박 사 학 위 논 문
Ph.D. Dissertation

딥 특징 및 물리 기반 비디오 모션 학습 향상

Advancing Video Motion Learning with Deep Features and Physics-Based Priors

2024

임 우 빈 (任 佑 彬 Im, Woobin)

한 국 과 학 기 술 원

Korea Advanced Institute of Science and Technology

박 사 학 위 논 문

딥 특징 및 물리 기반 비디오 모션 학습 향상

2024

임 우 빈

한 국 과 학 기 술 원

전산학부

딥 특징 및 물리 기반 비디오 모션 학습 향상

임 우 빈

위 논문은 한국과학기술원 박사학위논문으로
학위논문 심사위원회의 심사를 통과하였음

2024년 5월 2일

심사위원장 윤 성 의 (인)

심 사 위 원 김 태 균 (인)

심 사 위 원 안 성 진 (인)

심 사 위 원 조 성 호 (인)

심 사 위 원 성 민 혁 (인)

Advancing Video Motion Learning with Deep Features and Physics-Based Priors

Woobin Im

Advisor: Sung-Eui Yoon

A dissertation submitted to the faculty of
Korea Advanced Institute of Science and Technology in
partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Computer Science

Daejeon, Korea
May 10, 2024

Approved by

Sung-Eui Yoon
Professor of School of Computing

The study was conducted in accordance with Code of Research Ethics[1].

[1] Declaration of Ethical Conduct in Research: I, as a graduate student of Korea Advanced Institute of Science and Technology, hereby declare that I have not committed any act that may damage the credibility of my research. This includes, but is not limited to, falsification, thesis written by someone else, distortion of research findings, and plagiarism. I confirm that my thesis contains honest conclusions based on my own careful research under the guidance of my advisor.

DCS 임우빈. 딥 특징 및 물리 기반 비디오 모션 학습 향상. 전산학부 . 2024년. 70+iv 쪽. 지도교수: 윤성의. (영문 논문)
Woobin Im. Advancing Video Motion Learning with Deep Features and Physics-Based Priors. School of Computing . 2024. 70+iv pages. Advisor: Sung-Eui Yoon. (Text in English)

초 록

본 학위논문은 비디오에 나타나는 모션을 이해하는 핵심이 되는 비디오 모션 러닝에 대해 탐구한다. 이를 위해, 광학흐름과 운동학 필드의 두 가지의 모션 표현에 대해서 다룬다.

광학흐름은 비디오의 2차원 모션에 대해 다룬다. 광학 흐름은 주로 딥 네트워크에 의해 학습되는데, 이때 정확한 광학흐름 네트워크 학습을 위한 참값을 얻는 것은 어렵다. 본 학위논문의 연구는 이러한 배경을 고려하여, 학습을 위한 참값이 존재하지 않는 경우에 딥러닝으로 광학흐름을 학습하는 일련의 연구를 수행한다. 첫째, 비지도학습(unsupervised learning)을 이용하여 참값이 존재하지 않는 경우에도 새로운 유사도 기법을 제안하여 비지도학습 손실 함수를 통한 광학흐름을 개선하는 연구를 수행한다. 둘째, 준지도학습(semi-supervised learning)을 통해 렌더링으로 생성한 학습 참값 데이터를 활용하여 참값이 없거나 적은 목표 도메인으로 네트워크를 적응시키는 연구를 수행한다.

광학흐름은 2차원 비디오 프레임에서 모션을 인식하지만, **운동학 필드**는 3차원 공간에서 시간에 따른 움직임을 표현한다. 본 연구는 최근 4차원 방사 필드 기술의 진보를 이용하여 모션 필드인 운동학 필드를 운동학에 기반하여 학습한다. 이를 위해 비디오 데이터에서 방사 필드와 운동학 필드의 물리 기반 통합적 학습 방법론을 제시한다. 이러한 방법론은 더 정확한 모션 인식과 기하정보 재건을 가능케하여 3차원 비디오 재건을 향상한다.

핵 심 낱 말 광학흐름, 비지도학습, 준지도학습, 동적 방사 필드.

Abstract

This dissertation explores the video motion learning, a key element of computer vision that enables the understanding of motion within videos. We focus on two different motion representations: optical flow and kinematic fields.

Optical flow describes 2D motion in a video, which can be learned by a deep networks. The challenge lies in the acquisition of ground-truth data necessary for training deep networks to accurately learn optical flow. Thus, this dissertation presents a series of research aiming to the absence of ground-truth in deep learning of optical flow, mainly powered by self-supervised deep features. First, we study unsupervised optical flow to investigate a learning situation where no ground-truth dataset is available, where we propose a novel similarity measure for the unsupervised objective function. Second, we propose a semi-supervised approach for optical flow learning, which can effectively utilize a synthetic dataset with ample rendered ground-truth samples, then can adapt to unlabeled target domain which is unlabeled or labeled on few samples.

While optical flow captures motion within 2D frames, **kinematic fields** represent motion in 3D space over time. Our research capitalizes on the recent progress in spatio-temporal radiance field techniques to refine motion fields inspired by kinematics, referred to as kinematic fields. A novel approach for the joint learning of radiance and kinematic fields from video data is introduced, grounded on physics-based priors. This method not only enables more precise motion capture but also improves the reconstruction of geometry, leading to enhanced 3D video reconstruction.

Keywords Optical flow, unsupervised learning, semi-supervised learning, dynamic radiance fields.

Contents

List of Tables

List of Figures

Chapter 1. Introduction

Perceiving motion plays an important role for our visual system to understand our surroundings. For instance, we decide how dangerous a car is by its velocity, perceived by our visual system. Thus, computer vision researchers have devised a number of vision tasks to design a suitable algorithm for motion understanding. For example, action recognition is one of the tasks, where movement of objects decides a category of a video clip. An interesting discovery in action recognition is that a two-stream architecture which processes color appearance and low-level motion of a scene separately is essential for a better recognition accuracy [11, 12]. In addition to the two-stream architecture, neuroscientists have claimed that the motion perception is separate to other visual perception (e.g., color), so that a patient can lose only the ability to perceive motion even with the intact perception of color [13]. The evidences in the aforementioned studies indicate that a low-level motion perception technique is critical in computer vision, as it is in the biological vision system.

In exploring how computer vision systems can learn to perceive motion from videos (i.e., video motion learning), our research examines two different motion representations: optical flows and kinematic fields. Optical flow represents 2D motion within video frames. Our optical flow research focuses on leveraging deep feature priors to enhance self-supervised optical flow learning in both unsupervised and semi-supervised settings. Additionally, we investigate the representation of kinematic fields, motivated by the understanding that motion in natural videos is three-dimensional and governed by physical laws. Our findings suggest that kinematic fields, with their inherent priors about the three-dimensional world and kinematics, can improve the reconstruction of 3D structures and motion from 2D information.

1.1 Learning Optical Flow using Deep Feature Priors

Optical flow is a low-level computer vision task for motion perception, which has been a fundamental step towards developing certain motion-related tasks. The definition of optical flow is the displacement of a point within two consecutive frames, where the point is often all the pixels in an image, resulting in a dense optical flow field. Since the optical flow field usually has the same dimension with the given image, it gives a few useful properties. For example, we can use the same system to process the image and the optical flow field [11], and optical flow can be used to modify images in pixel-wise manner [14]. Thanks to the useful properties, optical flow has fertilized various computer vision tasks, such as activity recognition [11, 12], video inpainting [14], novel pose generation [15], video stabilization [16].

Estimating optical flow from videos is a long-standing problem and it has been studied for decades. Basically, the objective of optical flow is to match the same visual patterns in two images, which can be formulated with the brightness constancy assuming that the brightness of a point does not change by a movement [17]. However, the brightness constancy cannot fully solve the pixel-wise correspondence due to the aperture problem. Thus, early stage of optical flow research focuses on how to solve the ambiguity by smoothness assumption [17] and combining neighboring pixels [18]. Until 2010s, a line of research based on the brightness assumption had dominated [19, 20], until deep learning-based methods showed promising results.

The concept of learning optical flow from a training dataset is explored in the line of the brightness assumption work, where a filter comparing brightness patterns is learned from training data [21]. Since the

brightness assumption is often violated in the videos, the learning-based method enables to compensate for the violated values and shows the benefits compared to non-learnable counterparts. Apart from the violation of brightness constancy and the aperture problem, real-world videos have a lot more to overcome: motion blur, occlusion, deformation, and so on. In terms of the challenges in the wild, deep convolutional networks (CNNs) are shown to be highly effective in optical flow estimation, as they are in various other vision tasks.

Deep learning of optical flow is dominating top-rank of optical flow benchmarks [22, 23], whose testing samples usually contain challenging motion blur and lighting conditions. Deep networks for optical flow estimation [24, 25, 26, 27] have made architectural progresses, resulting in significant improvements in terms of flow accuracy, efficiency, and generalization capability.

For supervised optical flow training, large-scale synthetic datasets, e.g., FlyingChairs [24], have been primarily used. While there are real-world datasets including Middlebury [9] and KITTI [28, 23], their sizes are limited to few hundred images and each of them is limited to a specific scenario. This limitation is mainly due to the extremely prohibitive cost to obtain or manually label accurate matching points in thousands of video frames in the wild.

In this dissertation, our research focuses on deep learning of optical flow, when we do not have access to the training labels required for supervised learning.

For the purpose, unsupervised learning of optical flow has been studied extensively from the research community [29, 30]; we can plug in the brightness assumption from the classical methods [17] to loss functions for deep learning. We propose a novel unsupervised optical flow learning by deep feature similarity (Chap. 2). Deep unsupervised learning for optical flow uses a loss function which measures image similarity with the warping function parameterized by estimated flow. The census transform, instead of image pixel values, is often used for the image similarity. In our work, rather than the handcrafted features i.e. census or pixel values, we propose to use deep self-supervised features with a novel similarity measure, which fuses multi-layer similarities.

Although the unsupervised method shows remarkable results even without a label, supervised methods which only utilize a large-scale synthetic dataset outperforms the unsupervised work in some cases [8, 31]. Thus, we propose to improve the supervised network on unlabeled target domain by a semi-supervised learning strategy (Chap. 3) to take advantage of both supervised and unsupervised strategies. A training pipeline for optical flow CNNs consists of a pretraining stage on a synthetic dataset followed by a fine tuning stage on a target dataset. We propose a practical semi-supervised fine tuning method to adapt a pretrained model to a target dataset without ground truth flows, which has not been explored extensively. Specifically, we propose a flow supervisor for self-supervision, which consists of parameter separation and a student output connection. This design is aimed at stable convergence and better accuracy over conventional self-supervision methods which are unstable on the fine tuning task.

1.2 Learning 3D Motion using Physics-based Priors

Dynamic radiance fields have been used to reconstruct 3D spatio-temporal appearance from 2D videos. Based on the volume-rendering technique [32], dynamic radiance fields enable novel view and time synthesis; shortly, it makes 3D video from 2D video inputs.

In scenarios where scenes are captured with multi-view cameras, dynamic radiance fields generally present geometrically accurate and temporally consistent appearances [33]. However, the requirement for multi-view camera setups is often impractical for many video capturing environments. Consequently,

research has focused on developing dynamic radiance fields from monocular videos [34, 35, 36].

For monocular video reconstructions, integrating the learning of appearance with motion has proven beneficial. For example, the Neural Scene Flow Fields (NSFF) model [34] utilizes a displacement field to maintain temporal coherence across different frames.

Based on the physics-based nature of the real world, we introduce kinematic fields, a novel physics-based motion representation (Chap. 4). Kinematic fields are defined with the kinematic quantities, including velocity, acceleration, jerk, and so on. These quantities are learned in an unsupervised manner by the photometric loss intertwined with dynamic radiance fields. Thanks to the physics-based design of kinematic fields and learning objectives, our dynamic radiance fields and kinematic fields presents better appearance and motion than the baseline models which are not physics-based.

Unsupervised optical flow allows an artificial intelligence model to learn to perceive motion autonomously, without the need for external supervision. This figure is created by DALL.E (ChatGPT).

Chapter 2. Unsupervised Learning of Optical Flow

In computer vision, optical flow estimation is a fundamental step towards motion understanding. It describes the velocity of each point in the 3D world as the projection of points to the 2D motion field. Thanks to its effective motion description, it has been largely used for many applications, e.g., video recognition [11, 37], frame interpolation [9, 38], and inpainting [14], to name a few.

Recently, end-to-end deep networks for optical flow estimation [24, 25, 26, 27] have made architectural progresses, resulting in significant improvements in terms of flow accuracy, efficiency, and generalization capability. For supervised optical flow training, large-scale synthetic datasets, e.g., FlyingChairs [24], have been primarily used. While there are real-world datasets including Middlebury [9] and KITTI [28, 23], their sizes are limited to few hundred images and each of them is limited to a specific scenario. This limitation is mainly due to the extremely prohibitive cost to obtain or manually label accurate matching points in thousands of video frames in the wild.

To train deep networks without ground-truth flows, unsupervised approaches have been proposed. In principle, unsupervised methods exploit an assumption that two matching points have similar features and learn to generate flows maximizing the similarity. In this line of research, choosing appropriate features is critical for accurate optical flow estimation. The early work [29, 39] applies RGB pixel values and image gradients as the feature, and recently it has been shown that the census transform [40] is highly effective for optical flow learning [30, 1].

Another interesting aspect of the unsupervised optical flow networks is that a network learns more than just the loss it is trained with. A recent work [1] found that configuring the loss function only with

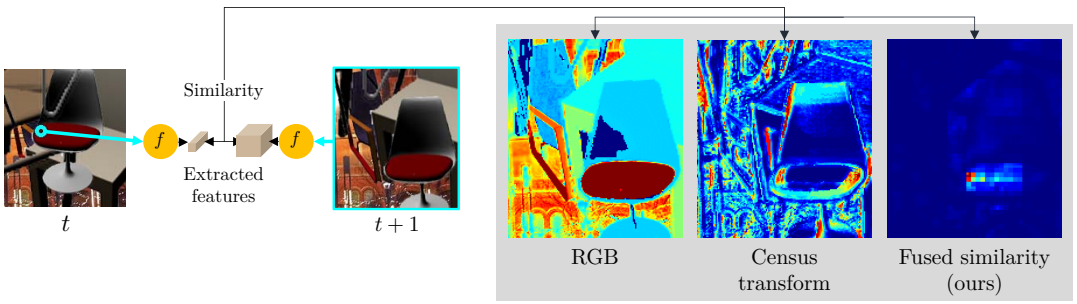


Figure 2.1: This figure shows different similarity maps of the reference point at time step t to all pixels in the target image at $t+1$; red means higher similarity. We compare the similarity computed by our deep feature against ones computed by census transform and RGB. The fused similarity shows improved discriminative response, while the census transform tends to be sensitive to local edge appearance and RGB shows high similarity with similar colors. For simple visualization, we compute similarity in the spatial domain

the data term does work without the smoothness term. This observation implies that the network feature learns meaningful patterns in moving objects only with the photometric constancy assumption. A similar observation is also found in the literature on self-supervision [41, 42], where deep features learn semantic patterns by conducting simple unsupervised tasks like a jigsaw puzzle or guessing rotation.

In this work, we learn self-supervised network features and use them to improve the unsupervised optical flow. To learn from self-features, we propose to use the similarity based on the product fusion of multi-layer features (Sec. 2.2.2). We visualize similarity maps computed by different features in Fig. 2.1. Our fused similarity demonstrates discriminative matching points highlighting the matching pair, while lessening unmatched areas. On the other hand, the similarity map (e.g., computed by the cosine) with RGB or the census transform shows many high-response points across the whole area. This is mainly because those features only encode patterns in a local area and do not represent semantic meanings. We propose three loss functions utilizing the feature similarity for optical flow training (Sec. 2.2.3). Across various quantitative and qualitative validations, we demonstrate the benefits of the proposed feature similarity and the loss function. (Sec. 2.3). When compared to other deep unsupervised methods, our method achieves state-of-the-art results under various measures across FlyingChairs, MPI Sintel, and KITTI benchmarks.

2.1 Related Work

End-to-end supervised deep methods. FlowNet [24] is the first end-to-end framework that exploits a deep network for optical flow estimation. To train the network, a large-scale labeled dataset called FlyingChairs was constructed [24]. Following the first work, FlowNet2 [25], SpyNet [26], and PWC-Net [27] have made architectural progresses, resulting in significant improvements in terms of flow accuracy, efficiency, and generalization capability.

Datasets. Large-scale synthetic datasets are available for supervised optical flow training including FlyingChairs [24]. Following FlyingChairs, FlyingThings3D [43] and FlyingChairs-Occ [44] datasets are incorporated into the collection with improved reality and additional information. Real-world datasets annotated with a rich optical flow are lacking. Middlebury [9] and KITTI [28, 23] are the most commonly used ones. However, not only the scenes they have are limited to few hundreds of images, but also they

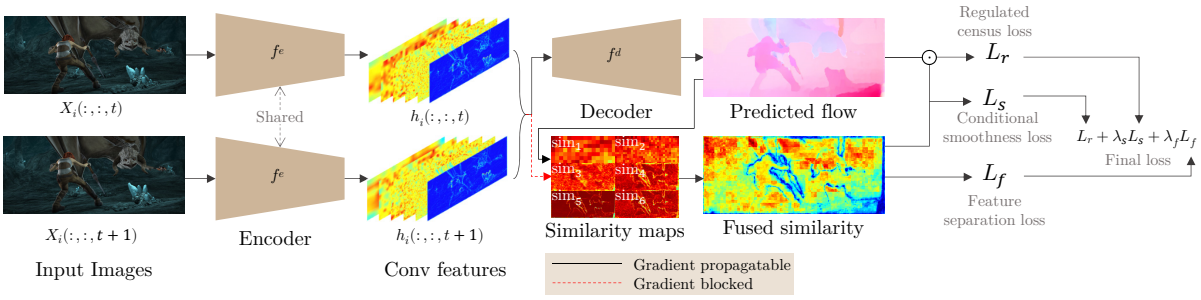


Figure 2.2: This shows the overview of our method, which is end-to-end trainable for both optical flow and self-supervised deep features

are constrained to specific scenarios: indoor static objects for Middlebury and driving for KITTI. This limitation is mainly due to the prohibitive cost to obtain or manually label accurate matching points in thousands of video frames in the wild.

End-to-end unsupervised deep methods. To make use of deep networks for optical flow without expensive ground-truth flows, deep unsupervised approaches have been proposed. Earlier methods [29, 39] brought ideas from the classical variational methods, which adopt the energy functional containing the data and smoothness terms into loss functions of deep learning. The loss function in unsupervised methods can be calculated using the warping technique [45].

In unsupervised optical flow learning, how to filter out unreliable signals from its loss is critical for achieving better results. In terms of the data term, the census transform [40] has been proven to be effective in deep unsupervised optical flow [30, 1]. Similarly, occlusion handling [30, 46] can eliminate possibly occluded points from the loss calculation, where no target pixels exist due to occlusion. Rather than just giving up supervision on occluded ones, hallucinated occlusion [1, 5] can be helpful to give meaningful loss for those occluded points. Additionally, training with multiple frames improves the noisy loss signal. Janai et al. [47] use the constant velocity assumption and Liu et al. [5] build multiple cost-volumes to give more information to a model.

Deep features for matching. In the fields of matching and tracking keypoints or objects, deep network features have been used for robust matching [48, 49, 50]. At a high level, matching techniques are related to optical flow estimation [51, 20, 52, 53, 54, 55]. However, addressing all the pixels and their matching in a dense manner (cf. sparse keypoints) is challenging. To the best of our knowledge, deep features have not been exploited in objective functions for optical flow estimation. On the other hand, there has been successful exploitation of deep features in pixel-wise tasks, e.g., depth estimation [56] and generation by warping [57]. In our evaluations, the Zhan's method [56], which was proposed for stereo matching, or the losses more direct to optimize the deep features, e.g., the triplet losses, have shown poor performance in unsupervised optical flow estimation. Given the high level of noise due to occlusion, deformation, motion blur, and the unsupervised settings of optical flow, the level of optimization, i.e., the loss function for features, needs to be carefully designed.

2.2 Approach

In this work, we successfully learn and exploit deep features for improving unsupervised optical flow estimation. The proposed framework simultaneously improves the features and optical flow while adding a small additional cost for training and no extra cost at runtime. For effective training, we build new

loss functions (Sec. 2.2.3) utilizing the fused similarity (Sec. 2.2.2). The feature separation loss separates certain and uncertain matchings by our fused similarity like a contrastive loss. Noting that even the best features can fail in cases like occlusions, the separation mechanism effectively improves resulting flows by discouraging to match uncertain pixels, as well as encoraging to improve certain ones. Additionally, the regulated census loss and conditional smoothness loss make use of the census features and the smoothness constraint adaptively considering the deep similarity to compensate for the low precision of deep features.

Figure 2.2 illustrates our method. Our method is an end-to-end trainable network, which takes a sequence of images as an input for optical flow estimation; we use PWC-Net [27] structure as a base model of the encoder and the decoder, and train it using self-features, i.e., spatial conv features. In the training phase, we add a similarity branch in which similarities of predicted flows are calculated and fused; the resulting fused similarity is actively used in our training process. The aggregated similarity over layers resolves disagreement among multiple-layer features. We apply three loss functions to be minimized upon the fused similarity: feature separation loss, regulated census loss, and conditional smoothness loss. In the learning process, both the encoder and decoder are initialized using the conventional unsupervised optical flow loss. Then, the proposed feature-based losses are minimized. The method converges under different initialization and parameter settings.

2.2.1 Background on Unsupervised Optical Flow Learning

The learning-based optical flow method commonly works on a dataset to train a model that has a set of spatio-temporal images $\mathcal{X} = \{X_1, X_2, \ldots, X_N\}$, $X_i \in \mathbb{R}^{H\times W\times T\times C}$, and ground-truth flows $\mathcal{Y} = \{F_1, F_2, \ldots F_N\}$, $F_i \in \mathbb{R}^{H\times W\times T-1\times 2}$, where H and W denote height and width, T is its sequence length, and C and N are the numbers of channels and data, respectively. The goal is to train a model f_θ that calculates flow $\hat{F}_i$ from the spatio-temporal sequence $X_i \in \mathcal{X}$. In a supervised case, we train a network by minimizing regression loss: $L_s = \frac{1}{N}\sum_i^N ||F_i - \hat{F}_i||_2^2$. In an unsupervised case, however, we cannot access $\mathcal{Y}$, but only $\mathcal{X}$. We thus configure an unsupervised loss term, L_p, with the photometric consistency assumption:

$$L_p = \frac{1}{N}\sum_i^N \sum_{(x,y,t)\in\Omega} \Psi(X_i(x,y,t) - X_i(x+u, y+v, t+1)), \tag{2.1}$$

where Ω contains all the spatio-temporal coordinates, $(u, v) = \hat{F}_i(x, y, t)$ is an estimated flow at (x, y, t), and Ψ is the robust penalty function [1]; $\Psi(x) = (|x|+\epsilon)^q$. Note that $X(x+u, y+v, \cdot)$ includes the warping operation using bilinear sampling [45], which supports back-propagation for end-to-end optimization. In this paper, we use the census transform in L_p with the same configuration used in [1] for all experiments unless otherwise stated.

Occlusion handling is performed by checking consistency [30] between forward and backward flows. The estimated occlusion mask is denoted with $\hat{C}_i^o(x, y, t)$, which is 1 if $\hat{F}_i(x, y, t)$ is not occluded, otherwise 0. This geometrically means the backward flow vector should be the inverse of the forward one if it is not occluded. Our loss terms use the occlusion map as previous work [30].

In this work, we use the data distillation loss [1] for occluded pixels:

$$L_d = \frac{1}{N}\sum_i^N \sum_{(x,y,t)\in\Omega} \Psi(\hat{F}_i^s(x,y,t) - \hat{F}_i^t(x,y,t))M_f(x,y,t), \tag{2.2}$$

where $\hat{F}_i^t$ is a flow from a teacher model, $\hat{F}_i^s$ is a flow from a student model, and M_f is a valid mask. In

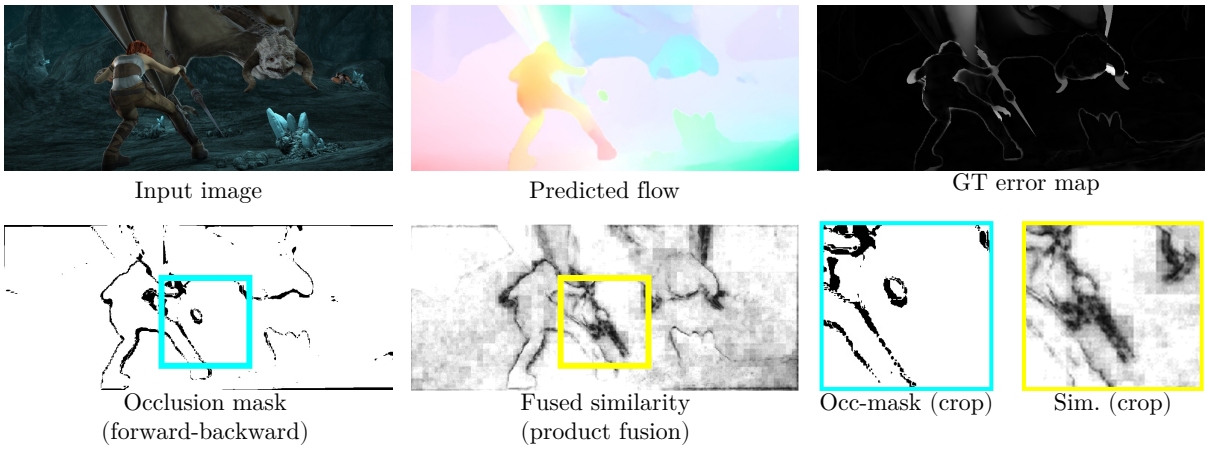


Figure 2.3: We visualize the occlusion mask ($\hat{C}^o$) and the similarity map (sim_f). By comparing the ground-truth error map and the fused similarity, we can observe that the similarity is low when the error is high. The bottom row shows occlusion mask and the similarity. Since the occlusion mask does not consider matching confidence, it does not represent how confident the matching is. On the other hand, our fused similarity marks whether the predicted flow is confident. Furthermore, we can use back-propagation since the similarity is differentiable.

short, the teacher model processes inputs without artificial synthetic occlusion, and the student model gets inputs with the generated occlusion. The student flow then learns from the teacher flow.

2.2.2 Feature Similarity from Multiple Layers

We use a model f_θ that estimates optical flow, where the model can be decomposed into an encoder $f^e_{\theta_e}$ and a decoder $f^d_{\theta_d}$, such that $f_\theta(X_i) = f^d_{\theta_d}(f^e_{\theta_e}(X_i))$; $\theta = \theta_e \cup \theta_d$. The encoder f^e is a function that outputs L-layered features $h_i = (h_i^1, h_i^2, \ldots, h_i^L)$; a lower numbered layer indicates a shallower layer in the deep network.

