The assumption that matching pixels have the same color may break down over long time horizons, and the method is limited to grayscale inputs. This approach was extended to obtain higher-accuracy matches with two-stage matching [35,38]. In contrast to our approach, the predictions are coarse, patch-level associations. Another line of work learns representations by maximizing cycle consistency. These methods track patches forward, then backward, in time and test whether they end up where they began. Wang et al. [74] proposed a method based on hard attention and spatial transformers [27]. Jabri et al. [26] posed cycle-consistency as a random walk problem, allowing the model to obtain dense supervision from multi-frame video. Tang et al. [67] proposed an extension that allowed for fully convolutional training. These approaches are trained on sparse patches and learn coarse-grained correspondences. In contrast, we learn pixel-to-pixel correspondences. Other work has encouraged patches in the same position in neighboring frames to be close in an embedding space [17,79].


**Unsupervised optical flow.** Early work [49] used Boltzmann machines to learn transformations between images. More recently, Yu et al. [83] train a neural network to minimize a loss very similar to that of optimization-based approaches. Later work extended this approach by adding edge-aware smoothing [76], hand-crafted features [48], occlusion handling [28,76], learned upsampling [46], and depth and camera pose [82,85]. Another approach learns flow by matching augmented image pairs [41,42,44]. Recently, Jonschkowski et al. [32] surveyed previous literature and conducted an exhaustive search to find the best combination of methods and hyperparameters. In contrast to these works, our goal is to learn self-supervised representations for matching, in lieu of hand-crafted features. Moreover, we aim to produce a distribution over motion trajectories suitable for label transfer, rather than motion estimates alone.

In very recent work, Stone et al. [60] adapted unsupervised flow methods to the RAFT architecture [68] and proposed new augmentation, self-distillation, and multi-frame occlusion inpainting methods. By contrast, we use PWCnet [65], since it is the standard architecture considered in prior work, and since it can obtain strong performance with careful training [64].

**Cycle consistency.** Cycle consistency has long been used to detect occlusions [3,23,36,66,80], and is used to discount the loss of occluded pixels in unsupervised flow [25,28,32,
3. Method

We first show how to learn dense space-time correspondences using mutiscale contrastive random walks, resulting in a model that obtains high quality motion estimates via simple nonparametric matching. We then describe how the learned representation can be combined with regression to handle occlusions and ambiguity, for improved optical flow.

3.1. Multiscale contrastive random walks

We review the single-scale contrastive random walk, then extend the approach to multiscale estimation.

3.1.1 Preliminaries: Contrastive random walks

We build on the contrastive random walk (CRW) formulation of Jabri et al. [26]. Given an input video with \( k \) frames, we extract \( n \) patches from each frame and assign each an embedding using a learned encoder \( \phi \). These patches form the vertices of a graph that connects all patches in temporally adjacent frames. A random walker steps through the graph, forward in time from frames \( 1, 2, \ldots, k \), then backward in time from \( k - 1, k - 2, \ldots, 1 \). The transition probabilities are determined by the similarity of the learned representations:

\[
A_{s,t} = \text{softmax}(X_sX_t^T/\tau),
\]

for a pair of frames \( s \) and \( t \), where \( X_s \in \mathbb{R}^{n \times d} \) is the matrix of \( d \)-dimensional embedding vectors, \( \tau \) is a small constant, and the softmax is performed along each row. We train the model to maximize the likelihood of cycle consistency, i.e., the event that the walker returns to the node it started from:

\[
\mathcal{L}_{\text{CRW}} = -\frac{1}{n} \text{tr}(\log(\bar{A}_{t,t+k}\bar{A}_{t+k,t})),
\]

where the \( \log \) is elementwise and \( \bar{A}_{t,t+k} \) are the transition probabilities from frame \( t \) to \( t + k \):

\[
\bar{A}_{t,t+k} = \prod_{i=t}^{i=t+k-1} A_{s,i+1}.
\]

3.1.2 Optical flow as a random walk

After training, the transition matrix contains the probability that a pair of patches is in space-time correspondence. We can estimate the optical flow \( f_{s,t} \in \mathbb{R}^{n \times 2} \) of a patch between frames \( s \) and \( t \) by taking the expected value of the change in spatial position:

\[
g_{\text{avg}}(A_{s,t}) = \mathbb{E}_{A_{s,t}}[f_{s,t}] = A_{s,t}D - D,
\]

where \( D \in \mathbb{R}^{n \times 2} \) is the (constant) matrix containing of pixel coordinates for each patch, and \( A_{s,t}D \) is the walker’s expected position in frame \( t \).

In contrast to widely-used forward-backward cycle consistency formulations [22,88], which measure the deviation of a predicted motion from a starting point, there is no trivial solution (e.g., all-zero flow). This is because cycle consistency is measured in an embedding space defined solely from the visual content of image regions.

3.1.3 Multiscale random walk

As presented so far, this formulation is expensive to scale to high-resolutions because computing the transition matrix is quadratic in the number of nodes. We overcome this by introducing hierarchy into the search problem. Instead of comparing all pairs, we only attend on a local neighborhood. By integrating local search across scales in a coarse-to-fine manner, the model can efficiently consider a distribution over pixel-level trajectories.
Coarse-to-fine local attention. Computing the transition matrix closely resembles cost volume estimation in optical flow [6, 14, 61]. This inspires us to draw on the classic spatial pyramid commonly used for multiscale search in optical flow, by iteratively computing the dense transition matrix, from coarse to fine spatial scales \( l \in [1..L] \).

For frames \( s \) and \( t \), we compute feature pyramids \( X^l_s \in \mathbb{R}^{h(l) \times w(l) \times d} \), where \( h(l) \) and \( w(l) \) are the width and height of the feature map at scale \( l \). To match each level efficiently, we sample the features of the target frame \( X^l_t \) into the coordinate frame of \( X^l_s \) using the coarse flow from the previous level \( f_{s,t}^l \); we then compute local transition probabilities on the warped feature to account for remaining motion. Thus, we estimate the transition matrix and flow in a coarse-to-fine manner (Fig. 2), computing levels recurrently:

\[
\begin{align*}
\hat{X}_l^t &= \text{warp}(X_l^t, f_{s,t}^l) \\
\hat{A}_{l,s}^t &= \text{masked}_\text{softmax}(X_l^t \hat{X}_l^s / \tau) \\
f_{s,t}^{l+1} &= \text{upsample}(g_{\text{avg}}(A_{s,t}^l) + f_{s,t}^l),
\end{align*}
\]

where \( \text{warp}(X, f) \) samples features \( X \) with flow \( f \) using bilinear sampling, and \( f^l = 0 \). For notational convenience, we write the local transition constraint using \( \text{masked}_\text{softmax} \), which sets values beyond a local spatial window to zero. In practice, we use optimized correlation filtering kernels to compute Eq. 5 efficiently, and represent the transition matrices \( A_{s,t}^l \) as a sparse matrix.

**Loss.** After computing the transition matrices between all pairs of adjacent frames, we sum the contrastive random walk loss over all levels:

\[
\mathcal{L}_{\text{msCRW}} = - \sum_{l=1}^{L} \frac{1}{n_l} \text{tr}((\hat{A}_{l,s,t+h}^l \hat{A}_{l,t+h,k}^l)),
\]

where \( \hat{A}_{l,s,t} \) is defined as in Eq. 2 and \( n_l \) is the number of nodes in level \( l \). In our experiments, we use \( L = 5 \) scales and consider \( k \in [2..4] \) length cycles.

### 3.1.4 Smooth random walks

Since natural motions tend to be smooth [54], we follow work in optical flow [61], and incorporate smoothness as an additional desiderata for our random walks. We use the edge-aware loss of Jonschkowski et al. [32], which penalizes spatial changes in flow near similarly-colored pixels:

\[
\mathcal{L}_{\text{smooth}} = \mathbb{E}_p \sum_{d \in \{x,y\}} \exp(-\lambda_c I_d(p)) \frac{|\partial^2 I_{s,t}(p)|}{|\partial d|^2} \tag{8}
\]

where \( I_d(p) = \frac{1}{d} \sum_{c=1}^{d} |\partial c| \) is the spatial derivative averaged over all color channels \( |\partial c| \) in direction \( d \). The parameter \( \lambda_c \) controls the influence of similarly colored pixels. We apply this loss to each scale of the model.

### 3.2 Handling occlusion

While effective for most image content, nonparametric matching has no mechanism for estimating the motion of pixels that become occluded, since it requires a corresponding patch in the next frame. We propose a variation of the model that combines the multiscale contrastive random walk with a regression module that directly predicts the flow values at each pixel.