Given a matching $\hat{F}(x, y, t)$, we define the similarity between $X_i(x, y, t)$ and $X_i(x + u, y + v, t + 1)$ in a layer l to be:

$$\text{sim}_l(x, y, t; h_i^l, \hat{F}_i) = \frac{h_i^l(x, y, t) \cdot h_i^l(x + u, y + v, t + 1) + 1}{2}, \tag{2.3}$$

where $(u, v) = \hat{F}_i(x, y, t)$. Since we use l2-normalization for h_i^l, the function $\text{sim}(\cdot)$ is equivalent to the normalized cosine similarity between the reference feature and the target feature. Note that, for flows going out of the frame, no gradient is propagated during training. Additionally, we update encoder weights by indirect gradient, i.e., back-propagation through the decoder, while ignoring direct gradient from sim_l to h_i^l. That strategy is chosen because the direct gradient easily bypasses the decoder, and the encoder easily suffers from overfitting, in the end, the output flow is downgraded.

Fused similarity. To utilize features from all layers, we propose to fuse multi-layer feature similarities $(\text{sim}_1, \text{sim}_2, \ldots, \text{sim}_L)$. In CNNs, lower layer features tend to show high response for low-level details, e.g. edge, color, etc., while higher layers focus on objects [58]. As a result, similarity response of a lower layer usually bursts around the whole image, while similarity of a deeper layer has few modes. Therefore, to get stable, yet discriminative feature similarity response as shown in Fig. 2.1, we define the fused similarity

as product of multiple features:

$$\text{sim}_f(\cdot) = \prod_{l=1}^{L} \text{sim}_l(\cdot; h_i^l, \hat{F}_i). \tag{2.4}$$

For efficient calculation during training, we downsample the flow field $\hat{F}_i$ to the size of each layer feature using area interpolation before calculating the similarity; we use area interpolation, since it can propagate gradient to all source points, while other interpolation methods, e.g., bilinear interpolation, only update few nearest source points.

2.2.3 Learning Optical Flow with Feature Similarity

In this section, we propose three loss functions to effectively use the similarity map for optical flow estimation.

Feature separation loss. Given deep similarity, a model can learn flow by simply maximizing $\text{sim}(\cdots; h_i, \hat{F}_i)$, since larger similarity possibly means better matching solution. However, this simple approach in practice leads to worse results, because matchings between pixels under occlusion do not get better, even as we increase the similarity. In other words, maximizing the similarity for these points makes the flows incorrectly matched to random pixels with higher similarity.

To address this matching issue with uncertainty, we suppress flows with lower similarity by minimizing their similarity further, while refining flows with higher similarity by maximizing their similarity. First, we define a similarity threshold k, which we separate the values from:

$$k = \frac{1}{2}(k_{noc} + k_{occ}), \tag{2.5}$$

where k_{noc} and k_{occ} are average similarities of non-occluded pixels and occluded pixels: $k_{noc} = \frac{\sum_\Omega(\text{sim}_f \cdot \hat{C}_i)}{\sum_\Omega(\hat{C}_i)}$, $k_{occ} = \frac{\sum_\Omega(\text{sim}_f \cdot (1-\hat{C}_i))}{\sum_\Omega(1-\hat{C}_i)}$. Since occlusion is an effective criterion to set the boundary value, so that other kinds of difficulties can also be covered as shown in the experiments (Fig. 2.6).

We then formulate the feature separating loss term as:

$$L_f = \frac{1}{N} \sum_{i}^{N} \sum_{(x,y,t)\in\Omega} -(\text{sim}_f(x,y,t) - k)^2. \tag{2.6}$$

L_f is a quadratic loss function encouraging the similarity to be far from k, which serves as a boundary value that decides the direction of update. In other words, it suppresses uncertain flows, i.e. $\text{sim}_f(x,y) < k$, down towards 0, and certain flows, i.e. $\text{sim}_f(x,y) > k$, up towards 1. This can be also interpreted as minimizing entropy in semi-supervised learning [59]; we make a network output more informative by regularizing the similarity to be at each polar. A similar approach separating feature similarity from a different domain, image retrieval, has been shown to be effective [60].

One may concern that minimizing the similarity of uncertain flows, i.e., $\text{sim}_f < k$, can lead to an arbitrary matching solution. However, the product operation in the fused similarity (Eq. 2.4) makes sim_l with a higher similarity relatively retained, while changing smaller similarity much faster. Given any a, b s.t. $1 \le a, b \le L$, whose similarity is not 0, one can derive the following equation:

$$\frac{\partial L_f}{\partial \text{sim}_a(x,y,t)} = \frac{\text{sim}_b(x,y,t)}{\text{sim}_a(x,y,t)} \left(\frac{\partial L_f}{\partial \text{sim}_b(x,y,t)} \right). \tag{2.7}$$

That is, the scale difference between the two gradients is proportional to the fractional ratio of them, which can grow much faster when the denominator becomes smaller in the scale of the multiplicative

inverse. As a result, L_f is minimized by the smaller similarity approaching zero; higher layer similarities can be preserved to prevent arbitrary matching.

Regulated census loss. Since L_f (Eq. 2.6) is fully self-regulated, using only L_f for training can mislead the network itself. Thus, by modifying the well-known unsupervised loss (Eq. 2.1), we additionally use a regulated census loss, L_r, controlled by similarity:

$$L_r = \frac{1}{N}\sum_{i}^{N}\sum_{(x,y,t)\in\Omega} \Psi(\cdot)\hat{C}_i^o(x,y,t)\text{sim}_f(x,y,t). \tag{2.8}$$

The warping operation used in Eq. 2.1 is the bilinear sampling. This can only take into account four nearest pixels, making it difficult to address the pixel position far from the estimation, as also pointed by Wang et al. [46]. Therefore, the unsupervised loss (Eq. 2.1) does not give a correct direction when the current estimation is far from the desired target flow; whatever the loss is, it would be a noise in that case. In contrast, deep features have larger receptive fields with global context, so does the fused similarity. Thus, we can suppress the noise signal by multiplying the similarity; the similarity is designed to indicate whether the current estimation is near the desired target point.

Conditional smoothness loss. We use the smoothness prior for spatial locations with low similarity. In general, using the smoothness prior for all pixels degrades the accuracy because the flow-field is blurred. Meanwhile, if clear matching is not found, the smoothness constraint can help by being harmonized with surrounding flows. We thus define our smoothness prior loss considering similarity as:

$$L_s = \frac{1}{N}\sum_{i}^{N}\sum_{(x,y,t)\in\Omega} (|\nabla u|^2 + |\nabla v|^2)M_l(x,y,t), \tag{2.9}$$

$$M_l(x,y,t) = \begin{cases} 1, & \text{if } \text{sim}_f(\cdot) < k, \\ 0, & \text{otherwise.} \end{cases} \tag{2.10}$$

Loss for training. We jointly use the aforementioned losses to train our model with stochastic gradient descent. Our final loss function is sum of these loss functions:

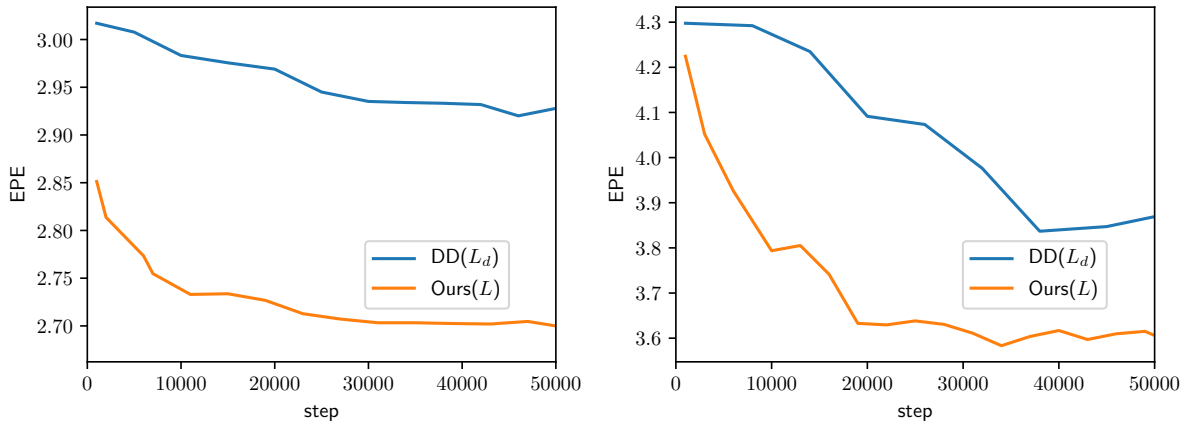
$$L = L_r + \lambda_f L_f + \lambda_s L_s + \lambda_d L_d, \tag{2.11}$$

where λs are weight parameters, and L_d is the data distillation loss defined in Eq. 2.2.

2.3 Experimental Results

Our network structure is based on PWC-Net [27], which is a deep network that contains warping, cost-volume, and context network to cover large displacements. We train the network from scratch using Adam optimizer [61] for stochastic gradient descent. In all experiment, we set the mini-batch size to 4.

Training procedure. Overall, we follow the training process of DDFlow [1] to initialize the network. To train the model, we first pretrain our network with FlyingChairs [24] and finetune it with each target dataset. For pretraining, we use the conventional photometric loss with RGB (Eq. 2.1) for 200k steps and additional 300k steps with occlusion handling. The resulting weights become the base network parameters for the following experiments. Next, using each target dataset, we finetune the base network. From this stage, we use the census transform for photometric loss. We train the model for 200k steps with occlusion handling. We then apply the final loss L (Eq. 2.11). We run first 1k steps without L_d, i.e. $\lambda_d = 0$. Then, the teacher network (Sec. 2.2.1) is fixed to use L_d and continues training using all the losses to 50k steps. We set hyper-parameters to $\lambda_s = 10^{-4}$ and $\lambda_f = 4$. We follow $\lambda_d = 1$ from the previous work [1].



(a) FlyingChairs

(b) Sintel

Figure 2.4: Training graphs of end-point-error (EPE) on two datasets. For DD(L_d), we use census, occlusion handling and L_d, which is the same setting to [1]. For ours(L), we use the full loss function L (Eq. 2.11). Ours performs consistently better during training

Data augmentation. For better generalization, we augment the training data using random cropping, random flipping, random channel swapping and color jittering; color jittering includes random brightness and saturation. We normalize the input RGB value into $[-0.5, 0.5]$.

FlyingChairs. The FlyingChairs [24] is a synthetic dataset created by combining chair and background images. The dataset consisting of 20 k pairs of images has a given train/test split, thus we use its training set for training our network and evaluate our model on its test set.

MPI Sintel. MPI Sintel [22] is a rendered dataset, originally from an open-source movie, Sintel. For training, we use 1k images from the training set and upload our results of the test set to its benchmark server for evaluation. We use both versions of rendering, i.e., clean and final, for training.

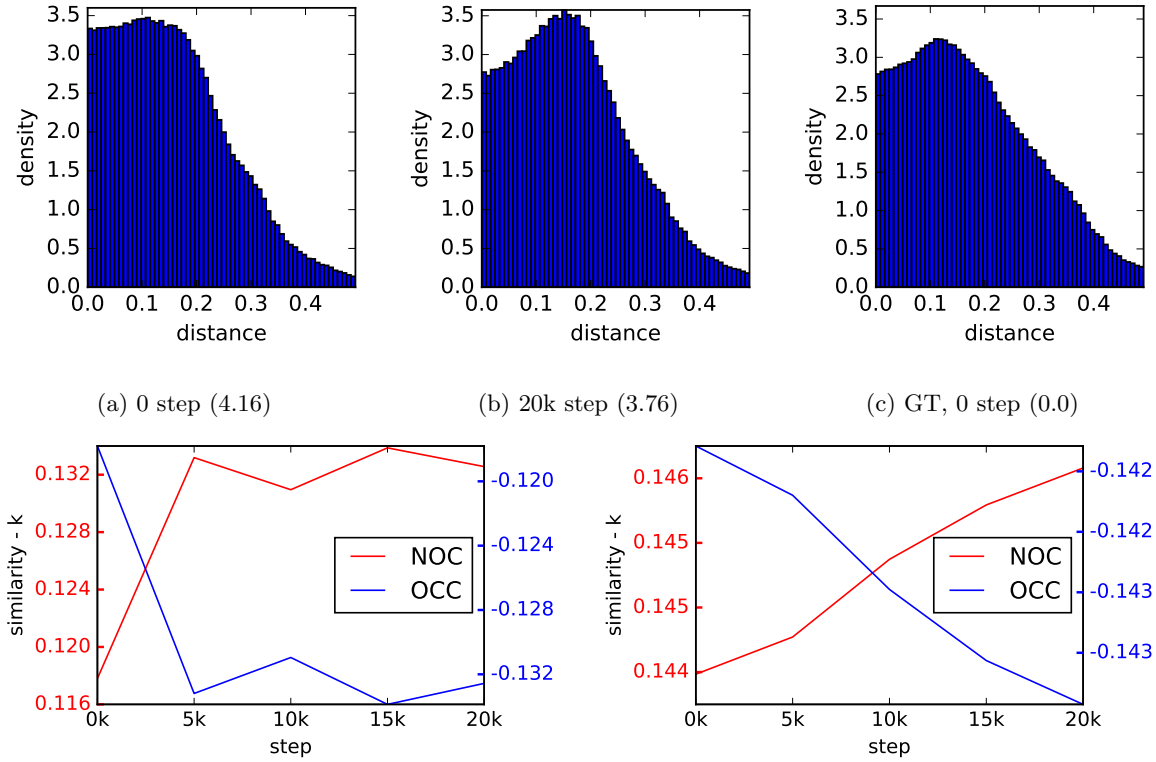
KITTI. KITTI [28] has a driving scene from the real world. This dataset has only 200 pairs of images with ground-truth flows. We thus train our model using unlabeled images from a multi-view extension set of KITTI without duplicated images in the benchmark training or testing sets, following the previous work [46].

2.3.1 Evaluation on Benchmarks

Ablation study. The results on benchmark datasets show that the network better learns to estimate flows with the feature similarity (Table 2.1). As can be seen in the last row of the table, our final method ($L_d + L_f + L_r$) works better in most cases than the other settings on both datasets. Due to low localization precision of deep features, however, using only L_f without L_r does not much improve the result. Interestingly, L_r that adaptively regulates the the conventional census loss is highly effective. When L_r is combined together with L_f, the combined loss ($L_f + L_r$) performs the best.

In Sintel where we report EPE on NOC and OCC, L_f improves better on OCC than on NOC. It implies that the suppression part in L_f takes effect, which discourages matching with higher similarity for uncertain flows. Table 2.2 shows that our method effectively covers various ranges of displacements.

We have also tested direct feature learning by the triplet loss [62], it did not improve the baseline accuracy DDFlow [1] i.e. $L_d + L_p$ alone in the experiments. The proposed losses and learning strategy are crucial to unsupervised learning of feature and optical flow.



(a) 0 step (4.16) (b) 20k step (3.76) (c) GT, 0 step (0.0)

(d) Estimated flows (e) GT flows

Figure 2.5: (a-c) We measure the distance from the boundary k (Eq. 2.5) to fused similarity on Sintel Final dataset. (a-c) show distance distributions of different steps; we show EPE inside parenthesis. During training, our feature separation loss pushes fused similarity away from the boundary k; note that the greater the distance is, the more separation we have. (d-e) show average $\text{sim}_f - k$ for occluded / non-occluded pixels. Note that occluded pixels show negative value. We measure the distance using ground-truth (GT) flows to observe the effect mainly on the encoder feature excluding the effect from the decoder side.

Loss parameter. We show results with different parameter settings in Table 2.3. Our smoothness constraint helps the network refine flows with lower similarity, when set to lower weight values; with a high weight value, the flow field becomes over-smoothed. With respect to λ_f, a weight greater than 4 can deteriorate the learning. As can be seen in Figure 2.4, our final loss (Eq. 2.11) converges well and it achieves higher performance than the baseline of using L_d.

Analysis on similarity. During training, our method gradually improves flow and encoder feature. Figure 2.5 illustrates the fused similarity with respect to each training step. Fig. 2.5a shows the distribution before we apply our feature separation loss. After 20k steps of training with our method, the distribution (Fig. 2.5b) becomes similar to ground-truth distribution (Fig. 2.5c). Figure 2.5d-2.5e plots average difference, i.e., $(\text{sim}_f - k)$, of each types of pixels, where we observe the feature similarity becomes more discriminative gradually; average similarities of non-occluded pixels and occluded pixels become higher and lower respectively. The last result using GT flows shows solely on the feature factor (encoder) without flow estimation factor (decoder), which confirms the updated, more discriminative features.

Qualitative results. Training with our feature similarity loss makes the flow and similarity much discriminative. During training, the similarity map changes in a way that higher similarity becomes

Table 2.1: Ablation study on various settings. Average end-point error (EPE) is used as a metric. For Sintel, the results are evaluated over all (ALL), non-occluded (NOC), and occluded (OCC) pixels. Results in parenthesis are achieved by testing the model using the data that the model is trained on. In left columns we show types of losses we use: occlusion handling ($\hat{C}$), data-distilation [1] (L_d), and ours: feature separation (L_f) and regulated census (L_r). The first two rows show the performance of our pretrained network trained with FlyingChairs

Base feature	$\hat{C}$	L_d	L_f	L_r	FlyingChairs	Sintel Clean			Sintel Final		
					ALL	ALL	NOC	OCC	ALL	NOC	OCC
RGB					4.01	5.61	3.01	38.64	6.44	3.76	40.45
RGB	✓				3.64	4.40	2.12	33.33	5.42	3.02	36.01
Census	✓				3.10	(3.22)	**(1.26)**	(28.14)	(4.37)	(2.25)	(31.25)
Census	✓	✓			2.93	(3.15)	(1.49)	(24.35)	(3.86)	(2.11)	(26.16)
Census	✓	✓	✓		2.87	(3.25)	(1.46)	(25.95)	(4.15)	(2.27)	(28.01)
Census	✓	✓		✓	2.81	(2.91)	(1.29)	(23.40)	(3.62)	(1.95)	(24.98)
Census	✓	✓	✓	✓	**2.69**	**(2.86)**	(1.28)	**(22.85)**	**(3.57)**	**(1.94)**	**(24.38)**

Table 2.2: Average EPE on different displacements. ALL: all pixels. *a*-*b*: pixels displaced within $[a, b)$

	Sintel clean				Sintel final			
	ALL	0-10	10-40	40+	ALL	0-10	10-40	40+
L_p	(3.22)	(0.59)	(3.86)	(20.45)	(4.37)	(0.83)	(5.41)	(27.17)
L_d	(3.15)	(0.59)	(4.02)	(19.51)	(3.86)	(0.71)	(5.06)	(23.63)
Ours	**(2.86)**	**(0.49)**	**(3.45)**	**(18.36)**	**(3.57)**	**(0.64)**	**(4.49)**	**(22.36)**

higher and lower similarity goes lower. As a result, the network can improve its prediction by reinforcing clear matching and suppressing uncertain matching. In Figure 2.6, we visualize how the similarity loss can be beneficial to flow learning. In the examples, our loss effectively improves the flow estimation by pushing similarity to each polar; that is why our method performs well in uncertain regions. In Figure 2.7, we compare ours with SelFlow [5]. In the first two examples, the uncertain snow regions are the most challenging part, so that the state-of-the-art SelFlow fails in such regions. On the other hand, our method is able to handle such regions effectively, since the similarity loss suppresses the flows in that regions.

Quantitative comparison to state-of-the-art. We compare our method with existing deep unsupervised methods in Table 2.4. Our method effectively improves the baseline framework [1] and shows competitive results against other unsupervised methods. For Sintel, SelFlow gets a better result in the Sintel Final testset; it is trained on 10 k additional Sintel movie frames, while our model uses 1 k frames from the MPI Sintel dataset. Since our method can be used jointly with hallucinated occlusion and multiple-frame schemes from SelFlow [5], we expect a much stronger unsupervised model if the two are combined. In the real dataset KITTI, our method effectively improves over our baseline model (DDFlow), and reduces the percentage of erroneous pixels to 13.38% in the test benchmark. Overall, our approach achieves top-1 or top-2 consistently across different benchmarks. This demonstrates the robustness of our

Table 2.3: Average EPE depending on loss weighting parameters

(a) Smoothness weight

λ_s	10^{-2}	10^{-3}	10^{-4}	10^{-5}
FlyingChairs	3.27	2.84	2.85	**2.83**
Sintel Final	(4.41)	(4.34)	**(4.16)**	(4.41)

(b) Separation weight

λ_f	2.0	3.0	4.0	5.0
FlyingChairs	2.95	2.90	**2.85**	3.21
Sintel Final	(4.33)	(4.31)	**(4.16)**	(4.36)

Table 2.4: Comparison to state-of-the-art deep unsupervised optical flow methods. Results in parentheses indicates it is evaluated using data it is trained on. We report average end-point-error for most categories and percentage of erroneous pixels for KITTI testset calculated from benchmark server. Best results in **red** and second best results in blue.

Method	Chairs	Sintel Clean		Sintel Final		KITTI 2015	
	test	train	test	train	test	train	test(Fl)
BackToBasic [29]	5.3	-	-	-	-	-	-
DSTFlow-ft [39]	5.52	(6.16)	10.41	(6.81)	11.27	16.79	39%
OccAwareFlow-best [46]	3.30	(4.03)	7.95	(5.95)	9.15	8.88	31.2%
UnFlow-CSS-ft [30]	-	-	-	(7.91)	10.22	8.10	23.30%
MultiFrameOccFlow-ft [47]	-	(3.89)	7.23	(5.52)	8.81	6.59	22.94%
DDFlow-ft [1]	2.97	(2.92)	6.18	(3.98)	7.40	5.72	14.29%
SelFlow-ft-Sintel [5]	-	(2.88)†	6.56†	(3.87)†	**6.57**†	**4.84**	14.19%
Ours-Chairs	**2.69**	3.66	-	4.67	-	16.99	-
Ours-ft-Sintel	3.01	**(2.86)**	**5.92**	**(3.57)**	6.92	12.75	-
Ours-ft-KITTI	4.32	5.49	-	7.24	-	5.19	**13.38%**

†: pretrained on the original Sintel movie

approach and benefits of utilizing deep self-supervised features and fused similarity.

Failure cases. Most unsupervised methods tend to estimate motion in a larger area than it really is, and it occurs more frequently for smaller and faster objects. In contrast, since our method estimates the similarity of a matching and refines it with the similarity, it sometimes reduces flows for small and fast-moving objects. In the last example in Figure 2.7, ours fails to catch the movement of few birds flying fast over the stairs.

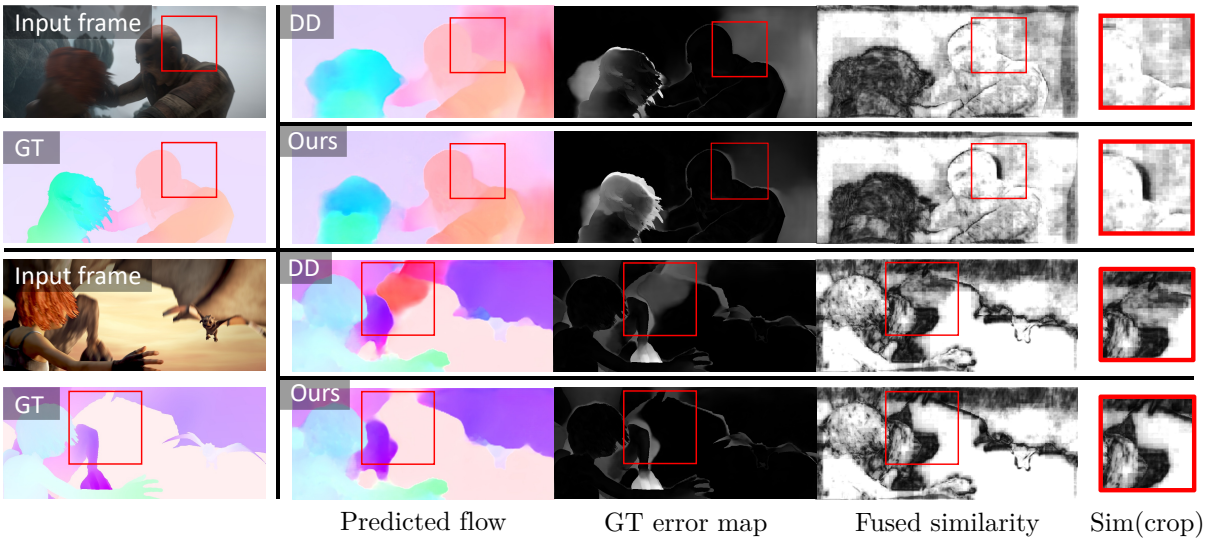


Figure 2.6: Fused similarity of models trained w/o and w/ the feature similarity loss, denoted by **DD** and **ours**, respectively. The similarity loss suppresses similarity of the background snow texture behind the fighting man, resulting in the similarity map in the second row. In the second example, the sky region is expanded since the simlilarity increases by the similarity loss. As a result, the flow field effectively separates the brown wing of the dragon and the sky in brown

Figure 2.7: Comparison to SelFlow [5] on the Sintel testset. We retrieve the resulting images and the visualizations of ground-truth from its benchmark website [6]; it provides only twelve test samples for each method

Chapter 3. Semi-Supervised Learning of Optical Flow

Optical flow describes the pixel-level displacement in two images, and is a fundamental step for various motion understanding tasks in computer vision. Recently, supervised deep learning methods have shown remarkable performance in terms of overcoming challenges – such as motion blur, change of brightness and color, deformation, and occlusion – and predicting more accurate flows. The key to success is end-to-end learning from large-scale data. For optical flow learning, large-scale datasets have been released [24, 25, 22] and deep architectures have been advanced [24, 27, 8].

Building a good optical flow model on a target dataset is critical; most training data are synthetic, and it requires tremendous effort to label random video frames by pixel-wise dense correspondences. Thus, to obtain a good model on a target dataset, synthetic training set generation [2, 63] and GAN-based adaptation [64, 65] have been studied. In addition, unsupervised loss functions [30, 31] – used without ground truth – have been proposed. However, generating a synthetic dataset for a target domain is often computationally expensive or confined to a specific domain. Moreover, unsupervised losses do not reach the state-of-the-arts, compared to supervised methods. Therefore, there have been needs for a simpler, general, and high-performance method to build a better model on a target dataset.

In this paper, we propose a fine tuning strategy for semi-supervised optical flow learning, which helps to build a better model on unlabeled or partly-labeled target datasets. Fig. 3.1 demonstrates the concept of our approach. Our method is a fine tuning method, where the pretrained network is adapted to the unlabeled target data. In the fine tuning stage, we use a labeled dataset with an unlabeled target dataset, further reducing errors on the target dataset.

To build our method, we investigate unsupervised and self-supervised approaches, where a network learns optical flow by unlabeled samples. However, the existing unsupervised loss does not show better performance in the fine tuning stage (Fig. 3.1). Moreover, self-supervision methods tend to show highly unstable behavior or lower performance in the fine tuning stage. To address the issue, we propose our flow supervisor with two strategies: the parameter separation and passing student outputs, which are effective for higher performance and stable semi-supervised learning. As shown in Fig. 3.1, our fine tuning method clearly reduces the error of the pretrained model, even without a label of the target dataset.

To summarize, we propose a semi-supervised fine tuning strategy to improve an optical flow network on a target dataset, which has not been explored extensively. Our strategy is distinguished by the flow supervisor module, designed with the parameter separation and passing student outputs. We show the effectiveness of our method by comparing it with alternative self-supervision methods, and confirm that our approach stabilizes the learning process and results in better accuracy. In addition, we test our method on Sintel and KITTI benchmarks and achieve meaningful improvements over the state-of-the-arts by exploiting additional unlabeled data.