The architecture of our regressor closely follows the refinement module of PWC-net [65]. We learn a function \( g_{\text{reg}}(\cdot) \) that regresses the flow at each pixel from the multiscale contrastive random walk cost-volume and convolutional features. These features are obtained from the same shared backbone that is used to compute the embeddings. We provide a diagram in the supplement.

**Regression loss.** We train the regressor using a loss that closely resembles the contrastive random walk objective. Under this loss, pixels that are already well-matched by the nonparametric model (e.g., non-occluded pixels) will be unlikely to change their flow values, while poorly-matched pixels (e.g., occluded pixels) will obtain their flow values using a smoothness prior. We use the same smoothness loss, \( \mathcal{L}_{\text{smooth}} \), as the nonparametric model (Eq. 8), and penalize the model from deviating from the nonparametric flow estimate (Eq. 3). We also use our learned embeddings as features for a learned photometric loss, i.e., incurring loss if the model puts two pixels with dissimilar embedding vectors into correspondence. This results in an additional loss:

\[
\mathcal{L}_{\text{feat}} = ||X_s - \text{warp}(X_t, f_{s,t})||^2 + \lambda_v ||f_{s,t} - g_{\text{avg}}(A_{s,t})||^2, \tag{9}
\]

where \( f_{s,t} \) is the predicted flow and \( \lambda_v \) is a constant. To prevent the regression loss from influencing the learned embeddings that it is based on, we do not propagate gradients from the regressor to the embeddings \( X \) during training. As in Eq. 7, we apply the loss to each scale and sum.
Augmentation and masking. We follow [32] and improve the regressor’s handling of occlusions through augmentation, and by discounting the loss of pixels that fail a consistency check [42]. To handle pixels that move off-screen, we compute flow, then randomly crop the input images and compute it again, penalizing deviations between the flows. This results in a new loss, $L_{\text{bound}}$. We also remove the contribution of pixels in the photometric loss (Eq. 9) if they have no correspondence in the backwards-in-time flow estimation from $t$ to $s$. Both are implemented exactly as in [32]. These losses apply only to the regressor, and thus do not directly affect the nonparametric matching.

3.3. Training

Objective. The pure nonparametric model (Section 3.1) can be trained by simply minimizing the multiscale contrastive random walk loss with a smoothness penalty:

$$L_{\text{non}} = L_{\text{msCRW}} + L_{\text{smooth}}.$$  

(10)

Adding the regressor results in the following loss:

$$L_{\text{reg}} = L_{\text{msCRW}} + L_{\text{smooth}} + L_{\text{feat}} + L_{\text{bound}}.$$  

We include weighting factors to control the relative importance of each loss (specified in the supplement).

Architecture. To provide a straightforward comparison with unsupervised optical flow methods, we use the PWC-net architecture [65] as our network backbone, after reducing the number of filters [32]. This network uses the feature hierarchy from a convolutional network to provide the features at each scale. We use the cost volume features from this network as the embedding for the random walk, $X^s_t$, after performing $\ell_2$ normalization. We also use its regressor architecture. We provide architecture details in the supplementary material.

Subcycles. We follow [26] and include subcycles in our contrastive random walks: when training the model on $k$-frame videos, we include losses for walks of length $k, k-1, ... 2$. These losses can be estimated efficiently by reusing the transition matrices for the full walk.

Multi-frame training. When training with $k > 2$ frames, we use curriculum learning to speed up and stabilize training. We train the model to convergence with 2, 3, ... $k$ frame cycles in succession.

Optimization. To implement the contrastive random walk, we exploit the sparsity of our coarse-to-fine formulation, and represent the transition matrices $A_{k^c,t}$ as sparse matrices. This significantly improved training times and reduced memory requirements, especially in the finest scales. It takes approximately 3 days to train the full model on one GTX2080 Ti. We train our network with PyTorch [50], using the Adam [34] optimizer with a cyclic learning rate schedule [59] with a base learning rate of $10^{-4}$ and a max learning rate of $5 \times 10^{-4}$. We provide training hyperparameters in supplementary material.

Avoiding shortcuts. The contrastive random walk can potentially obtain shortcut solutions when it is trained with a fully convolutional network by exploiting positional information [26]. While recent work has shown this can be solved through augmentation [67], we found that we avoided trivial shortcuts when using reflection padding in our network (for all convolution layers except for in the regressor). This may be because we simultaneously optimize multiple losses and use a limited search window, making the trivial solution harder to find.