3.1 Related Work

Supervised optical flow has been studied with the development of datasets for optical flow learning [24, 25, 2] and the advances of deep network architectures [24, 27, 8]. Due to the high annotation cost and label ambiguity in a raw video, synthetic datasets have been made, where optical flow fields are generated together with images [63, 24, 25, 2]. Along with the datasets, network architectures for optical flow have

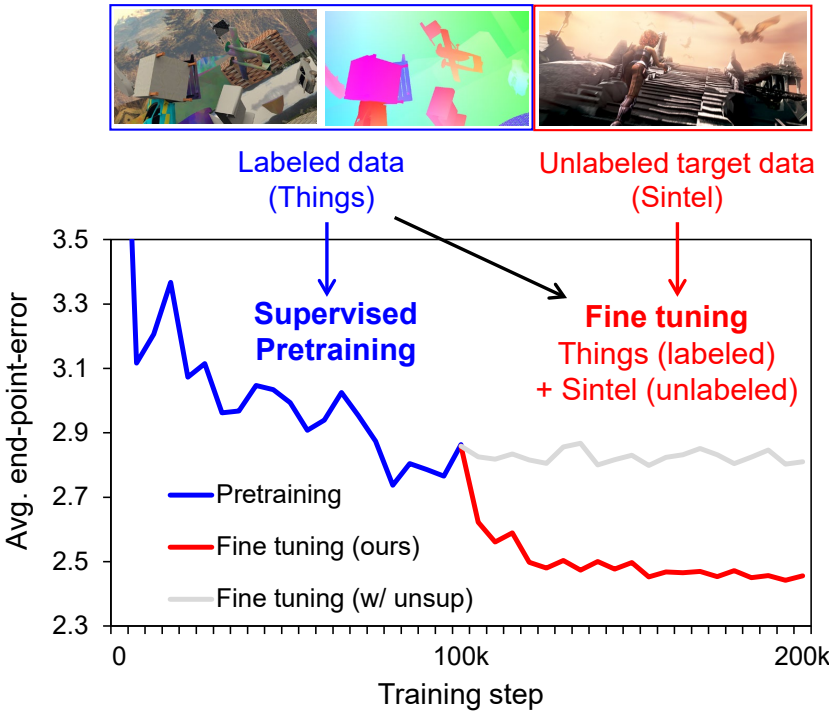


Figure 3.1: **End-point-error on Sintel**. Our method is used to adapt a pretrained model to a target domain without a target domain label; it is designed to overcome an unstable convergence and low accuracy in traditional methods. For instance, our method outperforms the unsupervised loss (Eq. 3.2) in fine tuning, which makes our method favorably better

been significantly improved by the cost-volume [24, 27, 66], warping [25, 27], and refinement scheme [44, 8]. Although generalization has been improved thanks to the synthetic datasets and the network architectures, it is still difficult to achieve better performance while being blind to a target domain [67].

Unsupervised optical flow is another stream of optical flow research, where optical flow is learned without an expensive labeling or label generation process [30, 68, 69, 31]. With the advanced deep architectures from supervised optical flow, unsupervised optical flow studies have focused on designing loss functions. Previous work has mainly focused on the fully unsupervised setting, while we explore semi-supervised optical flow where labeled data is accessible in addition to unlabeled target domain data.

Semi-supervised optical flow has been studied to utilize unlabeled target domain data with existing labeled synthetic data [64]. The experimental setting – similar to unsupervised domain adaptation [70] – is designed since it is relatively easier to exploit synthetic training datasets than annotating images of a target dataset. A simple baseline method for a target domain would be using a traditional unsupervised loss [30] and supervised learning, which, unfortunately, has shown inferior performance than the supervision-only training [64, 67] (Table 3.1). Thus, reducing the domain gap [64] and stabilizing unsupervised loss gradients [67] have been proposed.

In this work, we introduce the flow supervisor, which consists of the separate parameters and the student output connection. We found our design scheme is superior to the baseline designs in terms of training stability and performance in the semi-supervised setting.

Knowledge distillation and self-supervision in neural networks are proposed to train a network under the guidance of a teacher network [71, 72] or itself [73, 74]. Interestingly, the technique can build a better student network even when the same network architecture is used for both student and teacher,

i.e., self-distillation [71, 72]. In the optical flow field, knowledge distillation and self-supervision have been studied actively in the context of unsupervised optical flow [75, 5, 31]. Generally, these methods can be interpreted as learning using privileged information [76], in that a student usually sees a limited view, e.g., cropped images, while a teacher is given a privileged view, e.g., full images. In this work, we investigate the effectiveness of self-supervision in terms of semi-supervised optical flow, which has not been explored in previous work. In addition, we found that the traditional self-supervision method for optical flow [31] tends to make a loss diverge in the semi-supervised setting.

Thus, we propose a novel self-supervision method to ameliorate the unstable convergence; we found that the parameter separation of a teacher model and passing student output are the key components to make the training stable. By applying our method, a network can successfully exploit the unlabeled target data in the semi-supervised setting. In addition, we show that our method can address the lack of a target labeled dataset, e.g., 200 labeled pairs for KITTI, by the ability to utilize unlabeled samples.

3.2 Approach

3.2.1 Preliminaries on Deep Optical Flow

Deep optical flow estimation is defined with an optical flow estimator $f_\theta(\mathbf{x}_1, \mathbf{x}_2)$, which predicts optical flow $\hat{\mathbf{y}} \in \mathbb{R}^{H \times W \times 2}$ from two images $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^{H \times W \times C}$, where H is height, W is width, and C is channel.

In supervised learning, we train f_θ by minimizing the supervised loss:

$$\mathcal{L}_{\text{sup}}(\theta) = \mathbb{E}_{(\mathbf{x}_1, \mathbf{x}_2, \mathbf{y}) \sim p_s}[\ell_{\text{sup}}(f_\theta(\mathbf{x}_1, \mathbf{x}_2), \mathbf{y})], \tag{3.1}$$

where p_s is the labeled data distribution, $\mathbf{y}$ is the ground truth optical flow, and ℓ_{sup} is L_1, L_2 [27] or the generalized Charbonnier loss [30].

On the other hand, an unsupervised method defines a loss function with a differentiable target function $\ell_{\text{unsup}}(\cdot)$ which can be computed without a ground truth $\mathbf{y}$, resulting in the unsupervised loss:

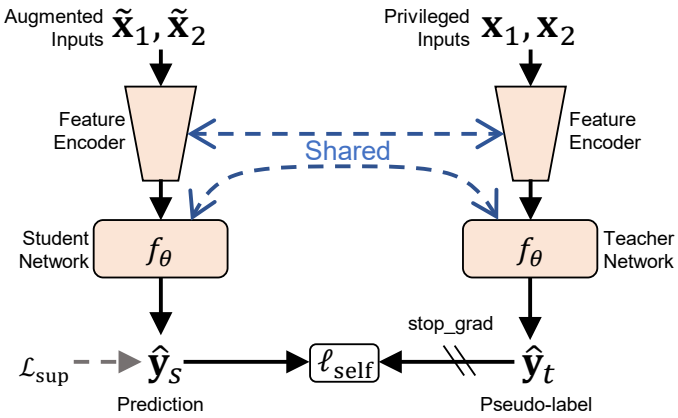
$$\mathcal{L}_{\text{unsup}}(\theta) = \mathbb{E}_{(\mathbf{x}_1, \mathbf{x}_2) \sim p_u}[\ell_{\text{unsup}}(f_\theta(\mathbf{x}_1, \mathbf{x}_2), \mathbf{x}_1, \mathbf{x}_2)], \tag{3.2}$$

where p_u is an unlabeled data distribution. Most commonly, ℓ_{unsup} is defined with the photometric loss $\ell_{\text{photo}} = \rho(\text{warp}(\mathbf{x}_2, \hat{\mathbf{y}}) - \mathbf{x}_1)$ where ρ is the Charbonnier loss [30] and $\text{warp}(\cdot)$ is the differentiable backward warping operation [45].

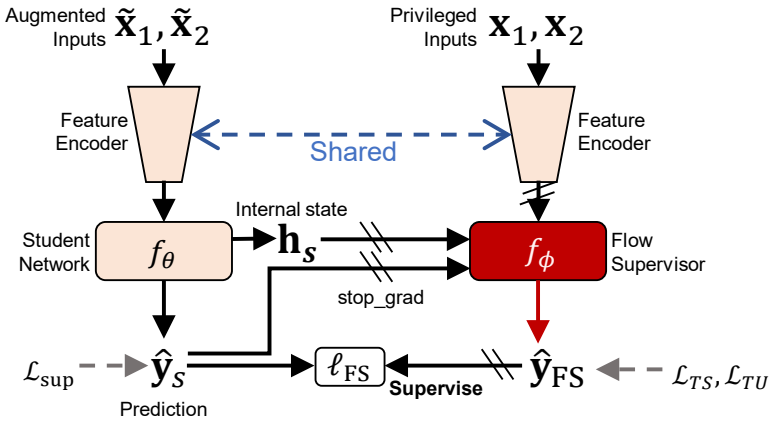
3.2.2 Problem Definition and Background

We train our model on labeled data and unlabeled data, which is similar to experimental settings that appear in [64, 67]. For instance, FlyingThings3D (rendered scene) and KITTI (driving scene) can be considered as labeled and unlabeled datasets, respectively. We aim to build a high-performance model on a target domain, with only a labeled synthetic dataset and an unlabeled target dataset. This is a practical scenario since a synthetic dataset is relatively inexpensive and publicly available, whereas specific target domain data is rarely annotated.

We focus on designing a self-supervision method with a stable convergence and better accuracy, since it is not trivial to adopt unsupervised losses and self-supervision for semi-supervised learning. First, unsupervised loss functions often lead a network to a worse local minimum when it is naively fine tuned from supervised (pretrained) weights. Second, traditional self-supervision strategies for semi-supervised



(a) Self-supervision



(b) Flow supervisor

Figure 3.2: **(a) Self-supervision** for optical flow is configured with a teacher network which is given privileged images as an input, i.e., full images before cropping. **(b) Flow supervisor** reviews the student flow $\hat{\mathbf{y}}_s$ and outputs the pseudo-label $\hat{\mathbf{y}}_{\text{FS}}$ to supervise the student without ground truth flows. We use the separate flow supervisor with parameter ϕ, which improves the stability and accuracy

learning do not converge well. Thus, we propose a simple and effective practice for semi-supervised learning, where we utilize synthetic datasets for better performance on a target dataset. By our strategy, a network can successfully exploit the unlabeled target data for better fine tuning performance. Detailed method is discussed in the next section.

3.2.3 Flow Supervisor

Our design for semi-supervised optical flow learning is based on self-supervised learning where a student network learns from a pseudo-label predicted by a teacher network. This concept has been explored in the optical flow field in terms of unsupervised learning, which is not directly applicable in the semi-supervised case due to unstable convergence.

Thus, we introduce two distinctive design schemes compared to the existing self-supervision techniques for optical flow, depicted in Fig. 3.2. First, we introduce a supervisor parameter ϕ, distinguished from the student parameter θ, which learns to supervise the student network. Second, we add a connection from

the student network to the supervisor network, which enables the supervisor network to learn conditional knowledge, i.e., $P(\mathbf{y}|\hat{\mathbf{y}}_s)$, instead of predicting from scratch.

We define the student network f_θ and the flow supervisor network f_ϕ, as shown in Fig. 3.2b. The student network f_θ includes a feature encoder and a flow decoder; for simplicity, we abstract the feature encoder and decoder parameters with θ. The flow decoder has the internal feature $\mathbf{h}_s$, and outputs the predicted flow $\hat{\mathbf{y}}_s$. The student network is the optical flow network used for inference, whereas the supervisor network is only used for training; this results in no additional computational cost for the testing time. In training time, we use the flow supervisor (FS) loss function $\mathcal{L}_{\text{FS}}$ to supervise the student flow $\hat{\mathbf{y}}_s$ with the teacher flow $\hat{\mathbf{y}}_{FS}$:

$$\mathcal{L}_{\text{FS}}(\theta) = \mathbb{E}_{d_s \sim p_s, d_u \sim p_u}[\ell_{\text{sup}}(d_s) + \alpha \ell_{\text{FS}}(d_u)], \tag{3.3}$$

where $\ell_{\text{FS}}(\cdot) = \rho(\hat{\mathbf{y}}_{FS} - \hat{\mathbf{y}}_s)$, α is a hyper-parameter weight and (d_s, d_u) are sampled data from (p_s, p_u). We use loss function $\mathcal{L}_{\text{FS}}$ for the student network to learn from both labeled and unlabeled data.

Separate parameters. We empirically observed that the plain self-supervision (Fig. 3.2a) leads to divergence in semi-supervised optical flow learning. This undesirable behavior is also observed in siamese self-supervised learning, where preventing undesirable equilibria is important [74, 73]. As our solution, we have the separate module, i.e., the flow supervisor to prevent the unstable learning behavior. Our flow supervisor is related to the predictor module in self-supervised learning work [73], which also prevents the unstable training. We also compare our design with the mean-teacher [77, 74] with exponential moving average (EMA), and a fixed teacher [78] in Fig. 3.3, since these designs have been widely adopted in semi-supervised learning.

Passing student outputs. We design our supervisor model to have input nodes for student outputs. Thus, the teacher output flow is conditioned by the student output. Specifically, our teacher model includes a residual function $\Delta f_\phi(\cdot)$, such that $f_\phi(\mathbf{x}, \hat{\mathbf{y}}_s, \mathbf{h}_s) = \Delta f_\phi(\cdot) + \hat{\mathbf{y}}_s$. We realize this concept with the residual flow decoder in the RAFT [8] architecture.

In residual learning, it has been believed and shown that learning a residual function $\Delta f(\mathbf{x}) = f(\mathbf{x}) - \mathbf{x}$, instead of the original function $f(\mathbf{x})$, is better for deeper inference [79] or domain discrepancy modeling [80]. With residual teacher function Δf_ϕ, the flow supervisor loss is reformulated as:

$$\ell_{\text{FS}} = \rho(\hat{\mathbf{y}}_{FS} - \hat{\mathbf{y}}_s) = \rho(f_\phi(\hat{\mathbf{y}}_s) - \hat{\mathbf{y}}_s) = \rho(\Delta f_\phi(\hat{\mathbf{y}}_s)). \tag{3.4}$$

Relation of $f_\phi(\cdot)$ to a meta learner. Meta-learning [81, 82] defines how to learn by learning an update rule $\Delta\theta^t$ within the parameter update $\theta^{t+1} \leftarrow \theta^t + \Delta\theta^t$. The residual function $\Delta f_\phi(\ldots, \hat{\mathbf{y}}_s)$ is regarded as a meta learner predicting update rule $\Delta\theta^t$, conditioned by student predictions. Assuming ρ is the square function, learning by $\mathcal{L}_{\text{FS}}$ is equivalent to using the update rule $\Delta\theta^t = -2\frac{\partial \hat{\mathbf{y}}_s}{\partial \theta^t}^T \cdot \Delta f_\phi(\hat{\mathbf{y}}_s)$; where ϕ is the parameter of our flow supervisor. That is, the learning rule is learned by the supervisor parameter ϕ to supervise the student parameter θ.

3.2.4 Supervision

Learning supervisor parameters ϕ is important for supervising the student network. Basically, the supervisor learns to maximize the likelihood, e.g., $\log P(\mathbf{y} - \hat{\mathbf{y}}_s | \mathbf{x}_1, \mathbf{x}_2, \hat{\mathbf{y}}_s, \phi)$, where the student output $\hat{\mathbf{y}}_s$ is inferred from an augmented input $\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2$. First, it is natural to give the conditional knowledge from

the labeled data p_s. For this purpose, we minimize $\mathcal{L}_{\text{TS}}$ – which stands for supervised teacher loss – to train the supervisor:

$$\mathcal{L}_{\text{TS}}(\phi) = \mathbb{E}_{(\mathbf{x}_1,\mathbf{x}_2,\mathbf{y})\sim p_s}[\ell_{\text{sup}}(f_\phi(\mathbf{x}_1,\mathbf{x}_2,f_\theta(\tilde{\mathbf{x}}_1,\tilde{\mathbf{x}}_2)),\mathbf{y})]. \tag{3.5}$$

In addition to using the labeled data, we found that if a labeled dataset and an unlabeled dataset are distant, e.g., Things $\leftrightarrow$ KITTI, using the unsupervised loss is especially effective. The unsupervised teacher loss $\mathcal{L}_{\text{TU}}$ on unlabeled data p_u is defined by:

$$\mathcal{L}_{\text{TU}}(\phi) = \mathbb{E}_{(\mathbf{x}_1,\mathbf{x}_2)\sim p_u}[\ell_{\text{unsup}}(f_\phi(\mathbf{x}_1,\mathbf{x}_2,\hat{\mathbf{y}}_s),\mathbf{x}_1,\mathbf{x}_2)]. \tag{3.6}$$

In the pretraining stage, we train the student model from scratch using the supervised loss $\mathcal{L}_{\text{sup}}$, resulting in a pretrained weight θ. In the fine tuning stage, we initialize ϕ with θ and jointly optimize θ and ϕ on labeled and unlabeled datasets. Formally, we use stochastic gradient descent to minimize

$$\mathcal{L}(\theta,\phi) = \mathcal{L}_{\text{FS}}(\theta) + \lambda_{\text{TS}}\mathcal{L}_{\text{TS}}(\phi) + \lambda_{\text{TU}}\mathcal{L}_{\text{TU}}(\phi), \tag{3.7}$$

with hyper-parameter loss weights λ_{TS} and λ_{TU}.

3.3 Experiments

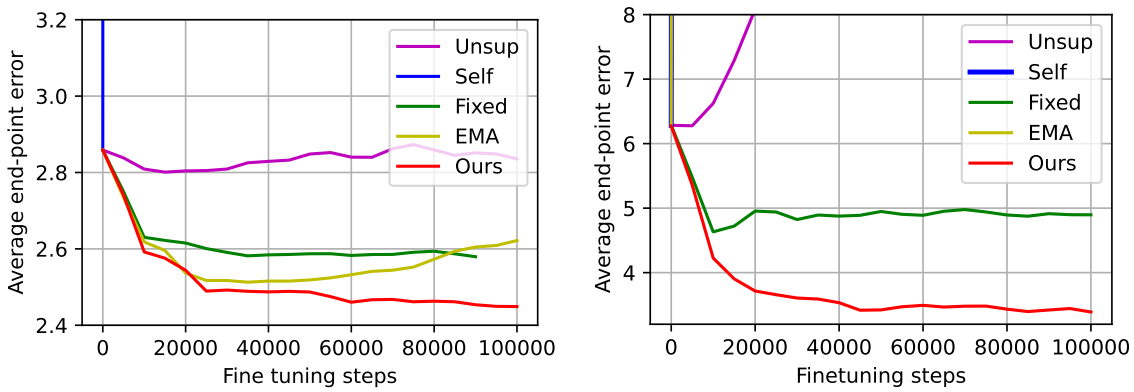
3.3.1 Experimental setup

Pretraining. Our pretraining stage follows the original RAFT [8]. We pretrain our student network with FlyingChairs [24] and FlyingThings3D [25] with random cropping, scaling, color jittering, and block erasing. The pretraining scheme includes 100k steps for FlyingChairs and additional 100k steps for FlyingThings3D, with the same learning rate schedule of RAFT, on which we base our network. For the supervised loss (Eq. 3.1), we use L_1 loss.

Flow supervisor. The flow supervisor is implemented with the GRU update module of RAFT, which performs an iterative refinement process with the output flow of the previous step. At the start of the semi-supervised training phase, we initialize the supervisor model with the pretrained weights of the GRU update module. To match the student prediction from cropped inputs to the supervisor module with full resolution, i.e., privileged, we pad the student predictions with zero. In a training and testing phase, we use 12 iterations for both the student and supervisor GRUs.

Semi-supervised dataset. To compare semi-supervised learning performance on Sintel [22] and KITTI [23], we follow the protocol [69], which utilizes the unlabeled portion, i.e., testing set, of each dataset for training; the difference is that we use a labeled dataset, e.g., FlyingThings3D, into training. Specifically, we use the unlabeled portion of Sintel for fine tuning on Sintel and unlabeled KITTI multiview dataset for fine tuning on KITTI; then the labeled splits are used for evaluation.

Optimization. We initialize our student and supervisor models with the pretrained weights, then minimize the joint loss (Eq. 3.7) for 100k steps. We use $\alpha = 1.0$, $\lambda_{\text{TS}} = 1.0$ and $\lambda_{\text{TU}} = 0$ by default and $\lambda_{\text{TU}} = 0.01$ for the Things + KITTI setting, unless otherwise stated. Detailed optimization settings and hyper-parameters are provided in the supplementary material and the code.



(a) Sintel Final

(b) KITTI

Figure 3.3: **Plots comparing finetuning stage**. We use FlyingThings3D as labeled data, and each target dataset as an unlabled data. The EPEs are measured on unseen portion of data

Method	Sintel		KITTI	
	Clean	Final	EPE	Fl-all (%)
Sup	1.46	2.80	5.79	18.79
Sup + Unsup	1.47	2.73	9.21	16.95
Sup + Self	diverged		diverged	
Sup + EMA	1.40	2.63	diverged	
Sup + Fixed	1.32	2.58	4.91	15.92
Sup + FS (Ours)	**1.30**	**2.46**	**3.35**	**11.12**

Table 3.1: **Comparison to semi-supervised baselines.** Semi-supervised baselines and ours are compared, as well as the supervised loss only (Sup) result. We use widely-used metrics in optical flow: end-point error (EPE) and ratio of erroneous pixels (Fl)

3.3.2 Empirical Study

Comparison to Semi-Supervised Baselines.

In Table 3.1 and Fig. 3.3, we perform an empirical study, comparing ours with several baselines for semi-supervised learning. In the experiments, we use FlyingThings3D as the labeled data p_s and each target dataset as the unlabeled data p_u.

Unsupervised loss. An unsupervised loss is designed to learn optical flow without labels. Exploring the unsupervised loss is a good start for the research since we could expect a meaningful supervision signal from unlabeled images.

In the second row of Table 3.1, we report results by the unsupervised loss. We use the loss function in Eq. 3.2, which includes the census transform, full-image warping, smoothness prior, and occlusion handling as used in the prior work [31]. Unfortunately, the unsupervised loss function does not give a meaningful supervision signal when jointly used with the supervised loss, resulting in a degenerated EPE on KITTI (5.79 $\rightarrow$ 9.21). Interestingly, applying the identical loss to our supervisor, i.e., $\mathcal{L}_{\text{TU}}$, results in a better EPE on KITTI (5.79 $\rightarrow$ 3.35), see also Table 3.3.

(a) **Comparison to SemiFlowGAN [64]**

Data	Method	KITTI	
		EPE	Fl-all (%)
C	SFGAN [64]	17.19	40.82
C+K	SFGAN [64]	16.02 (-6.8%)	38.77 (-5.0%)
C+T	RAFT	5.79	18.79
T+K	RAFT + FS	**3.35 (-42.1%)**	**11.12 (-40.8%)**

(b) **Comparison to AutoFlow [2]**

Method	Sintel		KITTI	
	Clean	Final	EPE	Fl-all (%)
RAFT† (C+T) [2]	1.68	2.80	5.92	-
RAFT (C+T)	1.46	2.80	5.79	18.79
RAFT† (A) [2]	1.95	2.57	4.23	-
RAFT + FS (Ours)	**1.30**	**2.46**	**3.35**	**11.12**

† model implemented by [2]

Table 3.2: **Comparison to optical flow approaches.** We report a percentage of improvement over each baseline in the parentheses. We mark used datasets: FlyingChairs (C), FlyingThings (T), unlabeled KITTI (K), and AutoFlow (A) [2]

Self- and teacher-supervision. We summarize the results of the self-supervision and teacher-based methods in Table 3.1 and Fig. 3.3. The baseline models include the plain self-supervision (Self), the fixed teacher (Fixed), and the EMA teacher (EMA). In the plain self-supervision, we use the plain siamese networks (Fig. 3.2a). The fixed teacher [78] and the EMA teacher [77] are inspired by the existing literature. The self-supervision loss used in the experiments is defined by:

$$\ell_{\text{self}} = \rho[f_\theta(\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2) - \texttt{stop\_grad}(f_t(\mathbf{x}_1, \mathbf{x}_2))], \tag{3.8}$$

where f_t is the teacher network of each baseline: $t = \theta$ for plain self-supervision, $t = \text{EMA}(\theta)$ for mean teacher, and $t = \theta_{\text{pretrained}}$ for fixed teacher. The plain self-supervision (Self) quickly diverges during the early fine tuning step for both Sintel and KITTI. In the EMA teacher (EMA) results, more stable convergence is observed for Sintel, not KITTI. We speculate that this unstable convergence is caused by the domain gap between the unlabeled data and the labeled data; there exists a wider domain difference between FlyingThings3D (p_s) and KITTI (p_u) than the difference between FlyingThings3D (p_s) and Sintel (p_u), since KITTI is a real-world dataset, while Things and Sintel are both three-dimensional rendered datasets. When the fixed teacher (Fixed) is used, we can observe more stable learning. Our method (FS), enabling the supervisor learning along with the student, shows superior semi-supervised performance on both datasets than the EMA and the fixed teacher. We further analyze our method in the following sections.

Comparison to SemiFlowGAN [64].

We compare our method with the existing semi-supervised optical flow method [64] in Table 3.2a. SemiFlowGAN uses a domain adaptation-like approach by matching distributions of the error maps

Experiment		Sintel		KITTI	
		Clean	Final	EPE	Fl-all
Parameter separation	on	**1.30**	**2.46**	**3.35**	**11.12**
	off	diverged		diverged	
Passing student output	full	**1.30**	**2.46**	**3.35**	**11.12**
	w/o res	1.34	2.50	6.16	12.34
	off	**1.30**	**2.46**	6.68	13.36
Shared encoder	on	**1.30**	**2.46**	**3.35**	**11.12**
	off	1.33	2.58	3.80	12.03
Teacher loss type	TS	**1.30**	**2.46**	4.69	14.48
	+TU	1.55	2.80	**3.35**	**11.12**
Teacher input	clean	**1.30**	**2.46**	**3.35**	**11.12**
	aug	1.33	2.56	4.17	11.55

Table 3.3: **Ablation experiments.** We underline the final settings

from each domain. Compared to SemiFlowGAN, our approach gives more direct supervision to optical flow predictions from the supervisor model. Here, we evaluate how much a supervised-only baseline, i.e., trained w/o KITTI, is able to be improved by exploiting the unlabeled target dataset, i.e. traind w/ KITTI. We can clearly observe much larger performance improvement (-6.8% vs. -42.1%) when our method is used.

Comparison to AutoFlow [2].

In Table 3.2b, we compare ours with AutoFlow, where our method shows better EPEs on both Sintel and KITTI. AutoFlow is devised to train a better neural network on target datasets by learning to generate data, as opposed to ours using semi-supervised learning. For instance, the AutoFlow dataset, whose generator is optimized on Sintel dataset shows superiority over a traditional synthetic dataset, e.g., FlyingThings, in generalization on unseen target domains. On the other hand, our strategy is to utilize a synthetic dataset and an unlabeled target domain dataset by the supervision of the flow supervisor; it boosts performance on the target domain without labels.

Ablation Study.

In Table 3.3, we compare ours with several alternatives.