4. Results

Our model produces two outputs: the optical flow fields and the pixel trajectories (which are captured in the transition matrices). We evaluate these predictions on label transfer and motion estimation tasks. We compare them to space-time correspondence learning methods, and with unsupervised optical flow methods.

4.1. Datasets

For simple comparison with other methods, we train on standard optical flow datasets. We note that the training protocols used by unsupervised optical flow literature are not standardized. We therefore follow the evaluation setup of [32]. We pretrain models on unlabeled videos from the Flying Chairs dataset [11]. We then train on the KITTI-2015 [16] multi-view extension and Sintel [10]. To evaluate our model’s ability to learn from internet video, we also trained the model on YouTube-VOS [81], without pretraining on any other datasets.

We also evaluate our model on standard label transfer tasks. The JHMDB benchmark [31] transfers 15 body parts to future frames over long time horizons. The DAVIS benchmark [52] transfers object masks.

4.2. Label propagation

We evaluate our learned model’s ability to perform video label propagation, a task widely studied in video representation learning work. The goal is to propagate a label map provided in an initial video frame, which might describe keypoints or object segments, to the rest of the video.

We follow Jabri et al. [26] and use our model’s probabilistic motion trajectories to guide label propagation. We infer the labels for each target frame $t$ auto-regressively. For each previous source frame $s$, we have a predicted label map $L_s \in \mathbb{R}^{n \times c}$, where $n$ is the number of pixels and $c$ is the number classes. As in [26], we compute $K_{t,s}^l$, the matrix of weights for attention on the source frames, by keeping the top-$k$ logits in each row of $A_{k^c,t}^l$. We use $K_{t,s}$ as an attention matrix for each label, i.e., $L_t = K_{t,s}^l L_s$. Using several source frames as context allows for overcoming occlusion.

We use variations of our model that was trained on the unlabeled Sintel and YouTube-VOS datasets, and use the transition matrix and flow fields at the penultimate level of
the pyramid, i.e. level 4. Since the transition matrix describes residual motion, we warp (i.e. with \( f_{i,s}^{t} \)) each label map before querying. Finally, since the features in level 4 have only 16 channels, we stack features from levels 3 and 4 to obtain hypercolumns [19], before computing attention.

Evaluation. We compared our model to recent video representation learning work, including single-scale CRW [26], UVC [38], and the state-of-the-art VFS [79] (Tab. 1). We also report two baselines that chain optical flow: unsupervised UFlow [32] and supervised RAFT [68].

For pose propagation, we evaluate our model on JHMDB [31] and report the PCK metric, which measures the fraction of keypoints within various distances to the ground truth. Our approach outperforms existing self-supervised approaches on this benchmark, especially at the stringent PCK@0.05, which is typically not reported. Note that our approach uses a significantly smaller network. While our model improves on fine-grained matching, it still struggles with occlusions (like other methods), which tend to involve large motions and motion blur (see Fig. 3 left). For object propagation, we evaluate our model on the DA VIS [52] benchmark, and report the mean of \( J \) and \( F \) metrics [51], which characterize segment overlap and boundary precision, respectively. Despite our focus on scaling to fine-grained matching, the model achieves competitive performance on DA VIS, outperforming the cycle-consistency method TimeCycle [75]. In attention visualizations of the propagated label distribution, we see that the transition distribution is robust to momentary occlusion (Fig. 3 mid bottom), but can nevertheless suffer from drift (Fig. 3 right bottom). Interestingly, our model significantly outperforms the two optical flow methods, suggesting that the “soft” motion trajectories provided by our model may convey useful information for propagation that is not captured by the flow.

4.3. Optical flow

We evaluate our model’s ability to predict optical flow.

4.3.1 Nonparametric motion estimation

Our model is able to estimate motion solely through nonparametric matching, as can be seen qualitatively in Figure 4. Despite the model’s simplicity and the fact that it is based on very different principles than existing flow methods, it obtains strong performance on matching non-occluded pixels (Tab. 2). It outperforms many unsupervised optical flow models, such as SelFlow [44] (Tab. 4) on KITTI noc metric (non-occluded endpoint error). We see that the full regression-based variation of the model obtains better results, particularly on the all metric.