Parameter separation. Self-supervision with the shared network (see Self in Fig. 3.3) is unsuitable for semi-supervised optical flow learning. On the other hand, having the separate parameters for the supervisor network effectively prevents the divergence. This separation can be viewed as the predictor strategy in the self-supervised learning context [73]. We also analyze the separate network as a meta-learner, which learns knowledge specifically for supervision. We have shortly discussed this aspect in Sec. 3.2.3.

Passing student outputs ($\hat{\mathbf{y}}_s, \mathbf{h}_s$). Our architecture passes the student output flow and internal state to the supervisor network to enable the supervisor to learn the residual function, as described in Eq. 3.4.

Table 3.4: **KITTI results of models trained on VKITTI [3]**

	Supervised		Semi-supervised
Metric	RAFT	SepFlow [83]	RAFT+FS (ours)
EPE	3.64	2.60	**2.39**
Fl-all (%)	8.78	7.74	**7.63**

In row 5 in Table 3.3, we show the results when we do not pass student outputs $(\hat{\mathbf{y}}_s, \mathbf{h}_s)$ to the teacher. Interestingly, the Sintel results remain the same even without passing student outputs. while the KITTI results benefit from passing student outputs. We ablate the residual connection from the network, while still passing student outputs to the supervisor (w/o res). In this case (w/o res), we can observe the better KITTI results over the no connection case (off), but worse than our full design. These results indicate that conditioning supervisor with the student helps find residual function $\Delta f_\phi(\hat{\mathbf{y}}_s) \approx \mathbf{y} - \hat{\mathbf{y}}_s$, especially on distant domains, i.e., Things $\leftrightarrow$ KITTI. Overall, passing with residual connection (full) performs consistently better.

Shared encoder. Shared encoder design results in a better flow accuracy, as shown in Table 3.3. Our method uses the shared encoder design (Fig. 3.2b). Separating the encoder results in worse EPEs for Sintel and KITTI. Not only the accuracy, but it also results in a less efficient training pipeline due to the increased number of parameters by the encoder.

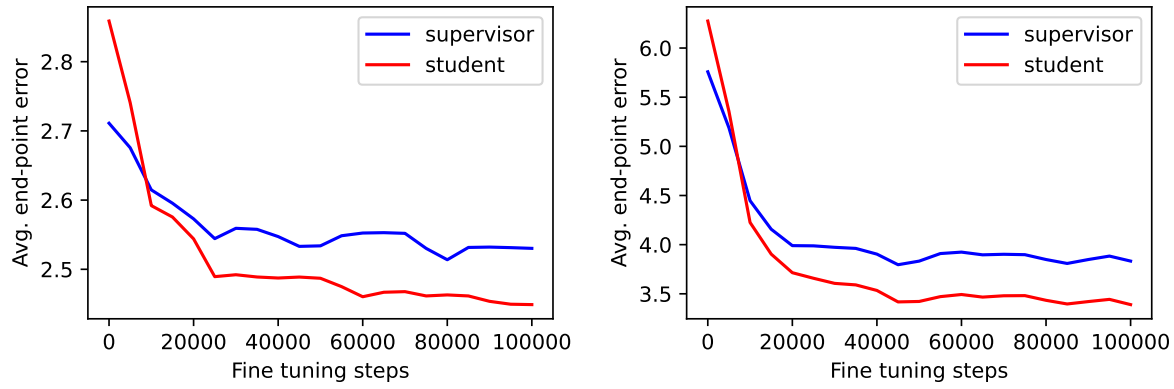
Teacher loss type. We propose to train the supervisor network with labeled and unlabeled data. Supervised teacher (TS) loss utilizes the labeled dataset for supervisor learning. In both datasets, TS improves the baseline performance. We additionally apply the unsupervised teacher loss (+TU), which results in better EPE in KITTI but is ineffective for Sintel. This is related to the results of pure unsupervised learning [31], where the loss is shown to be more effective on KITTI than on Sintel.

Virtual KITTI.

With our method, we can utilize a virtual driving dataset, VKITTI [3], for training a better model for the real KITTI dataset. In Table 3.4, we show results obtained by VKITTI. In the supervised case, our base network (RAFT) shows 3.64 EPE on KITTI, and SeperableFlow [83] shows 2.60 EPE. In the semi-supervised case, our method leverages the unlabeled KITTI multiview set additional to VKITTI, and we train the RAFT network for 50k steps; the resulting network performs better on KITTI.

Supervisor vs. student.

In Fig. 3.4, we show the performance of the student and the supervisor networks by training steps. We use clean inputs for both the student and the supervisor networks for evaluation. Interestingly, we observe that the student is better than the supervisor during fine tuning. In addition, we could observe EPEs of the student and the supervisor are correlated, and they are both improved by our semi-supervised training. In knowledge distillation, we can observe similar behavior [72], where a student model shows better validation accuracy than a teacher model.



(a) Sintel

(b) KITTI

Figure 3.4: **Supervisor vs. student EPEs** during semi-supervised fine tuning. EPEs measured on unseen portion of data

3.3.3 Qualitative results

We provide qualitative results on KITTI (Fig. 3.5, 3.6) and Sintel (Fig. 3.7). Since the KITTI dataset includes sparse ground truth flows, training by the ground truth supervision often results in incorrect results, especially in boundaries and deformable objects (e.g., person on a bike). Thus, in this case, we can expect better flows by exploiting semi-supervised methods. The Sintel Final dataset includes challenging blur, fog, and lighting conditions as shown in the examples. Interestingly, the semi-supervision without a target dataset label helps improve the challenging regions, when our method is used.

In Fig. 3.8, we show results on DAVIS dataset [7] by the pretrained models and our fine tuned models; the dataset does not contain any optical flow ground truth. From the training set with 3,455 frames, we utilize 90% of frames for training and the rest for qualitative evaluation. Even though our fine tuning does not utilize ground truth, we can observe a clear positive effect in several challenging regions. Especially, our method improves blurry regions (e.g., moving hand of the dancer) and object boundaries.

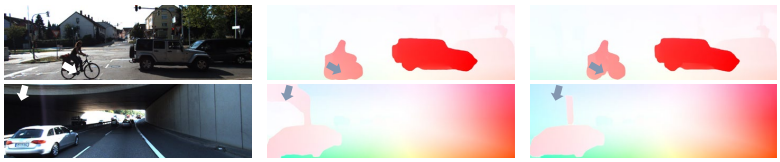
3.3.4 Comparison to State-of-the-arts

Experimental settings. In this experiment we compare our method to the existing supervised methods. The biggest challenge in supervised optical flow is that we do not have ample ground truth flows for our target domains, e.g., 200 pairs labeled in KITTI. Thanks to our semi-supervised method, we can train with related videos for better performance on the target datasets. For GMA, we use $\alpha = 0.25$ (Sintel) and $\alpha = 0.05$ (KITTI).

Dataset configuration. Results evaluated on the training sets of each dataset are experimented with the setting described in Sec. 3.3.1. On the other hand, the results on the test splits of the two benchmarks are obtained by justifiable external datasets for semi-supervised training. Though our method is able to utilize the unlabeled portion of Sintel, we bring another external set to avoid the relation to the testing samples, as follows. To assist Sintel performance, we use Spring (abbr. to Spg^u) [85], which is the 'open animated movie' by Blender Institute, similar to Sintel [86]. In Spring, we use the whole frames (frame no. 1-11,138) without any modification. Additionally, we use Sintel (train) with interval two as unlabeled. For KITTI, we additionally use KITTI multiview dataset, which does not contain ground truth optical

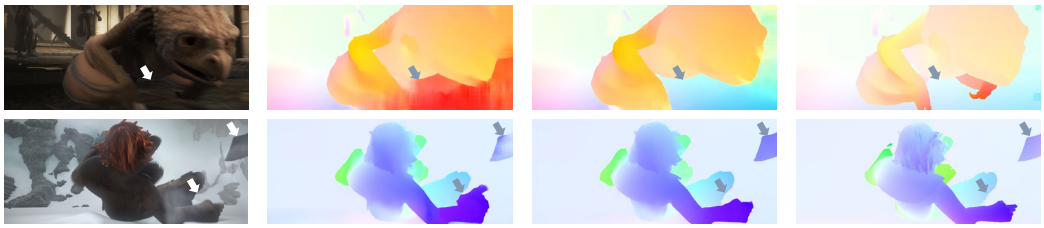
(a) Input (b) Sup on KITTI (c) Ours (d) Ground truth

Figure 3.5: **Qualitative results on KITTI.** We visualize optical flow predicted by **(b)** supervised on target dataset and **(c)** semi-supervised w/o target label. Note that sparse ground-truth **(d)** is not sufficient to make a clear boundary of objects (marked with arrows), while our method shows better results



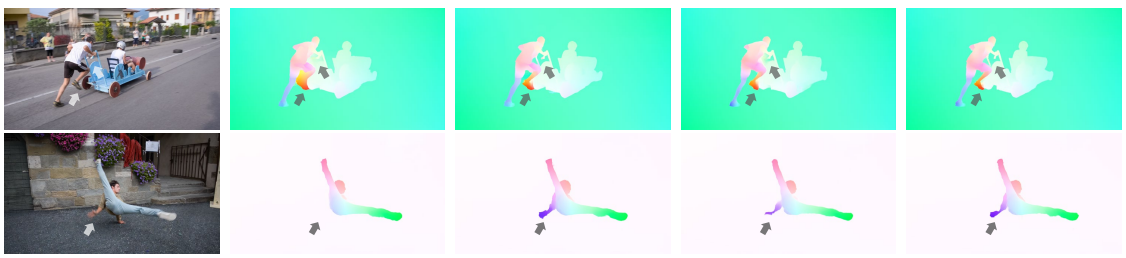
(a) Input (b) Sup (c) Ours

Figure 3.6: **Qualitative results on KITTI testing samples.** We compare the supervised model (Sup) with the semi-supervised model (Ours). Both models exploit KITTI labels; ours utilizes additional unlabeled KITTI for fine tuning



(a) Input (b) Sup on Things (c) Ours (d) Ground truth

Figure 3.7: **Qualitative results on Sintel.** We visualize optical flow predicted by **(b)** supervised on FlyingThings3D and **(c)** semi-supervised without target label. Though ours trained without target labels, it successfully adapts the pretrained model to the target domain. Improved areas are marked by arrows



(a) Input (b) CT (Sup) (c) CT (**Ours**) (d) CTSKH (Sup) (e) CTSKH (**Ours**)

Figure 3.8: **Qualitative results on DAVIS dataset [7].** We fine tune each pretrained network (Sup) on Davis dataset by our semi-supervised method (Ours). Improved regions are marked with arrows. These optical flows are inferred on unseen portion of the dataset

Table 3.5: **Comparison to state-of-the-arts.** Data usage is abbreviated to FlyingChairs (C), FlyingThings3D (T), Sintel (S), KITTI (K), HD1K [4] (H), Sintel unlabeled (S^u), KITTI unlabeled (K^u), and Spring (Spg^u). For labeled datasets, we follow the training scheme detailed in each paper. $RAFT^{tf}$ is our implementation in TensorFlow

Labeled data	Unlabeled data	Method	Sintel (train) Clean	Sintel (train) Final	KITTI (train) EPE	KITTI (train) Fl-all (%)	Sintel (test) Clean	Sintel (test) Final	KITTI (test) Fl-all (%)
C+T	-	RAFT [8]	1.43	2.71	5.05	17.4	-	-	-
		$RAFT^{tf}$	1.46	2.80	5.79	18.8	-	-	-
		GMA [84]	**1.30**	2.74	4.69	17.1	-	-	-
		SeparableFlow [83]	**1.30**	2.59	4.60	15.9	-	-	-
	S^u/K^u	$RAFT^{tf}$+FS (Ours)	**1.30**	**2.46**	**3.35**	**11.12**	-	-	-
C+T+S+K+H	-	RAFT [8]	(0.77)	(1.27)	(0.63)	(1.5)	1.61	2.86	5.10
		GMA [84]	(0.62)	(1.06)	(0.57)	(1.2)	**1.39**	2.47	5.15
		SeparableFlow [83]	(0.69)	(1.10)	(0.69)	(1.60)	1.50	2.67	**4.64**
	Spg^u/K^u	RAFT+FS (Ours)	(0.75)	(1.29)	(0.69)	(1.75)	1.65	2.79	4.85
		GMA+FS (Ours)	(0.63)	(1.05)	(0.61)	(1.47)	1.43	**2.44**	4.95

flows; we use the training split of the dataset to avoid duplicated scenes with testing samples.

Results. In Table 3.5, we report the C+T result, which is a commonly used protocol to evaluate the generality of models. Since our method is designed to use unlabeled samples, our method exploits each target dataset – which is not overlapped with each evaluation sample – without ground truth. The results indicate we could achieve better accuracy than the supervised-only approaches. Interestingly, our approach performs better than the advanced architectures, i.e., GMA [84] and SeperableFlow [83], in the C+T category.

In 'C+T+S+K+H', we report the test results with the external datasets. In the KITTI test result where we have access to 200 labeled samples, using additional samples results in better Fl-all (5.10 $\rightarrow$ 4.85, 5.15 $\rightarrow$ 4.95). For Sintel, we test on two base networks: RAFT and GMA. In two networks, our method makes improvements on Sintel Final (test). However, our method does not improve accuracy on Sintel Clean set. That is because we use the external Spring dataset, which includes various challenging effects, e.g., blur, as Sintel Final, the student model becomes robust to those effects rather than clean videos. Note that, for Sintel test, our improvement is made by the video of a different domain which is encoded by a lossy compression, like most videos on the web.

3.3.5 Limitations

A limitation of our approach is that it depends on a supervised baseline, which is sometimes less preferable than an unsupervised approach. A handful of unsupervised optical flow researches have achieved amazing performance improvement. Especially, on KITTI (w/o target label), unsupervised methods have performed better than supervised methods, and the recent performance gap has been widened (EPE: 2.0 vs 5.0). Unfortunately, we observe that our method is not compatible with unsupervised baselines; the semi-supervised fine tuning of SMURF [31] with our method ($T+K^u$) results in a worse EPE (2.0 $\rightarrow$ 2.5) on KITTI. Thus, one of the future work lies upon developing a fine-tuning method to improve the unsupervised baselines.

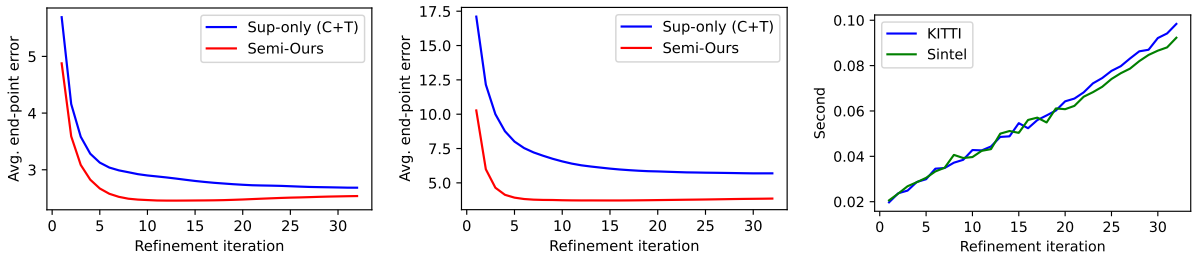
Nonetheless, our work contributes to the research by ameliorating the limitation of a supervised method, suffering from worse generalization. A supervised method is often preferable than an unsupervised

one for higher performance even without target labels; we show ours can be used in such cases. For example, on Sintel Final, ours shows better EPE (2.46) than the supervised one [8] (2.71) and the unsupervised method [31] (2.80).

3.4 Supplementary Material

3.4.1 Additional Results

Refinement Steps and Faster Inference



(a) Sintel Final EPE (b) KITTI EPE (c) Time vs. Iteration

Figure 3.9: **Results w.r.t. the refinement iteration**. **(a-b)** shows validation EPEs of the baseline (Sup-only) and ours (Semi-Ours). **(c)** shows the inference time per frame with different resolutions: KITTI (1242×375) and Sintel (1024×436). For the comparison of time, we use a single RTX 3090 GPU (24GB VRAM) with our TensorFlow implementation of RAFT.

In Fig. 3.9, we show EPEs with respect to refinement steps. Since we base our network on RAFT [8], setting an appropriate refinement step is crucial for the performance of estimation. By the experiments, we observe that training with our self-supervision method results in a faster convergence. In Fig. 3.9a-3.9b, we show that our semi-supervised learning method has a faster convergence iteration than the baseline (Sup-only) model. That is the reason we choose shorter refinement steps, i.e., 12, than the original paper, i.e., 32. Thanks to the fewer refinement steps, the inference time is reduced by about 50% (see Fig. 3.9c) on a target dataset, when we set the iteration to 12 instead of 32.

Stopping Gradients

Our design choice for `stop_grad` is to make the supervisor module an isolated module, which makes gradient computation step for the flow supervisor more compact and efficient. Furthermore, stop-gradient is a widely adopted technique in self-supervised learning to prevent a degenerated solution [73]. We discovered that disabling `stop_grad` for each module shows worse results on Sintel Final:

Teacher encoder	$\mathbf{h}_s$	$\hat{\mathbf{y}}_{FS}$	**Ours** (all enabled)
2.58	2.52	diverged	**2.46**

Note that `stop_grad` for $\hat{\mathbf{y}}_s$ belongs to original RAFT.

More Qualitative Results on Sintel and KITTI

We show more qualitative comparisons in Fig. 3.12-3.13. Even though we do not exploit a ground truth of the target dataset, our method clearly improves challenging regions. In the KITTI examples, our

method is especially effective on objects near image frames and shadows. For Sintel – which includes many motion blurs and lens effects – the network becomes robust to those challenging effects, as shown in the examples.

3.4.2 Experimental Details

Architecture

We show the implementation of our method in Fig. 3.11. Our architecture is based on RAFT [8], where the iterative refinement plays an important role. The self-supervision process is similar to the supervised learning, where the intermediate flows $\hat{\mathbf{y}}_s^i$ are supervised by a target flow $\mathbf{y}$. In the RAFT paper, the decaying parameter $0 < \gamma \leq 1$ is used in the loss function:

$$\ell_{\text{sup}} = \sum_{i=1}^{n} \gamma^{n-i} \|\hat{\mathbf{y}}_s^i - \mathbf{y}\|_1. \tag{3.9}$$

Similarly, we apply the decaying strategy in our self-supervision by minimizing:

$$\ell_{\text{FS}} = \sum_{i=1}^{n} \gamma^{n-i} \rho(\hat{\mathbf{y}}_s^i - \hat{\mathbf{y}}_{\text{FS}}^{n+m}), \tag{3.10}$$

where $\hat{\mathbf{y}}_{\text{FS}}^{n+m}$ is the pseudo label predicted by the flow supervisor. Specifically, we use $\gamma = 0.8$ (ℓ_{sup}), $\gamma = 0.8$ ($\mathcal{L}_{\text{FS}}$), and $\gamma = 1.0$ ($\mathcal{L}_{\text{TS}}$). For KITTI, we use $\gamma = 0.8$ ($\mathcal{L}_{\text{TS}}$, $\mathcal{L}_{\text{TU}}$) for the supervisor model.

Padding Operation

As mentioned in the main text, we use a cropping operation to give supervision from the supervisor network to the student network. Passing student outputs requires the student outputs to be aligned with the uncropped images (i.e., teacher inputs). Since RAFT use 1/8 processing resolution, we use a padding operation performed at 1/8 resolution, and the random offset coordinates for cropping operation is constrained to multiples of 8; this results in sizes of all inputs – augmented, privileged, and crop offsets – to be multiples of 8.

Optimization

In the fine tuining stage, we use batch size 1 each from a labeled dataset and an unlabeled dataset, which requires a single RTX3090 GPU, and it takes about one day to converge. We use Adam optimizer [61], and we decay the learning rate from 10^{-5} by 1/2 every 25,000 steps. Different from supervised training, we use the generalized Charbonnier loss $\rho(\cdot)$ in ℓ_{sup}, ℓ_{self}, and ℓ_{FS} for semi-supervised training. Detailed hyper-parameters and reproducible experimental settings are provided in our code.

Unsupervised Loss. In $\mathcal{L}_{\text{TU}}$ and ablation results, we use the unsupervised loss $\ell_{\text{unsup}}(\cdot)$ for comparison. As mentioned in the main paper, we use the photometric loss, occlusion handling, and the smoothness loss, which are brought from SMURF implementation [1]. Following the code, we use `census=1`, `smooth1=2.5`, `smooth2=0.0`, and `occlusion='wang'` for Sintel; `census=1.0`, `smooth1=0.0`, `smooth2=2.0`, and `occlusion='brox'` for KITTI. We do not use the self-supervision loss (`selfsup=0.0`) since our method includes the similar self-supervision strategy.

[1] https://github.com/google-research/google-research/tree/master/smurf

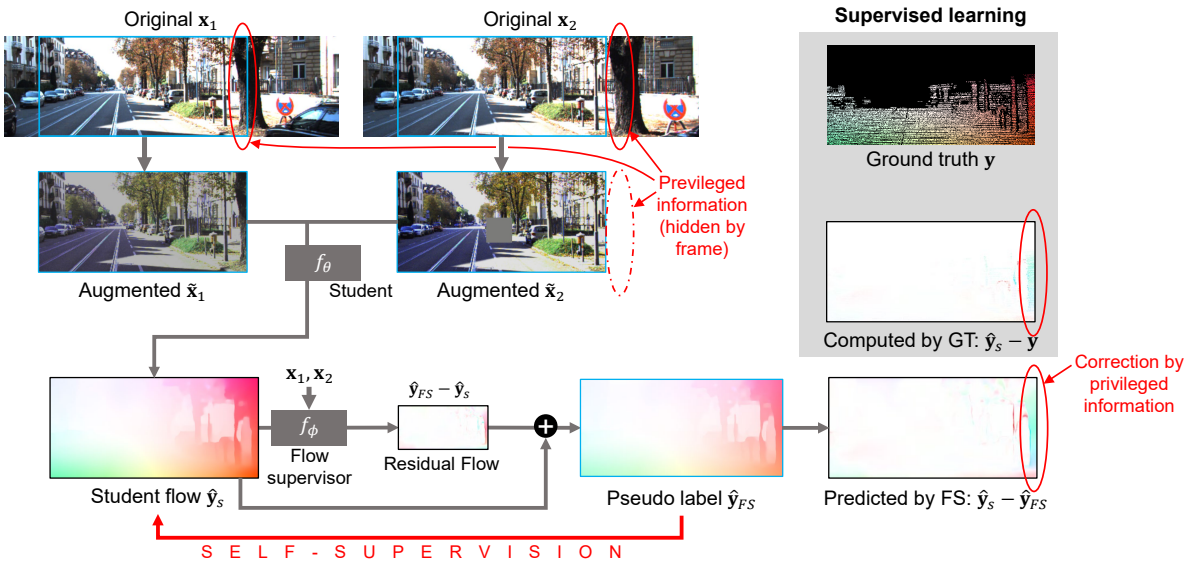
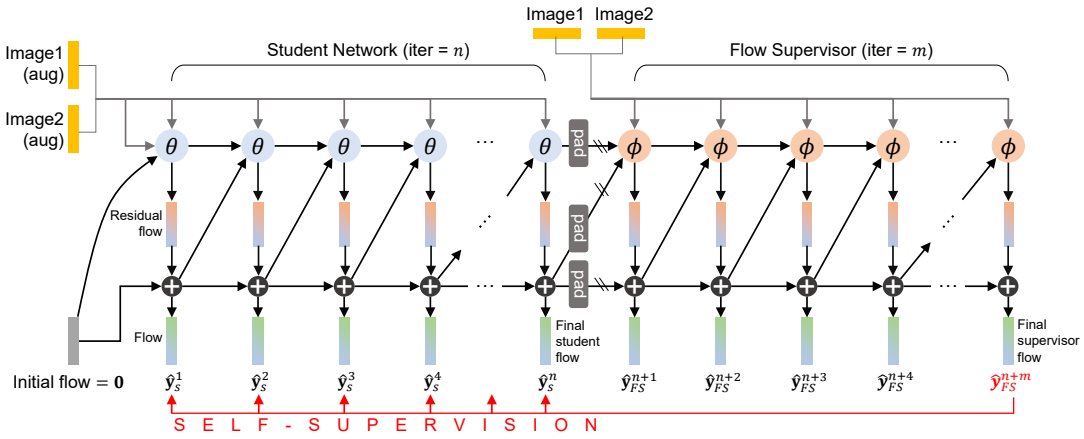


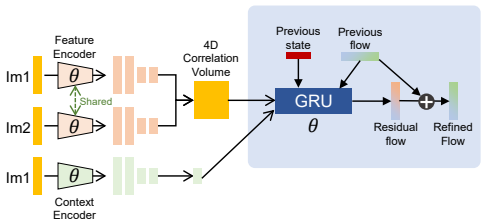
Figure 3.10: **Self-supervision example.**

3.4.3 Self-Supervision Example

Our self-supervision is performed by the flow supervisor which is conditioned on the student outputs and clean inputs. The process is summarized in Fig. 3.10. In the example, we show consecutive frames from a driving scene, where the observer is moving forward, so that objects near the image frame are hidden by the frame. For example, the tree on the right side is not visible in the second cropped image, while the original second image contains the tree. Thus, the teacher prediction can correct the student prediction by utilizing the privileged information, as shown in the figure. Compared to the ground truth, we can observe the correcting direction $\hat{\mathbf{y}}_s - \hat{\mathbf{y}}_{\text{FS}}$ is close to $\hat{\mathbf{y}}_s - \mathbf{y}$ computed by GT. Thus, our flow supervisor can generate a desirable supervision signal to guide the student network by the privileged inputs.

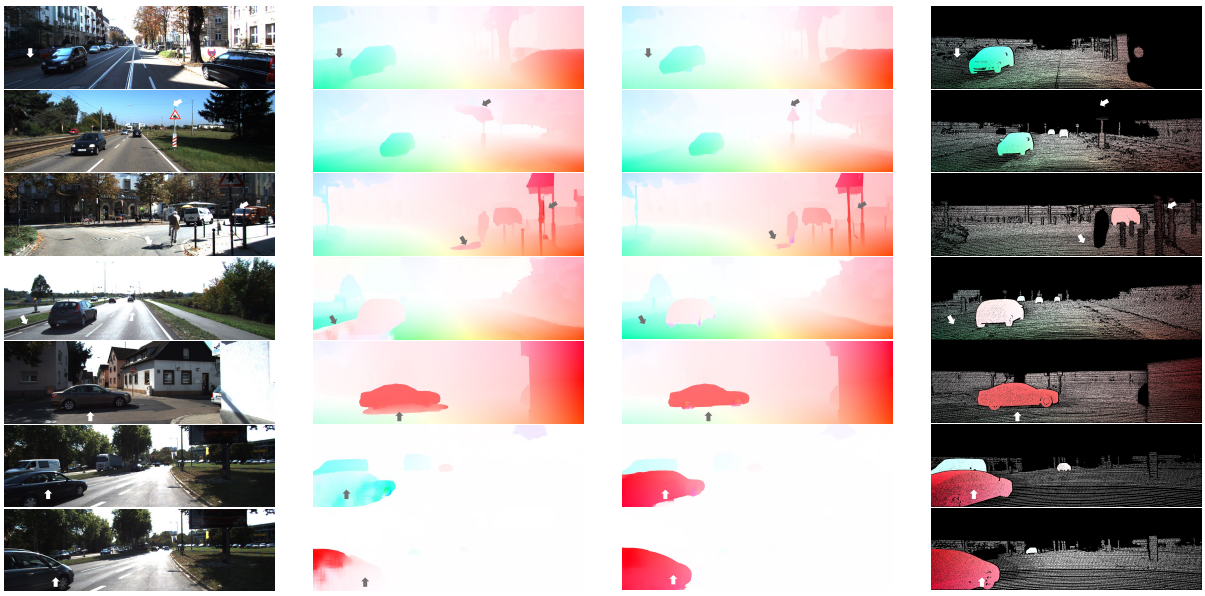


(a) Overall structure for self-supervision



(b) A refinement step of RAFT [8]

Figure 3.11: **Detailed architecture. (a)** summarizes the detailed structure of our flow supervisor. Our flow supervisor shares the design of iterative refinement RNN module of RAFT. Since we feed full images to the flow supervisor, we pad the outputs of student network to feed them to the supervisor. **(b)** depicts one refinement step of RAFT with the feature encoder and context encoder. For technical details of each layer, please refer to [8].

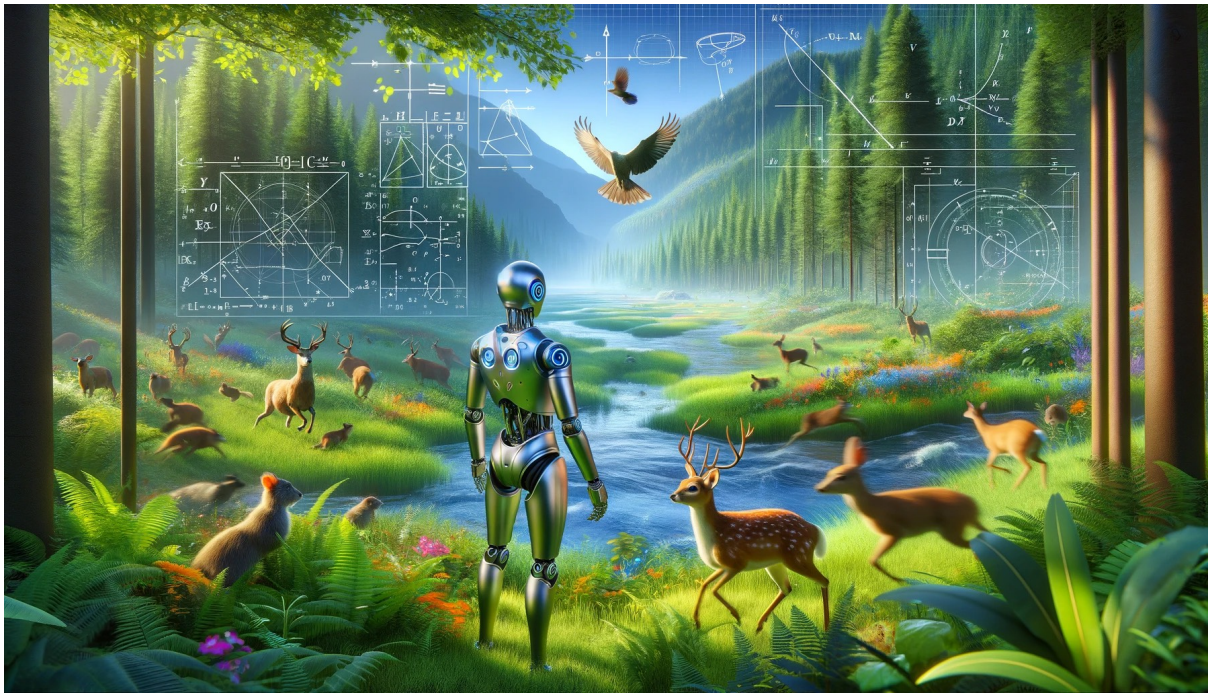


(a) Input (b) Sup-only (c) Semi-Ours (d) Ground truth

Figure 3.12: **Qualitative results on KITTI.** We compare results of RAFT trained on VKITTI. **(b)** shows optical flows predicted by RAFT pretrained on VKITTI. **(c)** shows flows prediected by our semi-supervised method, which utilizes an additional KITTI dataset without ground-truth. All results are obtained on unseen samples.

(a) Input (b) C+T (Sup-only) (c) C+T (Semi-Ours) (d) Ground truth

Figure 3.13: **Qualitative results on Sintel Final.** We compare results of RAFT trained on C+T. **(b)** shows optical flows predicted by RAFT pretrained on FlyingChairs and FlyingThings. **(c)** shows flows prediected by our semi-supervised method, which utilizes an additional Sintel dataset without ground-truth. All results are obtained on unseen samples.



Real world is three-dimensional and physics-based. We propose kinematic fields to represent physically-based motion in three-dimensional space to better reconstruct the dynamics and geometry of the world. This figure is created by DALL.E (ChatGPT).

Chapter 4. Learning Kinematic Fields for Better Reconstruction of Dynamic Radiance Fields

Recognizing motion is crucial to how we visually perceive the world [87], and video is the medium that carries the motion to viewers in a remote place or at a different time. Through the lens of video, we watch and enjoy movies, user-generated contents, and remote video chats. While videos carry motion in a 2D portrayal, transforming videos into three-dimensional experiences can elevate our perception of the content to co-presence via applications such as teleportation [88].

Dynamic radiance fields [89, 33, 35, 90, 91, 92] have been studied as a promising representation for reconstructing 3D dynamic scenes; it enables realistic novel view and time synthesis, which can upgrade 2D videos into 3D videos. Since multi-view cameras can give more information about the geometry in a scene, existing works have utilized camera rigs with dozens of cameras to capture multi-view video datasets [33, 93, 90]. Although high-quality multi-view videos result in undeniably better quality than monocular videos do, the multi-view devices and environments are prohibitive in most cases.

Thus, there have been efforts to reconstruct dynamic radiance fields from monocular videos [34, 36, 94]. Compared to multi-view videos, monocular videos provide sparser information, which makes the reconstruction more challenging. For instance, there is scale ambiguity in monocular video, making it difficult to accurately locate an object within a space [95]. Thus, dynamic scene reconstruction from monocular videos has been considered an ill-posed problem.

In this paper, we incorporate kinematics to fill the gap between sparse monocular videos and real-

world physics, potentially enhancing novel view and time synthesis. Our core idea is that motion within the real-world is governed by the principles of kinematics in physics, implying that representations of this world, such as videos, are inherently anchored in kinematics. Building on this foundation, we generalize conventional displacement fields – often called scene-flow fields – to kinematic fields. In prior work, motion in radiance fields has been controlled via the regularization in the frequency domain [96, 97], displacement fields [34], and deformation fields [98].

Different from these approaches, our kinematic field regulates motion in radiance fields using kinematics; it innately creates continuous trajectories and enables us to impose regularization based on kinematic theory. We integrate physics-based regularizers to make the kinematic field physically plausible and thus improve the dynamic radiance field accordingly. For instance, the rigidity regularizer based on the first and the second invariant of the strain rate is applied on the kinematic field, suppressing non-rigid motion in the field. We quantitatively and qualitatively validate the proposed regularizers on the NVIDIA Dynamic Scenes dataset, showing the effectiveness of our method over the existing models. Lastly, the kinematic field uncovers the kinematics within a scene without a ground truth; it estimates not only point-wise 3D displacement within a scene, but also underlying velocity, acceleration, and jerk.

4.1 Related Work

Neural radiance fields. In recent years, neural radiance fields [32, 99, 100] have emerged as a promising approach for learning an implicit neural representation of a 3D scene. Combined with the differentiable volume rendering technique, it has enabled realistic novel-view synthesis [32, 99, 100, 101].

In essence, a radiance field $\mathcal{F}_{\text{ST}}$ maps a 5D vector (x, y, z, θ, ϕ) to a 4D output vector $(\mathbf{c}_s, \sigma_s)$:

$$\mathcal{F}_{\text{ST}} : (x, y, z, \theta, \phi) \rightarrow (\mathbf{c}_s, \sigma_s), \tag{4.1}$$

where the view-dependent color $\mathbf{c}_s = (r, g, b)$ and density σ_s are learned in relation to the spatial position (x, y, z) and the viewing direction (θ, ϕ). With the volumetric representation learned via an iterative optimization, volume rendering techniques [102, 103, 32] are employed to aggregate colors and densities along a cast ray. Consequently, this leads to realistic novel-view synthesis from the inferred radiance, after being trained on dozens of images.

Dynamic radiance fields. Radiance fields also facilitate the novel-view synthesis of scenes with moving objects [33, 98, 104, 93, 90]. While this is more challenging than the static case, dynamic radiance fields incorporating motion allow for the rendering of a frame at a desired time and viewpoint. As a result, it can be utilized to make visual effects such as bullet time, stabilized, dolly zoomed, and stereo videos. Commonly, the time-dependent radiance fields are referred to as *dynamic radiance fields*, a term we will adopt throughout this paper.

In dynamic radiance fields, radiance is dependent on a varying time t. This modifies Eq. 4.1 by introducing a time-dependent term, resulting in:

$$\mathcal{F}_{\text{DY}} : (x, y, z, t, \theta, \phi) \rightarrow (\mathbf{c}_d, \sigma_d), \tag{4.2}$$

where the color and density are conditioned on time t. In this formulation, moving geometries of objects (*e.g.*, translation and deformation) and changing radiance (*e.g.*, moving shadows) can be learned via optimization.

Learning a dynamic representation presents inherent challenges due to the introduction of temporal dimension. To optimize this representation, a high-quality multi-view dataset can be employed to

obtain fine rendering results [33, 93, 90]. Meanwhile, realistically captured imagery of a scene tends to be non-synchronized and is likely to be obtained with a limited number of cameras and restricted displacements; this often results in a monocular setup [91, 98, 104, 34]. Previous researches have delved into deformable models for monocular videos [105, 98, 104, 94]. In these methods, a deformation field maps a spatio-temporal coordinate to a reference canonical space. While the canonical space aids in maintaining radiance consistency across global time frames, it tends to exhibit reduced effectiveness in generic monocular videos, where new objects might appear within a video [97]. In this study, we embrace the concept of scene flow fields [34], which is constructed without canonical space and deformation networks.

Regularization with kinematic priors. Monocular videos present challenges in reconstruction due to the constrained viewpoints of moving objects. To address this, integrating auxiliary flow fields has proven to be effective [34, 36, 106, 35]. Existing studies employ either scene flow fields [34, 36, 35] or velocity fields [106] to capture spatial displacements between consecutive frames. Since the scene flows can give spatial correspondences of the dynamic radiance field, it can be used to promote photometric consistency between the renderings of a given ray and its time-deformed counterpart.

Physical priors incorporated with scene flow fields play a pivotal role in regulating the radiance field. Regularization of motion has been applied to the dynamic field, bridging the gap between the sparse visual data and plausible real-world motion. For instance, a smoothness regularizer ensures that the displacements within close proximity are similar to each other, and kinetic energy regularizer promotes consistency between forward and backward flows, resulting in smoother displacement fields [34]. More deeply integrated with physics, the elastic energy regularizer minimizes the deformation between the canonical space and the deformed space [98]. In fluid reconstruction works [107, 108], velocity fields grounded in physics have been shown to be critical in accurate 4D reconstruction. For instance, Navier-Stokes and transport equations can be integrated in reconstructing smoke [107] and fluid surfaces [108] with dynamic radiance fields to reconstruct more physically plausible structures.

We integrate kinematics to reconstruct dynamic radiance fields from generic monocular videos. We introduce the kinematic field, which captures motion through kinematic quantities such as velocity, acceleration, and jerk. Different from existing velocity field-based methods [107, 106, 95], our method has higher-order terms (*i.e.*, acceleration and jerk) as latent of spatio-temporal correspondences. Further enhancing our approach, we incorporate physics-driven regularizers based on a transport equation [107] and the strain rate tensor, addressing dynamics of moving particles and rigid motion.

4.2 Approach

4.2.1 Overview

Our system takes as input a video sequence, denoted as $(I_i, t_i, P_i)_{i=1}^{N}$, where N is the number of frames. For each frame, we have an image $I_i \in \mathbb{R}^{H\times W\times 3}$, a timestamp $t_i \in \mathbb{R}$, and a precomputed camera pose $P_i \in SE(3)$ and intrinsic matrix $K_i \in \mathbb{R}^{3\times 3}$. From this input, our primary objective is to synthesize an image, denoted as $\hat{I}_k$, corresponding to a novel time and viewpoint (t_k, P_k, K_k). The parameters t_k, P_k, and K_k are not restricted to those found in the original video input, allowing our system to render images from previously unobserved times and viewpoints.

To achieve this image synthesis for novel times and views, we leverage the radiance-field-based volume rendering [32]. Inspired by existing methods [34, 109], we train two distinct radiance fields (Fig. 4.1): a

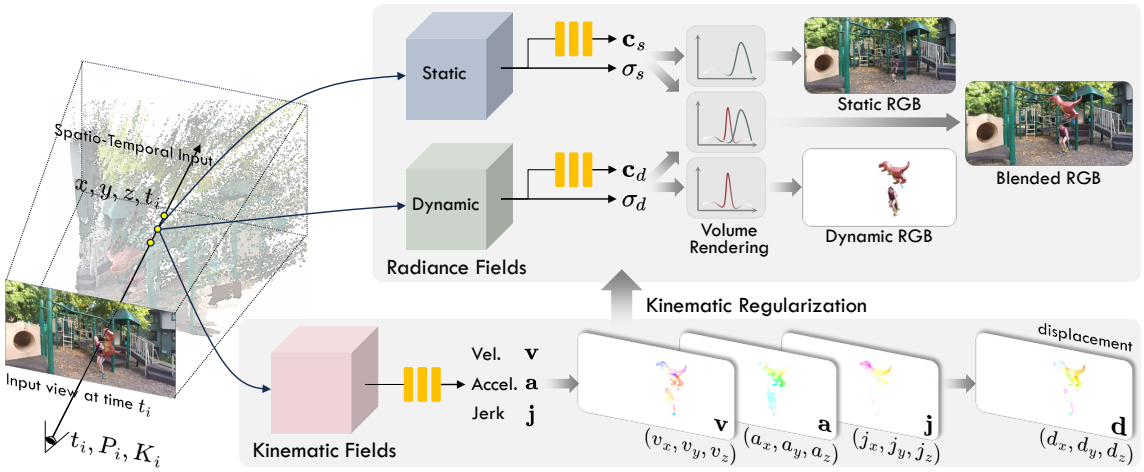


Figure 4.1: **Radiance and kinematic fields.** This figure summarizes the three fields our method utilizes. The static and dynamic radiance fields are used for rendering, and the kinematic field is used in the training phase, regularizing the dynamic radiance field.

dynamic radiance field, denoted as $\mathcal{F}_{\text{DY}}$ (Eq. 4.2), and a static radiance field, denoted as $\mathcal{F}_{\text{ST}}$ (Eq. 4.1). To blend the two fields, we utilize a technique from D^2-NeRF [109], where rendering weights for dynamic and static samples are derived from the volume densities.

Addressing the challenge of achieving physically plausible spatio-temporal structures, we introduce the novel kinematic field. The kinematic field accounts for the kinematic quantities (*e.g.*, velocity and acceleration). These quantities can be converted into displacement (Eq. 4.8), enabling us to use the photometric consistency loss (Eq. 4.14) for spatio-temporal regularization, as used in previous work [34]. With these kinematic estimations, we apply physics-based regularization techniques. These techniques involve density transport (Eq. 4.12) and motion rigidity through the strain rate tensor (Eq. 4.11). Combining the photometric loss (Eq. 4.14) with these techniques enhances the reconstruction of radiance and geometry, leading to improved novel time and view synthesis performance from underdetermined data such as a monocular video.

4.2.2 Dynamic and Static Radiance Fields

In monocular dynamic radiance field methods, it is common to decompose the radiance fields into dynamic and static fields [34, 91, 94] to prevent undesirable motion in the static area. We employ tensor-decomposition-based representations [90, 110] for both dynamic and static fields. Specifically, for the dynamic radiance field $\mathcal{F}_{\text{DY}}$, we use HexPlane [90], and for the static radiance field $\mathcal{F}_{\text{ST}}$, we use TensoRF [110]. Further details on the tensor decomposition can be found in the supplementary material.

Definition. The dynamic and static radiance fields are defined in Eq. 4.1-4.2 ($\mathcal{F}_{\text{ST}}$ and $\mathcal{F}_{\text{DY}}$). Here, the inputs correspond to the position (x, y, z), the timestamp t, and the viewing direction (θ, ϕ) from the camera origin. Each field yields the color $\mathbf{c}$ and the volume density σ. Note that the static field is independent of time t, ensuring a consistent structure throughout the whole time span.

Composite rendering. We use a composite rendering method to obtain the final rendering with the two radiance fields. It is to be noted that individual components (*i.e.*, dynamic-only and static-only) can be rendered using the standard volume rendering technique [32]. To combine both fields, we employ an additive composition method in which each field influences the transmittance of a ray [109]. Let's

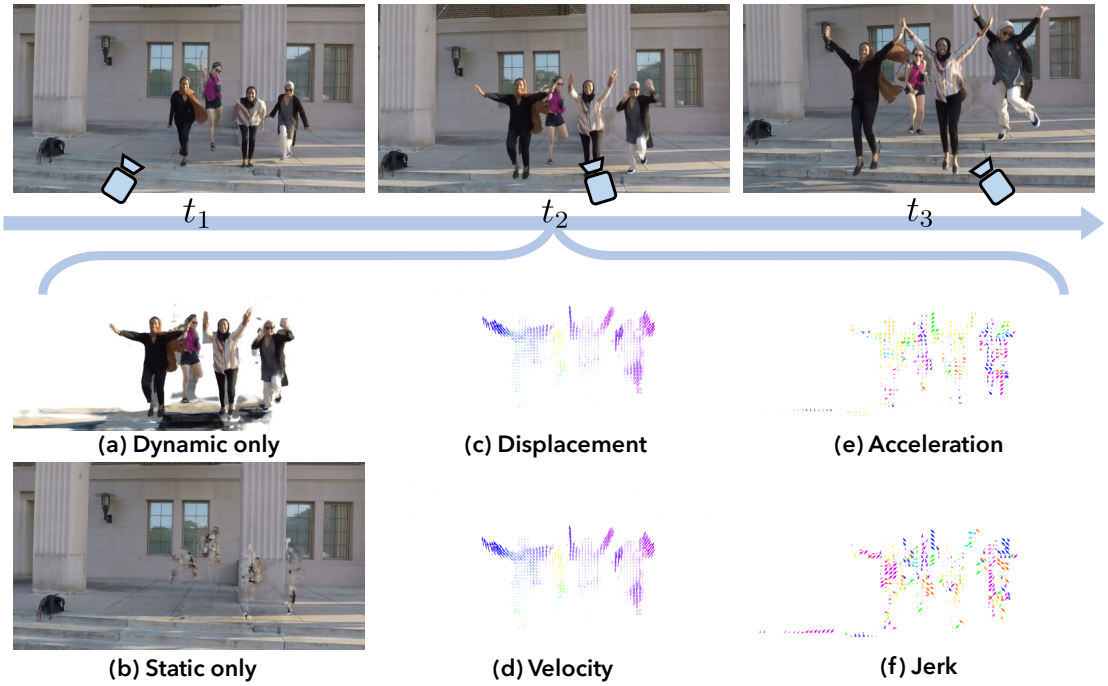


Figure 4.2: **Visualization of each predicted component.** The top row presents synthesized views at different times. In (a) and (b), the dynamic and static components of the scene at time t_2 are depicted separately. (b) In the case of an inobservable static area in the entire sequence (*e.g.*, the space behind the jumping people), radiance might not be correct. The displacement field (c) can be computed by Taylor approximation (Eq. 4.8) with motion fields (d-f): velocity, acceleration, and jerk. Each field is visualized through reprojecting each field to the camera view. The standard HSV visualization [9] is used to colorize arrows.

consider a camera ray $\mathbf{r}(\tau) = \mathbf{o} + \tau\mathbf{u}$, where $\mathbf{o}$ is the origin and $\mathbf{u}$ is the direction. With a predefined range for the ray distance $[\tau_n, \tau_f]$, the rendering function is formulated as:

$$\hat{C}(\mathbf{r}, t_i) = \int_{\tau_n}^{\tau_f} T(\tau) \left(\sigma_s(\tau) \cdot \mathbf{c}_s(\tau) + \sigma_d(\tau, t_i) \cdot \mathbf{c}_d(\tau, t_i)\right) d\tau, \tag{4.3}$$

$$\text{where } T(\tau) = \exp\left(-\int_{\tau_n}^{\tau} (\sigma_s(u) + \sigma_d(u, t_i)) du\right). \tag{4.4}$$

Here, each notation for the sample point is simplified for concise representation (*e.g.*, $\sigma_s(\tau) = \sigma_s(\mathbf{r}(\tau))$). We visualize the example of the composite rendering and separate rendering in the first and second rows in Fig. 4.2.

Dynamic and static separation. To factorize a scene into static and dynamic parts, we use a metric indicating the likelihood that a point belongs to the dynamic fields [109]. This metric is given by:

$$p_d = \frac{\sigma_d}{\sigma_d + \sigma_s}, \tag{4.5}$$

where σ_d and σ_s are density values sampled from a spatio-temporal point.

To facilitate the separation of the radiance fields, we employ the skewed binary entropy loss, as proposed by [109]. The entropy loss is defined as:

$$\mathcal{L}_b = -(p_d^k \cdot \log(p_d^k) + (1 - p_d^k) \cdot \log(1 - p_d^k)), \tag{4.6}$$

where the exponent $k \geq 1$ is the skew factor, controlling a bias toward static explanation. This entropy-based loss can encourage the radiance fields to have a clearer distinction between the static and dynamic components.

4.2.3 Kinematic Field

The kinematic field has a spatio-temporal structure to facilitate spatial and temporal queries. We use the tensor-decomposition-based representation [90] for the kinematic field. As depicted in Fig. 4.1, the kinematic field serves to regulate the radiance field, ensuring that the motion in the radiance field is physically plausible.

Definition. The kinematic field is defined as:

$$\mathcal{F}_{\mathrm{K}} : (x, y, z, t) \rightarrow (\mathbf{v}, \mathbf{a}, \mathbf{j}, \cdots), \tag{4.7}$$

where each term represents velocity, acceleration, and jerk, such that $\mathbf{v} = (v_x, v_y, v_z)$, $\mathbf{a} = (a_x, a_y, a_z)$, and $\mathbf{j} = (j_x, j_y, j_z)$. Note that we can extend the field into higher-order kinematic quantities such as snap (4th), crackle (5th), and so on.

Displacement. Using the provided kinematic quantities, we can approximate a particle's displacement at position (x, y, z) and time $t + \Delta t$ via the Taylor approximation, assuming that the kinematic quantities are consistent within a local temporal range Δt. The displacement $d(x, y, z, t, \Delta t)$ can be approximated by:

$$d(x, y, z, t, \Delta t) \approx \frac{(\Delta t)\mathbf{v}}{1!} + \frac{(\Delta t)^2\mathbf{a}}{2!} + \frac{(\Delta t)^3\mathbf{j}}{3!} + \cdots . \tag{4.8}$$

Computing the displacement of a point within a temporal range allows for computing the photometric consistency loss (Eq. 4.14) and scene flows from the kinematic field.

4.2.4 Kinematic Regularization

Given the kinematic field, it is important to make sure the learned motion dynamics are physically plausible. As Fig. 4.3 indicates, learning kinematic fields without adequate regularization may result in a sub-optimal kinematic field. The subsequent sections elucidate regularization strategies.

Integrity of kinematic fields. Learning the kinematic field $\mathcal{F}_{\mathrm{K}}$ naively does not guarantee the integrity of the kinematic quantities. During training, the quantities are supervised via the photometric consistency loss (Eq. 4.14), which is applied upon the approximated displacement (Eq. 4.8). It is important to note that even with supervision by the displacement, the physical relationships among the kinematic quantities may not be correct, since the higher-order terms in the displacement equation (Eq. 4.8) make the system underdetermined.

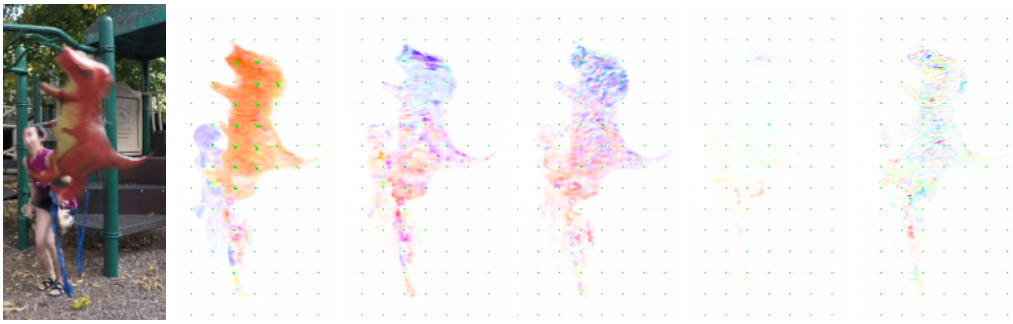
Given a velocity field, we can derive the acceleration field using the principle of advective acceleration. Given acceleration $\mathbf{a}$ and velocity $\mathbf{v}$ at a spatio-temporal coordinate, the partial derivative and advection yield $\mathbf{a} = \partial\mathbf{v}/\partial t + \mathbf{v} \cdot \nabla\mathbf{v}$. Similarly, we can formulate the relationship between $\mathbf{v}$, $\mathbf{a}$ and $\mathbf{j}$ as: $\mathbf{j} = \partial\mathbf{a}/\partial t + \mathbf{v} \cdot \nabla\mathbf{a}$.

To ensure kinematic consistency, we introduce the kinematic loss $\mathcal{L}_{\mathrm{K}}$ defined as:

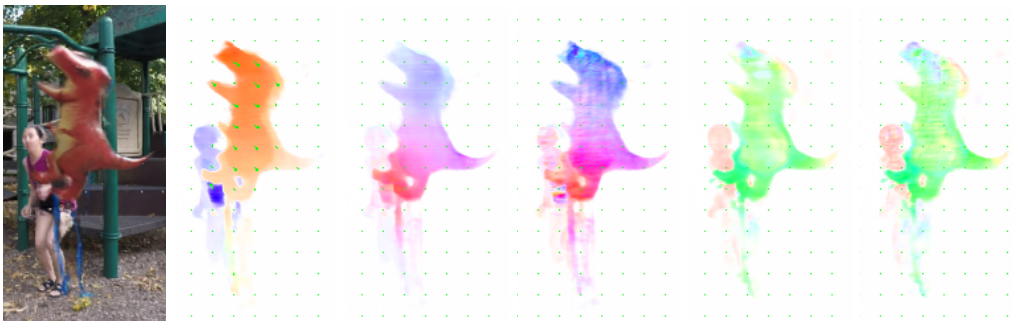
$$\mathcal{L}_{\mathrm{K}} = \left\| \mathbf{a} - \frac{\partial \mathbf{v}}{\partial t} - \mathbf{v} \cdot \nabla\mathbf{v} \right\|_2^2 + \left\| \mathbf{j} - \frac{\partial \mathbf{a}}{\partial t} - \mathbf{v} \cdot \nabla\mathbf{a} \right\|_2^2 + ..., \tag{4.9}$$

where ∇ is the Jacobian operation. To compute the partial derivatives and Jacobian matrices, we use a numerical method[1] as presented in a neural surface reconstruction work [111].