To help understand the importance of our multiscale formulation, we compared to Jabri et al. [26] on flow benchmarks, using their publicly released model (Tab. 2). This model resembles our nonparametric model, but with the random walk occurring at a single scale, with no smoothness prior. We evaluate the model using dense features, as in their approach. We found that our model significantly outperforms it. To control for training differences, we also tried removing scales from multiscale training (Tab. 6), finding that this significantly reduced performance.

4.3.2 Effects of multi-frame cycles

We asked how the quality of the representation changes as we vary the number of frames \( k \) used to train the random walk. We test all models on 2-frame optical flow. As seen in Table 6, the model obtains better performance on all metrics using 3-frame and 4-frame cycles.

4.3.3 Photometric feature learning

In contrast to other effective unsupervised optical flow approaches, our model does not use hand-crafted features. We evaluate the quality of our learned features when used as a photometric loss, compared to other common designs (Tab. 5). First, we compared our model to a variation of our
Figure 4. Optical flow qualitative results. We show results on images not seen during training, using our nonparametric-only and regression-based models. The highlighted regions show significant differences between the regression-based models. The optical flow vectors are coded by colors.

model that uses raw pixels (rather than hand-crafted features) within a photometric loss, with a robust Charbonnier penalty [61]. This baseline model is very similar to the Charbonnier variation of UFlow [32], though to control for other differences we use our own network. This amounts to simply replacing $\mathcal{L}_{\text{feat}}$ with the Charbonnier loss (which disables the contrastive random walk). We found that the resulting model performed significantly worse.

Next, we considered using a state-of-the-art hand-crafted feature, the Census transform [47], resulting in a model similar to the Census variation of UFlow. We found that our features obtained competitive performance on non-occluded pixels, but that there was a significant advantage to Census features on the all metric. This is understandable since the contrastive random walk does not have a way of learning features for occluded pixels. Interestingly, we found that combining the two features together improved performance, and that the gap improves further when multi-frame walks are used, obtaining the overall best results.

Moreover, the combined features show more robustness on image pairs with rapid exposure and hue changes. We evaluated models trained on hue- and brightness-jittered image pairs and found that the model with our learned features performed significantly better, and that the gap increased with the magnitude of the jittering (Tab. 4b). Please see the supplement for details.

Finally, we used our learned features as part of the photometric loss for ARFlow [41], a recent unsupervised flow model. We added the $\mathcal{L}_{\text{feat}}$ loss to their model and retrained it (while jointly learning the features through a contrastive random walk). The resulting model improves performance on all metrics, with a larger gain on Sintel (Tab. 4).

Table 2. Nonparametric motion estimation. Our single-scale, nonparametric, and regression-based methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>KITTI-15 train</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>noc</td>
</tr>
<tr>
<td>Jabri et al. [26]</td>
<td>12.63</td>
</tr>
<tr>
<td>Ours - Nonparametric</td>
<td>2.18</td>
</tr>
<tr>
<td>Ours - With Regression</td>
<td>2.09</td>
</tr>
</tbody>
</table>

Table 3. Model configurations. All models are 2-frame. Regressor-only finds a shortcut solution, predicting zero flow.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>KITTI-15 train</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>noc</td>
</tr>
<tr>
<td>Full</td>
<td>2.09</td>
</tr>
<tr>
<td>Nonparametric only</td>
<td>2.18</td>
</tr>
<tr>
<td>No feature consistency in $\mathcal{L}_{\text{feat}}$</td>
<td>2.62</td>
</tr>
<tr>
<td>Regressor only</td>
<td>14.21</td>
</tr>
<tr>
<td>No regressor constraint in $\mathcal{L}_{\text{feat}}$</td>
<td>5.54</td>
</tr>
<tr>
<td>No $\mathcal{L}_{\text{smooth}}$</td>
<td>2.14</td>
</tr>
<tr>
<td>No $\mathcal{L}_{\text{bound}}$</td>
<td>10.98</td>
</tr>
</tbody>
</table>

4.3.4 Motion estimation ablations

To help understand which properties of our model contribute to its performance, we perform an ablation study on KITTI-15 (Table 3). We asked how the different losses contribute to the performance. We ablated the smoothness loss (Eq. 8), the self-supervision loss (Sec. 3.2), and removing the constraint on the regressor in Eq. 9 by setting $\lambda_0 = 0$. We see that the smoothness loss significantly improves results. Similarly, we discarded the feature consistency term from Eq. 9, which reduces the quality of the results but outperforms the nonparametric model on the all metric.