[1] $f'(x) = \frac{f(x+\epsilon) - f(x-\epsilon)}{2\epsilon}$



(a) Without kinematic regularization



(b) With kinematic regularization

Figure 4.3: **Effect of kinematic regularization.** We visualize the rendered RGB and motion of each field. Without kinematic regularization, motion fields tend to show granular patterns. Our kinematic fields not only make the field smoother but also satisfy the kinematic property, *i.e.*, $\mathbf{a} = \partial\mathbf{v}/\partial t + \mathbf{v} \cdot \nabla\mathbf{v}$. We abbreviate the advective equation to 'Adv.' in the figure.

Physics-informed regularization. In the context of regularizing radiance fields through the use of an auxiliary kinematic field, it is important to acknowledge the highly underdetermined nature of the problem. Essentially, there exists a broad spectrum of possible potential kinematic field solutions that could perfectly fit the observed training views. This complexity necessitates the introduction of a regularization strategy that can guide the motion representation toward more accurately reflecting real-world physics.

While not universally ideal, rigidity [98] and incompressiblility [107, 108] have been identified as effective metrics for encapsulating real-world dynamics; in videos that we normally encounter, objects in a scene have nearly consistent density and are mostly rigid in a short time range.

Rigidity. One benefit of having a velocity field is that we can measure the rate of deformation by its derivatives. A strain rate tensor D defines a kinematic property of a deformation; it captures the rate at which a material undergoes deformation, and it is crucial for ensuring that objects maintain their structural integrity during motion.

In this paper, we propose using the invariants of strain rate tensor [112, 113] as a measure of rigidity. The velocity gradient $\nabla\mathbf{v}$ sampled from the kinematic field, leads to the strain rate tensor D by:

$$D = \frac{1}{2}\left(\nabla\mathbf{v} + (\nabla\mathbf{v})^T\right). \tag{4.10}$$

The rigidity loss function $\mathcal{L}_R$ is formulated based on the first and second invariants of the strain rate

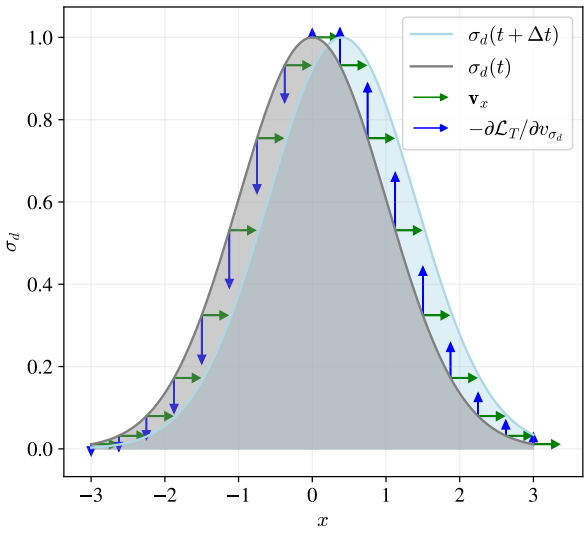


Figure 4.4: **Visualization of density variation by a spatial coordinate x.** The plot displays the gradient $-\partial\mathcal{L}_\mathrm{T}/\partial v_{\sigma_d}$ with arrows. With the velocity field directed to the right (*i.e.*, $\mathbf{v}_x = 0.3$), we can compute the gradient of the transport regularization $\mathcal{L}_\mathrm{T}$ w.r.t. the rate of density change $v_{\sigma_d} = \partial\sigma_d/\partial t$. Minimizing $\mathcal{L}_\mathrm{T}$ allows us to render the density at $t + \Delta t$ aligned with the flow field.

tensor:

$$\mathcal{L}_R = \underbrace{\lambda_\mathrm{div}(\nabla\cdot\mathbf{v})^2}_{\text{1st invariant}} + \underbrace{\left(\frac{1}{2}(\mathrm{tr}(D)^2 - \mathrm{tr}(D\cdot D))\right)^2}_{\text{2nd invariant}}, \tag{4.11}$$

where $\mathrm{tr}(\cdot)$ denotes the trace operator and λ_div controls the balance between the two terms. Note that the first invariant of strain rate measures dilational change (*i.e.*, volumetric change) with respect to time and the second invariant of strain rate measures distortional change.

Transport. A transport equation has been adopted in physics-informed neural networks [107]. By simplifying the Navier-Stokes equation, a transport loss can be defined as: $\frac{\partial\sigma}{\partial t} + \mathbf{v}\cdot\nabla\sigma = 0$. We leverage the physics-informed regularizers based on the transport equation and the divergence of the velocity:

$$\mathcal{L}_\mathrm{T} = \left(\frac{\partial\sigma_d}{\partial t} + \mathbf{v}\cdot\nabla\sigma_d\right)^2, \tag{4.12}$$

where σ_d is the volume density used for volume rendering. This aligns the velocity field with the density field (*i.e.*, Fig. 4.4), preventing a sudden disappearance or appearance of objects in the dynamic radiance field.

4.2.5 Learning Objectives

Cycle consistency. The cycle consistency is essential to ensure that the trajectories derived from the kinematic field do not introduce misalignments [34]. We use cycle constraints among four timestamps t, i, j, and γ. Given a reference time t, we randomly sample time i from the neighboring frames within a threshold. We define $j = t + 2(i - t)$, making it twice the distance from t as i is from t. $\gamma \sim \mathcal{U}(t, i)$ is an intermediate time, uniformly sampled between t and i. With the time variables, we formulate our loss:

$$\mathcal{L}_\mathrm{C} = \rho(\mathbf{d}_{t\to i} + \mathbf{d}_{i\to t}) + \rho(\mathbf{d}_{t\to i\to j} - \mathbf{d}_{t\to j}) + \rho(\mathbf{d}_{t\to\gamma\to i} - \mathbf{d}_{t\to i}), \tag{4.13}$$

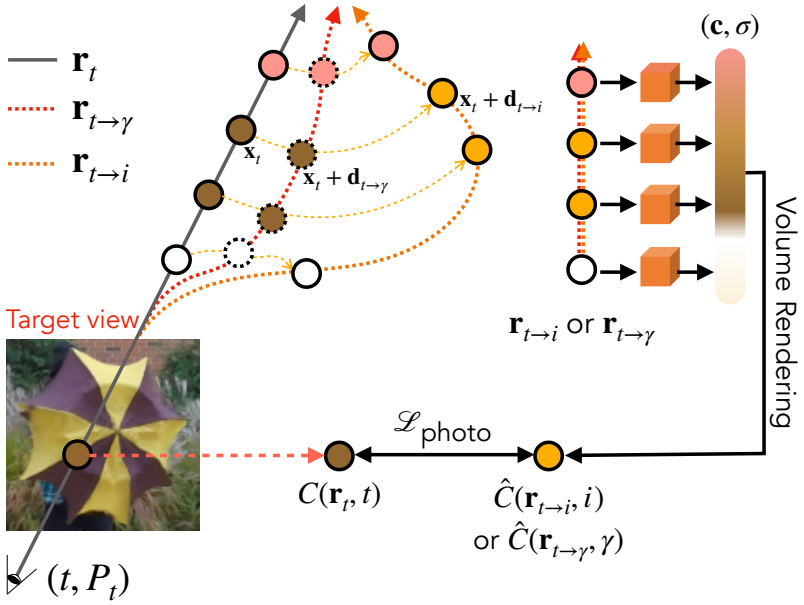


Figure 4.5: **Photometric consistency loss.** During optimization, we deform each ray from a reference time and pose (t, P_t) to different timestamps γ and i. Here, we sample i from the neighboring frame times, and we consequently sample an intermediate timestamp $\gamma \sim \mathcal{U}(t, i)$. Given each timestamp, we can deform a ray using Eq. 4.8. The photometric loss $\mathcal{L}_{\text{photo}}$ is computed based on the color consistency between the deformed ray and the original color, enhancing temporal consistency.

where ρ denotes the generalized chabonnier loss [114]. Here, we abbreviate the displacement $d(\cdot)$ (Eq. 4.8) for conciseness: $\mathbf{d}_{t\to i}$ is shorthand for $d(\mathbf{x}, t, i-t)$, and $\mathbf{d}_{t\to i\to j}$ simplifies $d(\mathbf{x}+\mathbf{d}_{t\to i}, i, j-i)$.

Photometric consistency loss. To enhance temporal coherency, we deform rays into nearby frame times and enhance the consistency among them (Fig. 4.5). Specifically, given a reference ray $\mathbf{r}_t$ at time t, a pixel color $\hat{C}(\mathbf{r}_t, t)$ can be obtained using the volume rendering (Eq. 4.4). In addition to the reference ray, we apply ray deformation based on the predicted kinematic quantities. As in cycle consistency (Eq. 4.13), we sample time i from neighboring frames, and time $\gamma \sim \mathcal{U}(t, i)$ between times t and i. Different from existing approaches [34, 36], we deform rays with kinematic quantities using Taylor approximation. The deforming process results in two deformed rays $\mathbf{r}_{t\to i}$ and $\mathbf{r}_{t\to\gamma}$, and two rendered colors $\hat{C}(\mathbf{r}_{t\to i}, i)$ and $\hat{C}(\mathbf{r}_{t\to\gamma}, \gamma)$, correspondingly. Based on the deformed rays, we apply the photometric consistency loss, comparing the rendered color with the ground-truth color. The loss function is formulated as:

$$\begin{aligned}\mathcal{L}_{\text{photo}} = &\|\hat{C}(\mathbf{r}_t, t) - C(\mathbf{r}_t, t)\|_2^2 \\ &+ \lambda\alpha(\mathbf{r}_t)\rho(\hat{C}(\mathbf{r}_{t\to i}, i) - C(\mathbf{r}_t, t)) \\ &+ \lambda\alpha(\mathbf{r}_t)\rho(\hat{C}(\mathbf{r}_{t\to\gamma}, \gamma) - C(\mathbf{r}_t, t)),\end{aligned} \tag{4.14}$$

where λ is a loss weight for the temporal consistency, and $\alpha(\mathbf{r}_t)$ is the sum of rendering weights of the dynamic densities, representing the dynamic likelihood of the ray $\mathbf{r}_t$.

Final objective. Our final objective function, denoted as $\mathcal{L}$, is the weighted sum of the various loss terms discussed previously. To keep things simple with the variety of loss functions, we symbolize the losses to $\mathcal{L}_{\text{photo}}$, $\mathcal{L}_{\text{kinematic}}$, and $\mathcal{L}_{\text{reg}}$. This results in the final loss:

$$\mathcal{L} = \mathcal{L}_{\text{photo}} + \mathcal{L}_{\text{kinematic}} + \mathcal{L}_{\text{reg}}. \tag{4.15}$$

Here, $\mathcal{L}_{\text{kinematic}}$ is the weighted sum of the kinematic integrity $\mathcal{L}_{\text{K}}$, transport $\mathcal{L}_{\text{T}}$, and rigidity $\mathcal{L}_{\text{R}}$. The additional regularization $\mathcal{L}_{\text{reg}}$ includes the skewed entropy $\mathcal{L}_b$, the cycle consistency $\mathcal{L}_{\text{C}}$, total-variation loss [110, 90], distortion loss [99], monocular depth supervision [34], normal regularization [115, 100] and optical flow supervision [34].

4.3 Experiments

4.3.1 Dataset

We use the NVIDIA dynamic view synthesis dataset (NDVS) [116] to evaluate the effectiveness of our model. For the monocular setup, we adopt the preprocessing steps proposed by NSFF [34] and DynamicNeRF [36]. This dataset has 8 scenes and is captured with 12 synchronized cameras. When used for a monocular video reconstruction task, the dataset shows relatively low effective multi-view factors (EMFs) [91], compared to other monocular datasets; this means that the dataset is more challenging to reconstruct due to the less camera motion and faster-moving objects.

Several different training and evaluation sequences for the monocular task have been proposed. In this paper, we use three different sequences.

24-frames. In this sequence, a scene is configured with 24 training frames, and 264 test frames from unseen multi-views. For a fair comparison, we utilize the dynamic mask, optical flow, and monocular depth, extracted by the same model with the prior work [34]. We resize the height of all images and input data to 288. In this protocol, we additionally report the quality in terms of dynamic regions using the pixel mask provided in the dataset suite.

24-frames-sparse. To test the novel time synthesis, we reduce the 24-frames sequence into 12 frames by sampling the even-numbered frames; we use the unseen inter-frames for testing.

12-frames. In this sequence, each scene has 12 training and 12 testing samples. We use the preprocessed dynamic mask, optical flow, and monocular depth downloadable from the code repository of DynamicNeRF [36].

4.3.2 Implementation Details

Fields configuration. Our radiance and kinematic fields are configured with HexPlane [90] and TensoRF [110]. After features are sampled from the point queries, small MLP decoders process the features to yield colors and kinematic outputs. For density outputs, we use a linear layer following the practices in [90, 110]. We use the coarse-to-fine strategy to learn the voxels properly, where we upsample each voxel at 100; 1,000; 2,500; 5,000; 10,000; 20,000; 30,000 steps by the logarithmic voxel growth algorithm [110]. We set the initial number of voxels to range from 16,000 to 32,768 depending on the datasets, and we upscale the number to 27,000,000 ($300 \times 300 \times 300$) at the last stage of training. For the maximum motion order, we use 3 (jerk) for the NDVS 24-frame sequences and 2 (acceleration) for the NDVS 12-frame sequences.

Training. We use 70,000 steps to optimize our models, which takes about 7 hours with one NVIDIA V100 or A100 GPU. We initialize the static field for 5,000 steps, taking approximately 5 minutes. After finishing the initialization of the static field, we jointly train the fields: dynamic, static, and kinematic fields. During the joint training, we lower the learning rate of the static field by the factor of 0.1 at predefined iterations to prevent dynamic objects from being portrayed as static objects.

Table 4.1: **Ablation study** on Balloon1 scene of the NDVS dataset (24-frames-sparse).

	Novel Times						Seen Times					
	Full			Dynamic Only			Full			Dynamic Only		
Methods	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NSFF [34]	13.18	0.146	0.603	14.21	0.201	0.574	12.49	0.153	0.499	13.16	0.202	0.487
HexPlane [90]	19.84	0.668	0.145	18.12	0.419	0.251	18.93	0.649	0.137	17.74	0.429	0.202
Ours	**25.67**	**0.865**	**0.064**	**21.16**	**0.625**	**0.170**	**25.80**	**0.867**	**0.050**	**21.35**	**0.632**	**0.121**
w/o Higher order motion	24.36	0.842	0.077	19.93	0.556	0.200	24.63	0.846	0.061	20.22	0.572	0.145
w/o $\mathcal{L}_{\text{kinematic}}$	24.96	0.853	0.068	20.15	0.568	0.185	25.30	0.858	0.052	20.65	0.594	0.129
w/o Kinematic Integrity	25.46	0.862	0.066	20.78	0.607	0.179	25.57	0.865	0.051	21.02	0.620	0.126
w/o Rigidity	25.11	0.856	0.066	20.40	0.582	0.181	25.43	0.860	0.051	20.82	0.603	0.123

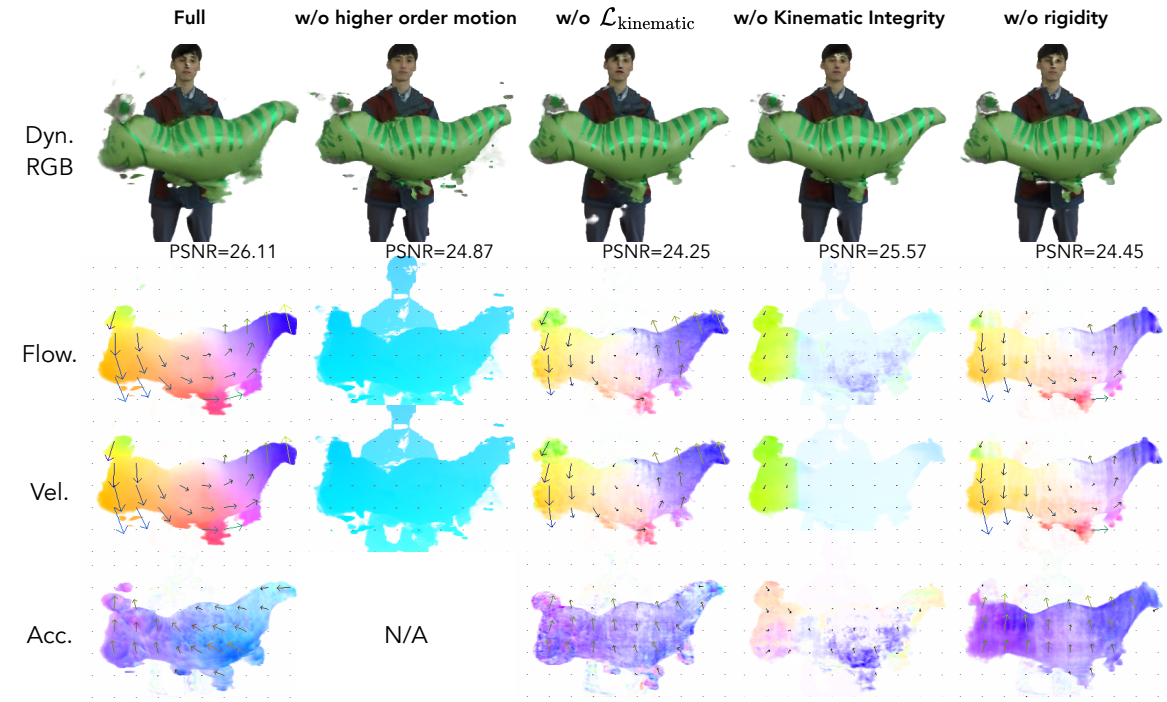


Figure 4.6: **Comparative visualization of kinematic fields.** The first row shows RGB values rendered from our dynamic radiance field, showing PSNR values from a ground truth RGB image. From the second to the last row, we show flow (*i.e.*, displacement), velocity, and acceleration.

Evaluation metrics. We report peak signal-to-noise ratio (PSNR), structural similarity (SSIM), and perceptual similarity (LPIPS). These three metrics have been used as standard in prior studies to evaluate the quality of renderings.

4.3.3 Baseline Model

HexPlane. Since our model is based on tensor-decomposed radiance fields, we compare our models with HexPlane [90]. The baseline model is the dynamic radiance field $\mathcal{F}_{\text{DY}}$, without the static radiance field $\mathcal{F}_{\text{ST}}$ and the kinematic field $\mathcal{F}_{\text{K}}$. For a fair comparison, we use monocular depth supervision both to the baseline model and our models, although the original HexPlane model does not support the supervision from monocular depth. Since the base HexPlane model does not have an auxiliary flow field, we do not use the optical flow supervision and the photometric consistency loss.

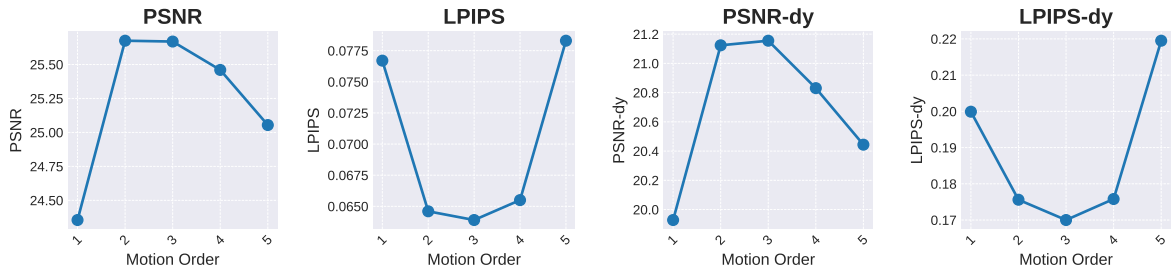


Figure 4.7: **Motion Order**. Reconstruction Accuracy for the Balloon1 Scene (24-frames-sparse): The x-axis of each plot categorizes maximum motion order as follows: velocity (1), acceleration (2), jerk (3), snap (4), and crackle (5). The metric utilized for each plot is specified at the top.

4.3.4 Evaluation of Kinematic Fields

Ablation of kinematic losses. To show the effectiveness of our approach, we conduct a comprehensive ablation study on the Balloon1 scene of the NDVS dataset, focusing on kinematic loss components. In Table 4.1, we show ablative results from seen and unseen frame times. Our findings indicate that the model achieves optimal performance when all components are integrated.

Fig. 4.6 delves into the kinematic field analysis. Omitting higher order motion restricts the model to utilizing only the velocity field, which proves insufficient for capturing non-linear motion. This limitation hinders the model's ability to minimize kinematic losses, leading to a suboptimal flow representation. In the absence of $\mathcal{L}_{\text{kinematic}}$, the radiance field suffers from reduced PSNR scores, which can be attributed to the physically incorrect motion and geometry. Compared to the full model, the model without the rigidity loss results in noisy, non-rigid motion, resulting in incorrect motion and the worse reconstruction accuracy.

Motion order. Our study represents motion through various kinematic quantities. Adjusting the kinematic field to include different orders of motion, we can model our system using just velocity [106, 107], or extend to higher orders – fourth (snap) and fifth (crackle) – by extending the Taylor series (Eq. 4.8) and the kinematic integrity (Eq. 4.9) equations. Fig. 4.7 illustrates the influence of motion order on reconstruction accuracy for the NDVS dataset. As motion order increases, the reconstruction accuracy first improves, peaking at jerk, but further orders degrades the accuracy. We conjecture that the augmentation of model complexity with higher orders of kinematic quantities may render the model more susceptible to overfitting.

4.3.5 Comparison with SOTA models

Quantitative results. NSFF [34] is a dynamic neural radiance field model, which utilizes scene-flow fields for the regularization of a dynamic radiance field. Since NSFF is a neural model, the network training takes about 2 days on two V100 GPUs (96 GPU hours), using the official code by the authors. Note that optimizing a scene using our method takes about 7 hours per scene, which makes ours 12 times faster than NSFF in terms of GPU hours. We compare our results with NSFF [34] in Table 4.2. In the table, our method shows better accuracy in terms of all metrics. The per-scene results are reported in Table 4.4. In addition, we report full results on NDVS (12-frames) in Table 4.3.

Qualitative results. Fig. 4.8 illustrates qualitative results, including the synthesized images, ground truth, and their overlaid images. As shown in the first and third rows, our method improves the visual

Table 4.2: Average results on NDVS (24-frames).

	Full			Dynamic Only		
Methods	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
NeRF [32]*	24.90	0.893	0.098	16.98	0.532	0.314
Luo et al. [117]*	21.37	0.746	0.141	16.97	0.530	0.207
Yoon et al. [116]*	21.78	0.761	0.127	17.34	0.547	0.20
NSFF [34]*	28.19	0.928	0.045	21.91	0.758	0.097
NSFF [34]	28.04	0.927	0.045	21.34	0.740	0.111
Ours	**28.78**	**0.936**	**0.042**	**22.13**	**0.774**	**0.094**

*Results reported in [34]

quality of the rendered views. Additionally, the overlaid images demonstrate the geometrical accuracy of our model, showcasing its better alignment with the ground truth. This enhanced performance is attributed to the kinematics and physics incorporated in our methodology, which is fundamental to the improvement.

4.4 Supplementary Material

In this supplementary material, we provide additional experimental results. In addition, we describe details of our method including the tensor-decomposition-based architectures, coordinate systems, and detailed training strategies.

4.5 Additional Results

In this section, we show additional results, which can supplement the main paper's results.

4.5.1 Ablation on Different Scenes

Quantitative results. In Table 4.5, we report additional ablation study results on the NDVS (24-frames-sparse) scenes. In most scenes, our full model consistently surpasses alternative approaches. Especially, the effectiveness of our method is most evident within dynamic regions (*i.e.*, Dynamic Only metrics) of the scenes, showing meaningful gaps over the ablated models.

Qualitative results. In Fig. 4.9-4.12, we visualize the rendered RGB maps and velocity fields inferred from ablated models. In the figure, our full model not only shows better visual quality, but also presents

Table 4.3: **Quantitative comparison** on NDVS (12-frames).

PSNR↑/LPIPS↓	Jumping	Skating	Truck	Umbrella	Balloon1	Balloon2	Playground	Average
NeRF [32]*	20.99/0.305	23.67/0.311	22.73/0.229	21.29/0.440	19.82/0.205	24.37/0.098	21.07/0.165	21.99/0.250
NSFF [34]*	24.65/0.151	29.29/0.129	25.96/0.167	22.97/0.295	21.96/0.215	24.27/0.222	21.22/0.212	24.33/0.199
DynamicNeRF [36]*	24.68/0.090	32.66/0.035	28.56/0.082	23.26/0.137	22.36/0.104	27.06/0.049	24.15/0.080	<u>26.10</u>/0.082
HyperNeRF [104]*	18.34/0.302	21.97/0.183	20.61/0.205	18.59/0.443	13.96/0.530	16.57/0.411	13.17/0.495	17.60/0.367
RoDynRF [94]*	25.66/ 0.071	28.68/0.040	29.13/0.063	24.26/0.089	22.37/0.103	26.19/0.054	24.96/0.048	25.89/**0.065**
Ours	23.74/0.102	31.89/0.035	28.34/0.074	25.46/0.098	23.72/0.079	26.49/0.055	24.64/0.053	**26.33**/<u>0.071</u>

*Results reported in [94]

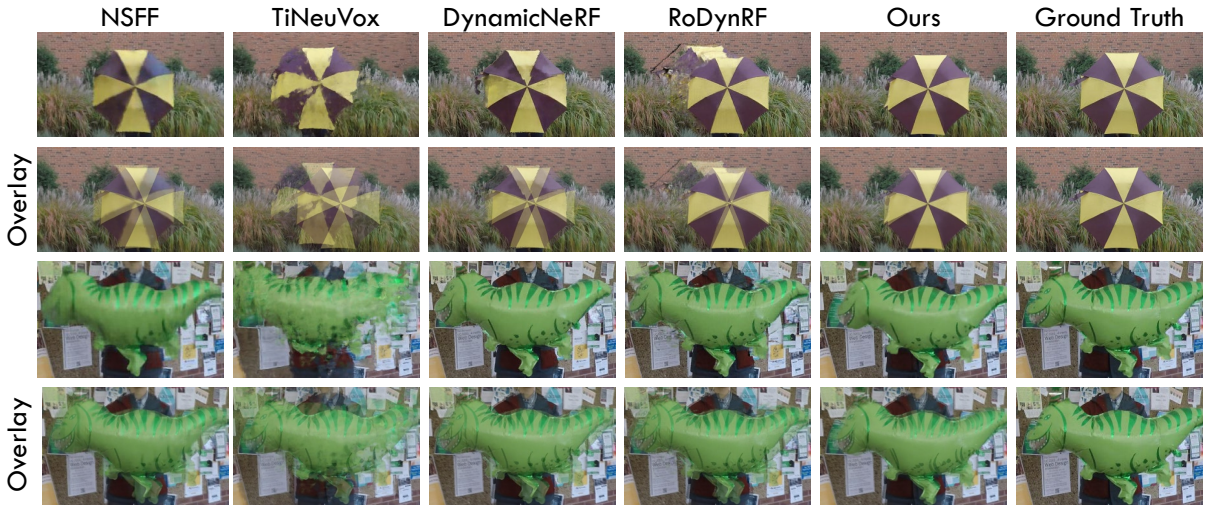


Figure 4.8: **Qualitative results** on NDVS (12-frames). The second and fourth rows illustrate the overlay of the synthesized and ground truth views of the scene.