Training on internet video. We found that our model generalized well to benchmark datasets when training solely on YouTube-VOS [81] (Tab. 7). For comparison, we also trained ARFlow [41] on YouTube-VOS. Our model obtained better performance on KITTI, while ARFlow per-
Table 5. **Photometric features.** We evaluate the effectiveness of our features when they are used to define a photometric loss.

<table>
<thead>
<tr>
<th>Losses</th>
<th>KITTI-15 train noc</th>
<th>all</th>
<th>ER%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charbonnier</td>
<td>2.28</td>
<td>5.69</td>
<td>19.30</td>
</tr>
<tr>
<td>Census</td>
<td>2.05</td>
<td>3.14</td>
<td>11.04</td>
</tr>
<tr>
<td>Ours feats.</td>
<td>2.09</td>
<td>3.86</td>
<td>12.45</td>
</tr>
<tr>
<td>Ours feats. + Charbonnier</td>
<td>2.51</td>
<td>4.57</td>
<td>14.25</td>
</tr>
<tr>
<td>Ours feats. + Census</td>
<td>2.04</td>
<td>3.10</td>
<td>10.89</td>
</tr>
<tr>
<td>Ours feats. (3-frame) + Census</td>
<td>2.02</td>
<td>3.03</td>
<td>10.67</td>
</tr>
</tbody>
</table>

Table 6. **Contrastive random walk ablations.** We evaluate different model parameters, including cycle length and number of scales in the multiscale random walk (from coarse to fine).

<table>
<thead>
<tr>
<th></th>
<th>KITTI-15 train noc</th>
<th>all</th>
<th>ER%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cyc. len</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2.09</td>
<td>3.86</td>
<td>12.45</td>
</tr>
<tr>
<td>3</td>
<td>2.05</td>
<td>3.46</td>
<td>12.28</td>
</tr>
<tr>
<td>4</td>
<td>2.04</td>
<td>3.39</td>
<td>12.14</td>
</tr>
</tbody>
</table>

Table 7. **Training on internet video.** We train on YouTube-VOS [81] and evaluate on optical flow benchmarks.

<table>
<thead>
<tr>
<th></th>
<th>Sintel train</th>
<th>KITTI-15 train noc</th>
<th>all</th>
<th>ER%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clean</td>
<td>Final</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ours (2-cycle)</td>
<td>3.37</td>
<td>4.65</td>
<td>2.32</td>
<td>5.73</td>
</tr>
<tr>
<td>ARFlow [41]</td>
<td>3.22</td>
<td>4.51</td>
<td>2.65</td>
<td>6.01</td>
</tr>
</tbody>
</table>

formed better on Sintel.

### 4.3.5 Comparison to recent optical flow methods

To help understand our model’s overall performance, we compare it to recent optical flow methods (Table 4). We include models that use different numbers of frames for the random walk, and a variation of ARFlow [41] that uses our self-supervised features to augment its photometric loss.

We found that our models outperform many recent unsupervised optical flow methods, including the (3-frame) MFOccFlow [29] and EPIFlow. In particular, we significantly outperform the recent SelFlow method [44], despite the fact that it takes 3 frames as input at test time and uses Census Transform features. By contrast, our model uses no hand-crafted image features. The highest performing method is the very recent, highly optimized SMURF model [60], which uses the RAFT [68] architecture instead of PWC-net and which extends UFlow [32]. This model uses a variety of additional training signals, such as extensive data augmentation, occlusion inpainting with multi-stage training, self-distillation, and hand-crafted features.

### 5. Discussion

We have proposed a method for learning dense motion estimation using multiscale contrastive random walks. We see our work as a potential step toward unifying self-supervised tracking and optical flow. Moreover, the model can learn from internet video, which suggests that the emergent representations of such a hierarchical tracker may learn interesting part-whole structure at scale.

**Limitations and impact.** Motion analysis has many applications, such as in health monitoring, surveillance, and security. There is also a potential for the technology to be used for harmful purposes if weaponized. The released models are limited in scope to the datasets used in training.

**Acknowledgements.** We thank David Fouhey and Jeff Fessler for their helpful feedback. AO thanks Rick Szeliski for introducing him to multi-frame optical flow. This research was supported in part by Toyota Research Institute, Cisco Systems, and Berkeley Deep Drive.
References