Table 4.4: **Quantitative comparison** on NDVS (24-frames). Numbers that surpass their counterparts by a margin greater than 5% are highlighted.

method		Playground			Balloon1			Balloon2			Umbrella		
		PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NSFF [34]	full	24.80	0.865	0.056	24.14	0.848	0.064	29.34	0.918	0.034	24.46	0.787	**0.087**
(train: 96 h)	dyn.	19.12	0.671	0.120	18.29	0.475	0.183	21.28	0.670	0.112	17.00	0.444	0.155
Ours	full	**26.65**	**0.911**	**0.042**	**27.26**	**0.893**	**0.043**	29.50	0.916	0.035	**26.24**	**0.792**	0.098
(train: 7 h)	dyn.	18.75	0.658	0.122	**21.85**	**0.654**	**0.117**	22.23	0.692	**0.104**	**22.25**	**0.695**	**0.097**
method		Jumping			DynamicFace			Skating			Truck		
		PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NSFF [34]	full	26.84	0.893	0.053	26.993	0.947	0.022	**35.10**	**0.969**	**0.018**	32.70	0.940	**0.022**
(train: 96 h)	dyn.	19.60	0.616	0.130	25.311	0.886	0.028	**22.28**	**0.724**	0.109	27.86	0.840	**0.054**
Ours	full	26.55	0.895	**0.049**	**29.15**	0.967	**0.011**	32.33	0.959	0.024	32.57	0.930	0.033
(train: 7 h)	dyn.	19.16	0.603	**0.114**	25.68	0.901	**0.018**	19.58	0.626	0.108	27.54	0.854	0.068

the most smooth and consistent velocity field. Notably, learning kinematic fields without the kinematic integrity loss results in highly inconsistent flow field. In Fig. 4.13, we qualitatively compare ours with existing works.

4.5.2 Motion Extrapolation

We have implemented an extrapolation application utilizing the kinematic field in Fig. 4.14. This application leverages displacement calculations derived from the kinematic field combined with a pixel splatting technique [10]. Our comparison illustrates that motion extrapolation based on our kinematic field and proposed regularization yields higher accuracy compared to the counterparts, showcasing its potential for motion prediction.

4.6 3D and 4D Volume Structures

Our method is configured with the dynamic radiance field $\mathcal{F}_{\text{DY}}$, the static radiance field $\mathcal{F}_{\text{ST}}$, and the kinematic field $\mathcal{F}_{\text{K}}$. The 4D fields $\mathcal{F}_{\text{DY}}$ and $\mathcal{F}_{\text{K}}$ require $O(N^3TF)$ space, where N is the spatial, T is the temporal resolutions, and F is the feature size for each voxel. Similarly, the 3D field $\mathcal{F}_{\text{ST}}$ which does

Table 4.5: **Ablation study** on the NDVS dataset (24-frames-sparse).

(a) Balloon1

	Novel Times						Seen Times					
	Full			Dynamic Only			Full			Dynamic Only		
Methods	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NSFF [34]	13.18	0.146	0.603	14.21	0.201	0.574	12.49	0.153	0.499	13.16	0.202	0.487
HexPlane [90]	19.84	0.668	0.145	18.12	0.419	0.251	18.93	0.649	0.137	17.74	0.429	0.202
Ours	**25.67**	**0.865**	**0.064**	**21.16**	**0.625**	**0.170**	**25.80**	**0.867**	**0.050**	**21.35**	**0.632**	**0.121**
w/o Higher order motion	24.36	0.842	0.077	19.93	0.556	0.200	24.63	0.846	0.061	20.22	0.572	0.145
w/o $\mathcal{L}_{\text{kinematic}}$	24.96	0.853	0.068	20.15	0.568	0.185	25.30	0.858	0.052	20.65	0.594	0.129
w/o Kinematic Integrity	25.46	0.862	0.066	20.78	0.607	0.179	25.57	0.865	0.051	21.02	0.620	0.126
w/o Rigidity	25.11	0.856	0.066	20.40	0.582	0.181	25.43	0.860	0.051	20.82	0.603	0.123

(b) DynamicFace

	Novel Times						Seen Times					
	Full			Dynamic Only			Full			Dynamic Only		
Methods	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NSFF [34]	10.95	0.131	0.539	8.98	0.064	0.628	10.77	0.120	0.545	8.88	0.055	0.636
HexPlane [90]	12.11	0.265	0.471	14.38	0.431	0.397	11.39	0.218	0.491	13.74	0.442	0.395
Ours	26.47	**0.896**	0.051	**25.22**	**0.893**	**0.028**	25.20	**0.881**	0.060	**24.74**	**0.887**	**0.031**
w/o Higher order motion	**26.49**	**0.896**	0.051	25.03	0.882	0.031	**25.22**	**0.881**	0.059	24.47	0.874	0.034
w/o $\mathcal{L}_{\text{kinematic}}$	26.24	0.895	0.052	24.34	0.867	0.034	24.98	0.880	0.060	23.87	0.859	0.037
w/o Kinematic Integrity	26.24	0.898	**0.050**	24.65	0.871	0.032	25.02	0.884	**0.057**	24.08	0.861	0.036
w/o Rigidity	26.20	0.897	0.051	24.73	0.878	0.032	24.95	0.883	0.059	24.19	0.870	0.035

(c) Playgruond

	Novel Times						Seen Times					
	Full			Dynamic Only			Full			Dynamic Only		
Methods	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NSFF [34]	12.54	0.113	0.581	11.58	0.072	0.629	12.15	0.117	0.515	11.41	0.067	0.530
HexPlane [90]	14.79	0.286	0.341	13.40	0.191	0.379	13.80	0.230	0.341	12.66	0.163	0.360
Ours	**24.64**	**0.832**	**0.068**	**18.24**	**0.621**	**0.141**	**23.82**	**0.811**	**0.074**	**18.14**	**0.621**	**0.133**
w/o Higher order motion	24.42	0.829	**0.068**	17.63	0.576	0.149	23.58	0.808	0.075	17.45	0.569	0.144
w/o $\mathcal{L}_{\text{kinematic}}$	24.38	0.830	**0.068**	17.61	0.595	0.153	23.57	0.809	0.075	17.49	0.596	0.147
w/o Kinematic Integrity	24.52	0.830	0.069	17.93	0.605	0.150	23.69	0.809	0.075	17.81	0.605	0.140
w/o Rigidity	24.40	0.830	**0.068**	17.64	0.597	0.146	23.62	0.810	0.075	17.65	0.603	0.137

(d) Skating

	Novel Times						Seen Times					
	Full			Dynamic Only			Full			Dynamic Only		
Methods	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NSFF [34]	18.18	0.411	0.418	12.41	0.136	0.608	17.20	0.397	0.364	11.75	0.124	0.513
HexPlane [90]	22.38	0.783	0.151	12.43	0.145	0.352	21.77	0.769	0.154	12.12	0.140	0.357
Ours	**30.99**	**0.941**	0.033	**18.60**	**0.563**	**0.165**	**31.91**	**0.943**	0.029	**20.57**	**0.671**	**0.107**
w/o Higher order motion	30.89	0.940	**0.032**	18.45	0.536	0.161	31.36	0.940	**0.028**	19.70	0.610	0.109
w/o $\mathcal{L}_{\text{kinematic}}$	5.88	0.463	0.698	3.73	0.138	0.857	5.88	0.463	0.698	3.74	0.141	0.859
w/o Kinematic Integrity	26.04	0.916	0.059	14.59	0.316	0.267	26.50	0.918	0.055	14.94	0.372	0.228
w/o Rigidity	29.72	0.935	0.036	16.98	0.425	0.198	30.05	0.935	0.032	17.78	0.492	0.150

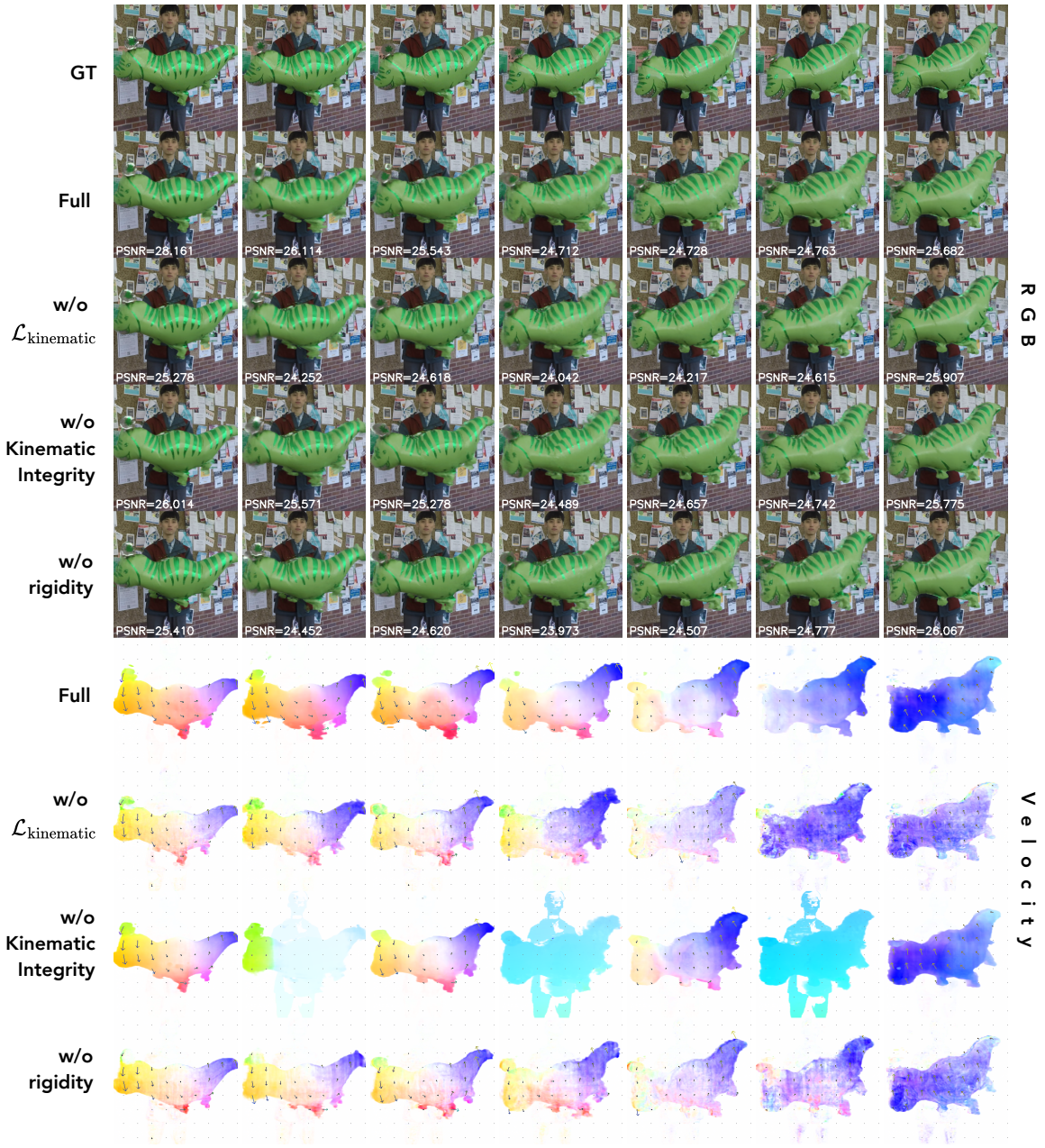


Figure 4.9: **Qualitative comparison** on Balloon1 scene of the NDVS dataset (24-frames-sparse)

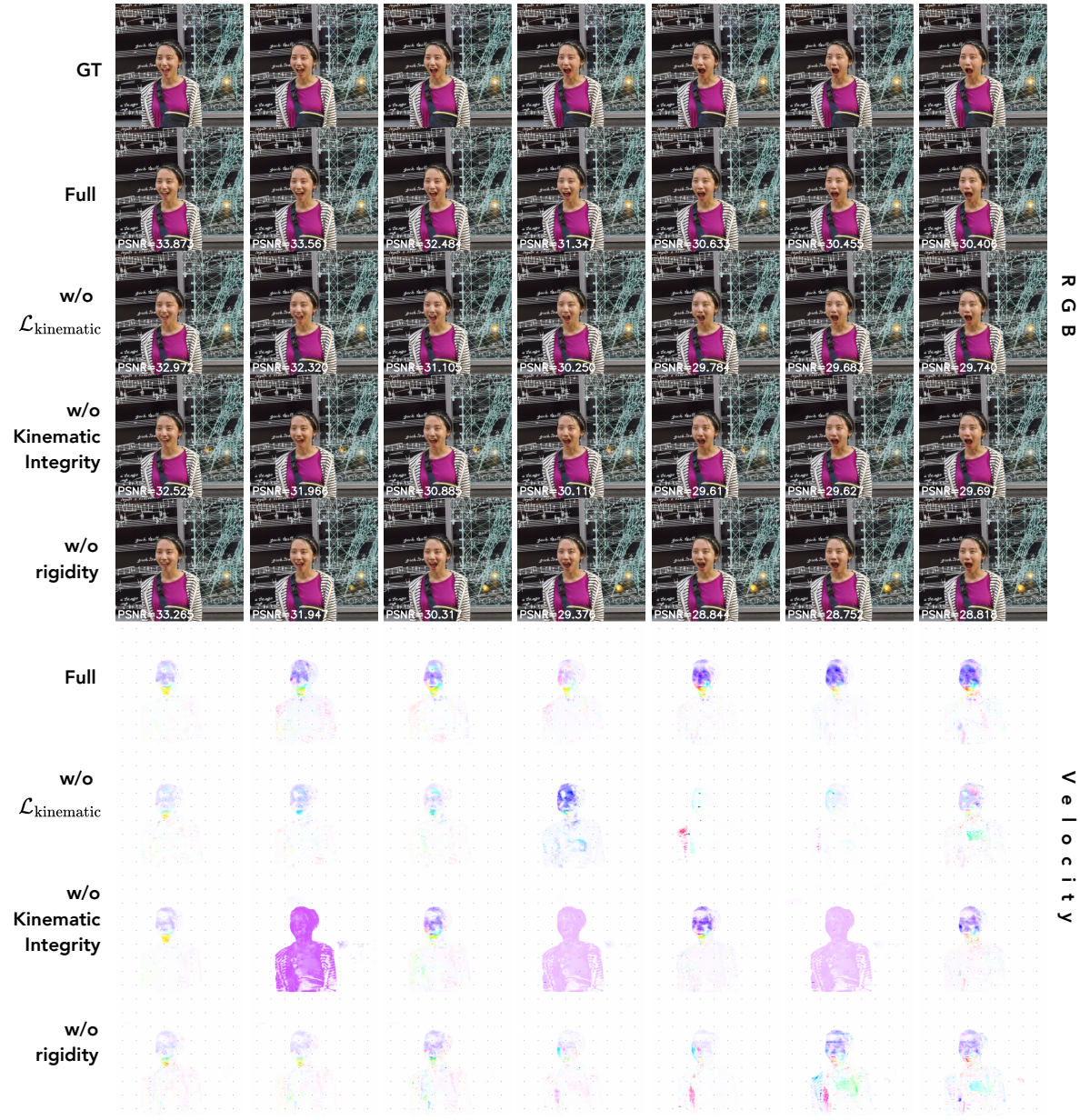


Figure 4.10: **Qualitative comparison** on DynamicFace scene of the NDVS dataset (24-frames-sparse)

Figure 4.11: **Qualitative comparison** on Playground scene of the NDVS dataset (24-frames-sparse)

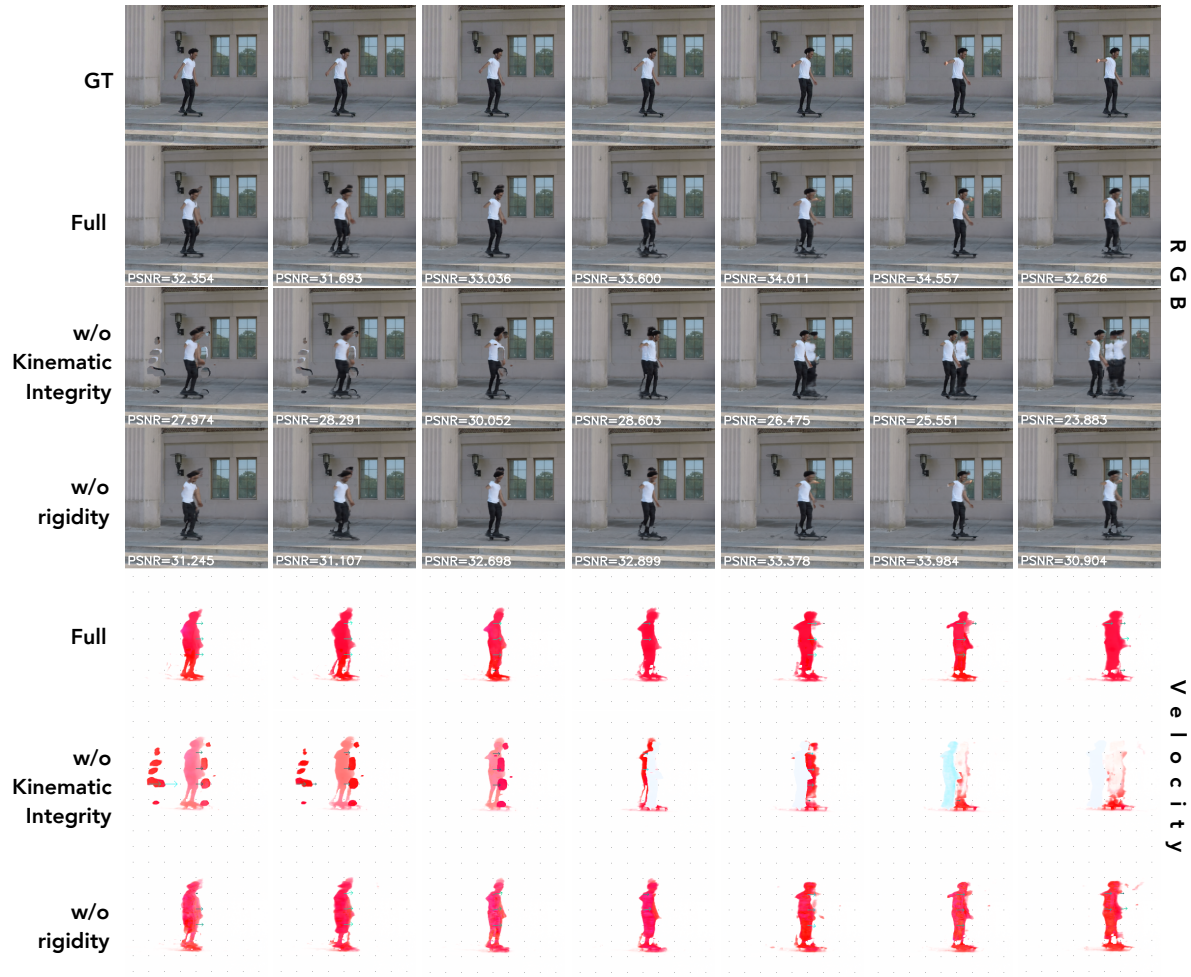


Figure 4.12: **Qualitative comparison** on Skating scene of the NDVS dataset (24-frames-sparse). Note that the w/o $\mathcal{L}_{\text{kinematic}}$ model did not converge in this scene.

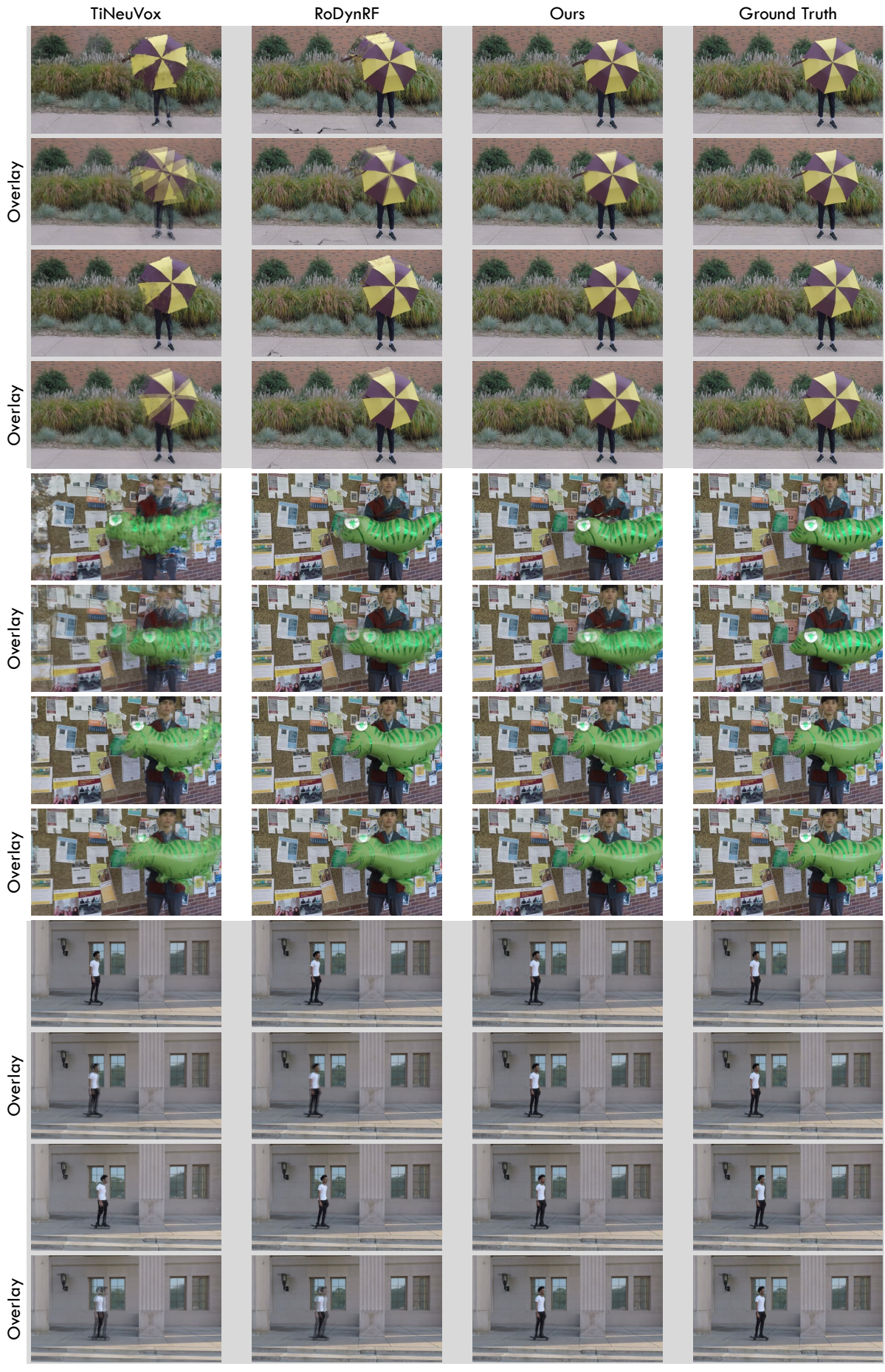
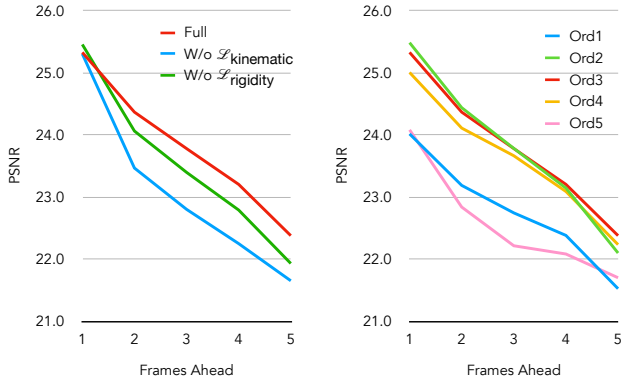
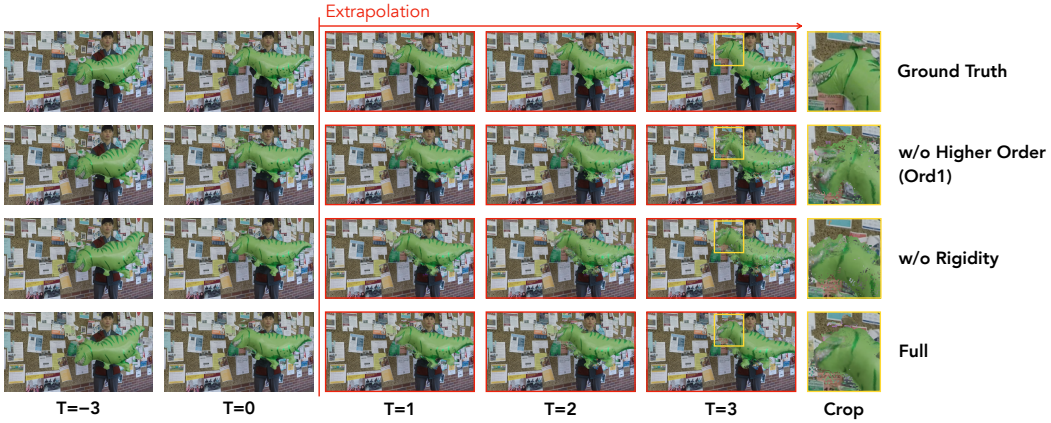


Figure 4.13: **Qualitative comparison** on NDVS dataset (12-frames)



(a) Accuracy of extrapolated frames (Balloon1 scene)



(b) Extrapolated frames

Figure 4.14: Extrapolation using kinematic fields. In this example application, we combine 2D displacements inferred from our kinematic field and a pixel splatting technique [10].

not depend on the time varible t, requires $O(N^3F)$. As stated in the HexPlane paper [90] storing the whole volume in a naïve data structure takes about 48GB memory when $N = 512$ and $T = 32$.

Thus, we use a tensor-decomposition-based structure, HexPlane [90], to compress the large space into a more compact representation so that we can make the framework feasible. Our 3D feature volume $V_{\text{3D}} \in \mathbb{R}^{XYZF}$ and 4D feature volume $V_{\text{4D}} \in \mathbb{R}^{XYZTF}$ are defined as:

$$V_{\text{3D}} = \sum_{r=1}^{R_1} \mathbf{M}_r^{XY} \circ \mathbf{v}_r^Z \circ \mathbf{v}_r^1 + \sum_{r=1}^{R_2} \mathbf{M}_r^{XZ} \circ \mathbf{v}_r^Y \circ \mathbf{v}_r^2 + \sum_{r=1}^{R_3} \mathbf{M}_r^{YZ} \circ \mathbf{v}_r^X \circ \mathbf{v}_r^3, \tag{4.16}$$

$$V_{\text{4D}} = \sum_{r=1}^{R_1} \mathbf{M}_r^{XY} \circ \mathbf{M}_r^{ZT} \circ \mathbf{v}_r^1 + \sum_{r=1}^{R_2} \mathbf{M}_r^{XZ} \circ \mathbf{M}_r^{YT} \circ \mathbf{v}_r^2 + \sum_{r=1}^{R_3} \mathbf{M}_r^{YZ} \circ \mathbf{M}_r^{XT} \circ \mathbf{v}_r^3, \tag{4.17}$$

where $\circ$ is the outer product, $\mathbf{M}_r^{AB} \in \mathbb{R}^{AB}$ is a plane representing A-B dimensions, $\mathbf{v}_r^C \in \mathbb{C}^C$ is the vector along the C-axis, $\mathbf{v}_r^n \in \mathbb{R}^F$ is the vector along the F-axis, and R_n is the number of low-rank components. Note that, we adopt the notations from the HexPlane paper [90].

When querying a coordinate $(x, y, z, t, \theta, \phi)$, we sample F-dimensional feature from each volume; here, we leverage multi-resolution sampling method [94] to sample features from the volumes. We configure each radiance field with two volumes representing density and RGB, respectively. For density fields, we employ an MLP-free design that does not depend on a ray direction (θ, ϕ). On the other hand, we use a tiny MLP which takes as inputs the sampled feature and the ray direction for the RGB. These designs are mostly equivalent to the default configuration of HexPlane [90]. For the kinematic field, we use a

tiny MLP to produce the kinematic quantities, but without the ray direction as an input; this makes the motion and density of particles in a scene be independent of a ray direction.

Table 4.6: Number of low-rank components in HexPlane (or TensoRF) structures.

	X-Y	Y-Z	X-Z	Z-T	Y-T	X-T
Dy. Density	16	4	4	16	4	4
Dy. RGB	48	12	12	48	12	12
St. Density	16	4	4	$16^{\dagger}$	$4^{\dagger}$	$4^{\dagger}$
St. RGB	48	12	12	$48^{\dagger}$	$12^{\dagger}$	$12^{\dagger}$
Kinematic	32	16	16	32	16	16

†: VM-factorization [110].

Plane and vector sizes. We report the number of low-rank components used for HexPlane (or TensoRF) structures in Table 4.6. Note that the number of low-rank components defines the feature dimension of each plane; if the X-Y plane has 48 components, then the shape of the plane becomes $X \times Y \times 48$. We use `multiply` followed by `concat` for the feature fusion design since the combination shows better results than alternative designs [90].

4.7 Details on Kinematic Fields

4.7.1 Coordinate Systems

Since a volume-rendering method requires sampling color and density from a bounded space, learning a radiance field model from unbounded scenes might produce undesired artifacts in the regions outside the sampling boundaries. Thus, there have been various methods to solve this problem [32, 101, 118]. One common solution for unbounded forward-facing scenes is using the normalized device coordinates (NDC) [32], where a camera frustum with the unbounded z-axis is transformed into a $[-1, 1]^3$ cube. Further details regarding the conversion from NDC to world space are provided in Sec. 4.8.

Thus, we use NDC to learn the radiance fields for unbounded forward-facing scenes. However, the physics-related elements should be computed in the world space, rather than in the NDC space. Therefore, the kinematic quantities are represented in the world space in our method.

Let $g(\cdot)$ be a transformation which converts a coordinate in NDC to world:

$$g(\mathbf{x}_{\text{ndc}}) = \mathbf{x}_{\text{world}}, \tag{4.18}$$

where $\mathbf{x}_{\text{ndc}}$ in NDC corresponds to $\mathbf{x}_{\text{world}}$ in world space. Then, our kinematic field $\mathcal{F}_{\text{K}}$ takes a NDC and produce outputs in world space:

$$\mathcal{F}_{\text{K}}(\mathbf{x}_{\text{ndc}}) = (\mathbf{v}_{\text{world}}, \mathbf{a}_{\text{world}}, \mathbf{j}_{\text{world}}), \tag{4.19}$$

where $\mathbf{v}_{\text{world}}$, $\mathbf{a}_{\text{world}}$, and $\mathbf{j}_{\text{world}}$ are velocity, acceleration, and jerk defined in the world space. To compute the displacement in world space, it is trivial to use Eq. 8, resulting in a displacement in the world space. On the other hand, in the case of computing the photometric consistency loss (Eq. 15) and the cycle consistency loss (Eq. 14), we need to compute displacement in NDC. A displacement in NDC can be computed as:

$$d_{\text{ndc}}(\mathbf{x}_{\text{ndc}}, t, \Delta t) = g^{-1}(g(\mathbf{x}_{\text{ndc}}) + d(g(\mathbf{x}_{\text{ndc}}), t, \Delta t)) - \mathbf{x}_{\text{ndc}}, \tag{4.20}$$

where $g(\cdot)$ transforms an NDC into the world space, and $g^{-1}(\cdot)$ is the inverse of $g(\cdot)$.

4.7.2 Computing Numerical Gradients

As noted in the main paper, numerical gradients can be computed as:

$$\frac{\partial f(x)}{\partial x} \approx \frac{f(x+\epsilon) - f(x-\epsilon)}{2\epsilon}, \tag{4.21}$$

where $\epsilon > 0$ is a small offset. Regarding this numerical method, we need to consider two factors: how to set ϵ and how to deal with the NDC system.

How to set ϵ. Since we leverage an explicit feature representation for radiance and kinematic fields, our voxel structure has a certain resolution in each phase of training. Hence, we can leverage the voxel resolution to decide ϵ [111].

When dimensions of the radiance and kinematic fields are X,Y, Z, and T we set ϵ to $\frac{2\lambda}{X}$ for the spatial axis and $\frac{2\lambda}{T}$ for the temporal axis, where λ is the constant controlling the scale. Note that we increase the spatial resolution from $22 \times 22 \times 22$ to $300 \times 300 \times 300$; during the upsampling process, ϵ decreases accordingly. In all experiments, we use $\lambda = 0.2$.

How to deal with NDC. Since the voxel features are defined in the NDC space, we compute initial ϵ in the NDC voxel space. However, to compute Jacobians of the kinematic quantities we need to have ϵ in the world space to correctly compute the gradient (Eq. 4.21). Thus, we convert ϵ from NDC to ϵ_{world} in world space by:

$$\epsilon_{\text{world}} = 0.5 \cdot \|g(x+\epsilon, y, z) - g(x-\epsilon, y, z)\|_2, \tag{4.22}$$

where $g(\cdot)$ is the transformation from NDC to world, and we will omit y, z for concise notations in the following equation. Having ϵ_{world} computed, we can rewrite Eq. 4.21 to:

$$\frac{\partial f(x)}{\partial x} \approx \frac{f(g^{-1}(g(x) + \epsilon_{\text{world}})) - f(g^{-1}(g(x) - \epsilon_{\text{world}}))}{2\epsilon_{\text{world}}}, \tag{4.23}$$

where g^{-1} is the world to NDC transformation. Note that we can extend Eq. 4.23 into y and z axis trivially. For t-axis, it is not required to convert the space from NDC to world; we can simply use Eq. 4.21 to compute derivatives with respect to time t.

4.8 Normalized Device Coordinates

In neural radiance fields, reconstructing scenes without considering unbounded regions can generate undesirable artifacts. The normalized device coordinate (NDC) is proposed to deal with a scene unbounded in the z-axis (*i.e.*, infinite depth), which is effective in covering mostly forward-facing scenes. Since a short monocular clip is usually forward-facing, NDC has been frequently utilized for 4D reconstruction from a monocular video [34, 94, 36]. Similarly, we utilize NDC for the fields defined in our framework.

In this section, we describe about the conversion between the world coordinates and the normalized device coordinates (NDC).

4.8.1 Projection Matrix

In this description, we follow the notation given in the supplementary material of the NeRF paper [32]. The conversion between the two spaces is defined using the perspective projection matrix with the near

plane n, the far plane f, the right bound r, and the left bound l. The projection matrix M is defined by:

$$M = \begin{pmatrix} \frac{n}{f} & 0 & 0 & 0 \\ 0 & \frac{n}{t} & 0 & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}. \tag{4.24}$$

The projection M projects a homogeneous coordinate $(x, y, z, 1)$ in world space to NDC. The projected coordinates in NDC are mapped to a $[-1, 1]^3$ cube.

$$\begin{pmatrix} \frac{n}{f} & 0 & 0 & 0 \\ 0 & \frac{n}{t} & 0 & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{4.25}$$

$$\propto \begin{pmatrix} \frac{n}{r}\frac{x}{-z} \\ \frac{n}{t}\frac{y}{-z} \\ \frac{f+n}{f-n} - \frac{2fn}{f-n}\frac{1}{-z} \\ 1 \end{pmatrix} = \begin{pmatrix} x_{\text{ndc}} \\ y_{\text{ndc}} \\ z_{\text{ndc}} \\ 1 \end{pmatrix}, \tag{4.26}$$

where the coordinate (x, y, z) in world space is converted into the coordinate $(x_{\text{ndc}}, y_{\text{ndc}}, z_{\text{ndc}})$ in NDC space. Note that, it is trivial to inverse the process to convert a homogeneous coordinate in NDC into world space.

4.8.2 Density Transformation

Following the notation used in Sec. 4.7.2, let $g(\cdot)$ be the NDC to world transformation, where a point $\mathbf{x}_{\text{ndc}}$ in the NDC is converted to $\mathbf{x} = g(\mathbf{x}_{\text{ndc}})$ in world space.

In the transport loss (Eq. 12), we need to compute $\nabla\sigma$, with respect to the spatial coordinate $\mathbf{x}$ in world space. However, if we formulate the rendering equation in NDC, the density value is referenced in NDC space, not in world space. Thus, we need to define a density transformation function $\sigma = g_d(\sigma_{\text{ndc}}, \mathbf{x}_{\text{ndc}})$, which transforms density σ_{ndc} defined at $\mathbf{x}_{\text{ndc}}$ into the density σ in world space.

Given two different coordinate systems A and B, a small volume element V_{A} in coordinate system A and the corresponding volume element V_{B} in coordinate system B can be expressed as:

$$V_{\text{B}} = |\det(J_f(\mathbf{x}_{\text{A}}))| V_{\text{A}}, \tag{4.27}$$

where $\det(J_f(\mathbf{x}_{\text{A}}))$ is the determinant of the Jacobian matrix J_f of the coordinate transfomation f at point $\mathbf{x}_{\text{A}}$.

Thus, we formulate density transformation g_d as:

$$g_d(\sigma_{\text{ndc}}, \mathbf{x}_{\text{ndc}}) = \frac{C\sigma_{\text{ndc}}}{|\det(J_g(\mathbf{x}_{\text{ndc}}))|} = \sigma, \tag{4.28}$$

where C is a constant. Note that relationship between density ρ, mass m, and volume V is given by the formula $\rho = m/V$. Assuming equal mass in two spaces, we can use Eq. 4.28.

4.9 Training Details

4.9.1 Fields

As mentioned in the main paper, our radiance and kinematic fields consist of HexPlane [90] and TensoRF [110]. The feature sizes of the planes and the vectors are specified in Table 4.6.

For RGB and kinematic outputs, we use a tiny 3-layer MLP; we use 64 dimension and 128 dimension in hidden layers for RGB and kinematic fields, respectively. When we concatenate a direction vector, we do not use positional encoding; we use the ℓ^2-normalization of a ray direction vector as an input to an MLP.

During training, we upsample the size of hexplanes at 100; 1,000; 2,500; 5,000; 10,000; 20,000; 30,000 steps by the logarithmic voxel growth algorithm [110]. The initial size is from 16,000 to 32,768 depending on the datasets, and we increase the size to 27,000,000 ($300 \times 300 \times 300$). Similarly, the dimension of time-axis grows from $N-1$ to $2N$, where N is the number of frames in the monocular video.

4.9.2 Losses

Total variation (TV) loss. For the total-variation (TV) loss [110], we use 0.001 as the loss weight. Since using TV loss for the kinematic fields often results in an equilibrium where all feature values are identical, we do not leverage the TV loss for the kinematic field.

Photometric loss. In the photometric loss (Eq. 14) and the cycle loss (Eq. 13), we need to define the maximum hop we sample i within. Like the logarithmic voxel growth algorithm [110], we increase the maximum hop from a two-frame gap to a three-frame gap during the training phase.

Trajectory smoothness. Since higher-order kinematic quantities can generate a jerky trajectory, we regularize on the predicted higher-order terms and partial derivatives using an L2 penalty regularizer:

$$\mathcal{L}_S = \|\mathbf{a}\|_2^2 + \|\mathbf{j}\|_2^2 + \|\partial\mathbf{v}/\partial t + \mathbf{v}\cdot\nabla\mathbf{v}\|_2^2 + \|\partial\mathbf{a}/\partial t + \mathbf{v}\cdot\nabla\mathbf{a}\|_2^2. \tag{4.29}$$

In addition to this loss, we start training with the first order, and gradually add higher order kinematic quantities to prevent jerky trajectories.

Initialization w/o kinematic loss. For stable training, we apply the kinematic loss from the 1000 th iteration; before the 1000 th, we only utilize $\mathcal{L}_{\text{photo}}$ and $\mathcal{L}_{\text{reg}}$ in the total loss $\mathcal{L}$ (Eq. 15).

Chapter 5. Summary and Conclusion

This dissertation explores various aspects of video motion learning across different types of motion representations. Initially, we discussed deep optical flow estimation, focusing on the lack of ground truth required for supervised learning. Thus, we introduced the unsupervised approach where a deep feature similarity helps a network better learn flows and features from a sequence of images in the unsupervised way. We observed that, using the feature separation loss, the flows are updated to make the fused similarity more discriminative, while suppressing uncertain flows and reinforcing clear flows. The experiments showed that the proposed method achieves competitive results in both qualitative and quantitative evaluation. The promising results confirm that, without labels, the self-supervised features can be used to improve itself.

Second, we presented a self-supervision strategy for semi-supervised optical flow, which is simple, yet effective. Our flow supervisor module supervises a student model, which is effective in the semi-supervised optical flow setting where we have no or few samples in a target domain. Our method outperforms various self-supervised baselines, shown by the empirical study. In addition, we showed that our semi-supervised method can improve the state-of-the-art supervised models by exploiting additional unlabeled datasets. The unsupervised and semi-supervised approaches have proven the self-supervised deep features can be effectively used to improve video motion learning.

Last, we extended our video motion learning into 3D space, using physics-based priors. Incorporated with dynamic radiance fields, kinematic fields are modeled based on kinematics. Our novel formulation enables the physics-based regularizers, which are effective in filling the gap between 2D input video and real-world dynamics.

Overall, our research introduces highly effective methods for utilizing self-supervised deep features and physics-based priors in video motion learning. Our work primarily targets low-level perception, which forms a robust foundation for high-level video understanding and contributes to the broader goal of achieving artificial general intelligence. We anticipate that our findings will pave the way for developing systems that are capable of understanding motion in the world, and improving by themselves. We believe these capabilities potentially help an AI model approach human-level intelligence.

Declaration of Inclusion

I, Woobin Im, hereby declare that the following chapters of this dissertation are based on my previously published or submitted work:

- Chapter 2: Unsupervised Learning of Optical Flow

  [68] **Woobin Im**, Tae-Kyun Kim, and Sung-Eui Yoon, "Unsupervised learning of optical flow with deep feature similarity", in *The European Conference on Computer Vision (ECCV)*, 2020.

- Chapter 3: Semi-Supervised Learning of Optical Flow

  [119] **Woobin Im**, Sebin Lee, and Sung-eui Yoon, "Semi-supervised learning of optical flow by flow supervisor", in *The European Conference on Computer Vision (ECCV)*, 2022.

- Chapter 4: Learning Kinematic Fields for Better Reconstruction of Dynamic Radiance Fields

  [120] **Woobin Im**, Geonho Cha, Sebin Lee, Jumin Lee, Ju-hyeong Seon, Dongyoon Wee, and Sung-eui Yoon, "Regularizing dynamic radiance fields with kinematic fields", *Under Review*, 2024.

This dissertation as a whole represents my original work, and the previously published materials have been incorporated to provide a comprehensive overview of my research contributions during the course of my PhD studies.

I confirm that the content presented in these chapters has been adapted and integrated into the dissertation. The previously published work has been reformatted to comply with the formatting requirements of this dissertation. I have obtained necessary permissions from the co-authors to include the published material in this dissertation.

Bibliography

[1] Pengpeng Liu, Irwin King, Michael R. Lyu, and Jia Xu, "Ddflow: Learning optical flow with unlabeled data distillation", *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 8770–8777, Jul. 2019.

[2] Deqing Sun, Daniel Vlasic, Charles Herrmann, Varun Jampani, Michael Krainin, Huiwen Chang, Ramin Zabih, William T Freeman, and Ce Liu, "Autoflow: Learning a better training set for optical flow", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10093–10102.

[3] Yohann Cabon, Naila Murray, and Martin Humenberger, "Virtual kitti 2", 2020.

[4] Daniel Kondermann, Rahul Nair, Katrin Honauer, Karsten Krispin, Jonas Andrulis, Alexander Brock, Burkhard Gussefeld, Mohsen Rahimimoghaddam, Sabine Hofmann, Claus Brenner, et al., "The hci benchmark suite: Stereo and flow ground truth with uncertainties for urban autonomous driving", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 19–28.

[5] Pengpeng Liu, Michael Lyu, Irwin King, and Jia Xu, "Selflow: Self-supervised learning of optical flow", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4571–4580.

[6] "Mpi sintel dataset", http://sintel.is.tue.mpg.de/.

[7] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation", in *Computer Vision and Pattern Recognition*, 2016.

[8] Zachary Teed and Jia Deng, "Raft: Recurrent all-pairs field transforms for optical flow", in *European Conference on Computer Vision*. Springer, 2020, pp. 402–419.

[9] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski, "A database and evaluation methodology for optical flow", *International Journal of Computer Vision*, vol. 92, no. 1, pp. 1–31, 2011.

[10] Simon Niklaus and Feng Liu, "Softmax splatting for video frame interpolation", in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 5437–5446.

[11] Karen Simonyan and Andrew Zisserman, "Two-stream convolutional networks for action recognition in videos", in *Advances in neural information processing systems*, 2014, pp. 568–576.

[12] Joao Carreira and Andrew Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset", in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6299–6308.

[13] Josef Zihl, D Von Cramon, and Norbert Mai, "Selective disturbance of movement vision after bilateral brain damage", *Brain*, vol. 106, no. 2, pp. 313–340, 1983.

[14] Rui Xu, Xiaoxiao Li, Bolei Zhou, and Chen Change Loy, "Deep flow-guided video inpainting", in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[15] Yining Li, Chen Huang, and Chen Change Loy, "Dense intrinsic appearance flow for human pose transfer", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[16] Shuaicheng Liu, Lu Yuan, Ping Tan, and Jian Sun, "Steadyflow: Spatially smooth optical flow for video stabilization", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 4209–4216.

[17] Berthold KP Horn and Brian G Schunck, "Determining optical flow", *Artificial intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[18] Bruce D Lucas, Takeo Kanade, et al., *An iterative image registration technique with an application to stereo vision*, vol. 81, Vancouver, 1981.

[19] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert, "High accuracy optical flow estimation based on a theory for warping", in *European conference on computer vision*. Springer, 2004, pp. 25–36.

[20] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid, "Epicflow: Edge-preserving interpolation of correspondences for optical flow", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1164–1172.

[21] Deqing Sun, Stefan Roth, John P Lewis, and Michael J Black, "Learning optical flow", in *European Conference on Computer Vision*. Springer, 2008, pp. 83–97.

[22] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation", in *European Conf. on Computer Vision (ECCV)*, A. Fitzgibbon et al. (Eds.), Ed. Oct. 2012, Part IV, LNCS 7577, pp. 611–625, Springer-Verlag.

[23] Moritz Menze and Andreas Geiger, "Object scene flow for autonomous vehicles", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3061–3070.

[24] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox, "Flownet: Learning optical flow with convolutional networks", in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2758–2766.

[25] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2462–2470.

[26] Anurag Ranjan and Michael J Black, "Optical flow estimation using a spatial pyramid network", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4161–4170.

[27] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8934–8943.

[28] Andreas Geiger, Philip Lenz, and Raquel Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite", in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.

[29] J Yu Jason, Adam W Harley, and Konstantinos G Derpanis, "Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness", in *European Conference on Computer Vision*. Springer, 2016, pp. 3–10.

[30] Simon Meister, Junhwa Hur, and Stefan Roth, "Unflow: Unsupervised learning of optical flow with a bidirectional census loss", in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[31] Austin Stone, Daniel Maurer, Alper Ayvaci, Anelia Angelova, and Rico Jonschkowski, "Smurf: Self-teaching multi-frame unsupervised raft with full-image warping", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3887–3896.

[32] 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", in *ECCV*, 2020.

[33] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al., "Neural 3d video synthesis from multi-view video", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5521–5531.

[34] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang, "Neural scene flow fields for space-time view synthesis of dynamic scenes", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6498–6508.

[35] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely, "Dynibar: Neural dynamic image-based rendering", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 4273–4284.

[36] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang, "Dynamic view synthesis from dynamic monocular video", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 5712–5721.

[37] Joao Carreira and Andrew Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset", in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[38] Manuel Werlberger, Thomas Pock, Markus Unger, and Horst Bischof, "Optical flow guided tv-l 1 video interpolation and restoration", in *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*. Springer, 2011, pp. 273–286.

[39] Zhe Ren, Junchi Yan, Bingbing Ni, Bin Liu, Xiaokang Yang, and Hongyuan Zha, "Unsupervised deep learning for optical flow estimation", in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[40] Ramin Zabih and John Woodfill, "Non-parametric local transforms for computing visual correspondence", in *European conference on computer vision*. Springer, 1994, pp. 151–158.

[41] Mehdi Noroozi and Paolo Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles", in *European Conference on Computer Vision*. Springer, 2016, pp. 69–84.

[42] Spyros Gidaris, Praveer Singh, and Nikos Komodakis, "Unsupervised representation learning by predicting image rotations", in *International Conference on Learning Representations*, 2018.

[43] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4040–4048.

[44] Junhwa Hur and Stefan Roth, "Iterative residual refinement for joint optical flow and occlusion estimation", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5754–5763.

[45] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al., "Spatial transformer networks", in *Advances in neural information processing systems*, 2015, pp. 2017–2025.

[46] Yang Wang, Yi Yang, Zhenheng Yang, Liang Zhao, Peng Wang, and Wei Xu, "Occlusion aware unsupervised learning of optical flow", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4884–4893.

[47] Joel Janai, Fatma Guney, Anurag Ranjan, Michael Black, and Andreas Geiger, "Unsupervised learning of multi-frame optical flow with occlusions", in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 690–706.

[48] Nikolai Ufer and Bjorn Ommer, "Deep semantic feature matching", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6914–6923.

[49] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr, "Fully-convolutional siamese networks for object tracking", in *European conference on computer vision*. Springer, 2016, pp. 850–865.

[50] Zhipeng Zhang and Houwen Peng, "Deeper and wider siamese networks for real-time visual tracking", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4591–4600.

[51] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid, "Deepflow: Large displacement optical flow with deep matching", in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 1385–1392.

[52] Christian Bailer, Bertram Taetz, and Didier Stricker, "Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation", in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4015–4023.

[53] Christian Bailer, Kiran Varanasi, and Didier Stricker, "Cnn-based patch matching for optical flow with thresholded hinge embedding loss", in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[54] Jia Xu, Rene Ranftl, and Vladlen Koltun, "Accurate optical flow via direct cost volume processing", in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[55] Fatma Güney and Andreas Geiger, "Deep discrete flow", in *Asian Conference on Computer Vision*. Springer, 2016, pp. 207–224.

[56] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian Reid, "Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 340–349.

[57] Xintong Han, Xiaojun Hu, Weilin Huang, and Matthew R Scott, "Clothflow: A flow-based model for clothed person generation", in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 10471–10480.

[58] Matthew D Zeiler and Rob Fergus, "Visualizing and understanding convolutional networks", in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[59] Yves Grandvalet and Yoshua Bengio, "Semi-supervised learning by entropy minimization", in *Advances in neural information processing systems*, 2005, pp. 529–536.

[60] Chundi Liu, Guangwei Yu, Maksims Volkovs, Cheng Chang, Himanshu Rai, Junwei Ma, and Satya Krishna Gorti, "Guided similarity separation for image retrieval", in *Advances in Neural Information Processing Systems*, 2019, pp. 1554–1564.

[61] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

[62] Xingping Dong and Jianbing Shen, "Triplet loss in siamese network for object tracking", in *The European Conference on Computer Vision (ECCV)*, September 2018.

[63] Nikolaus Mayer, Eddy Ilg, Philipp Fischer, Caner Hazirbas, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox, "What makes good synthetic training data for learning disparity and optical flow estimation?", *International Journal of Computer Vision*, vol. 126, no. 9, pp. 942–960, 2018.

[64] Wei-Sheng Lai, Jia-Bin Huang, and Ming-Hsuan Yang, "Semi-supervised learning for optical flow with generative adversarial networks", in *Advances in neural information processing systems*, 2017, pp. 354–364.

[65] Wending Yan, Aashish Sharma, and Robby T Tan, "Optical flow in dense foggy scenes using semi-supervised learning", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13259–13268.

[66] Gengshan Yang and Deva Ramanan, "Volumetric correspondence networks for optical flow", in *Advances in neural information processing systems*, 2019, pp. 794–805.

[67] Tomáš Novák, Jan Šochman, and Jirí Matas, "A new semi-supervised method improving optical flow on distant domains", in *Computer Vision Winter Workshop*, 2020.

[68] Woobin Im, Tae-Kyun Kim, and Sung-Eui Yoon, "Unsupervised learning of optical flow with deep feature similarity", in *The European Conference on Computer Vision (ECCV)*, 2020.

[69] Rico Jonschkowski, Austin Stone, Jonathan T Barron, Ariel Gordon, Kurt Konolige, and Anelia Angelova, "What matters in unsupervised optical flow", *arXiv preprint arXiv:2006.04902*, 2020.

[70] Yaroslav Ganin and Victor Lempitsky, "Unsupervised domain adaptation by backpropagation", in *International conference on machine learning*. PMLR, 2015, pp. 1180–1189.

[71] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, "Distilling the knowledge in a neural network", *arXiv preprint arXiv:1503.02531*, 2015.

[72] Samuel Stanton, Pavel Izmailov, Polina Kirichenko, Alexander A Alemi, and Andrew Gordon Wilson, "Does knowledge distillation really work?", *arXiv preprint arXiv:2106.05945*, 2021.

[73] Xinlei Chen and Kaiming He, "Exploring simple siamese representation learning", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15750–15758.

[74] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al., "Bootstrap your own latent: A new approach to self-supervised learning", *arXiv preprint arXiv:2006.07733*, 2020.

[75] Pengpeng Liu, Michael R Lyu, Irwin King, and Jia Xu, "Learning by distillation: A self-supervised learning framework for optical flow estimation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[76] Vladimir Vapnik, Rauf Izmailov, et al., "Learning using privileged information: similarity control and knowledge transfer.", *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 2023–2049, 2015.

[77] Antti Tarvainen and Harri Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results", *arXiv preprint arXiv:1703.01780*, 2017.

[78] Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin D Cubuk, and Quoc V Le, "Rethinking pre-training and self-training", *arXiv preprint arXiv:2006.06882*, 2020.

[79] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[80] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan, "Unsupervised domain adaptation with residual transfer networks", *arXiv preprint arXiv:1602.04433*, 2016.

[81] Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier, *Learning a synaptic learning rule*, Citeseer, 1990.

[82] Chelsea Finn, Pieter Abbeel, and Sergey Levine, "Model-agnostic meta-learning for fast adaptation of deep networks", in *International Conference on Machine Learning*. PMLR, 2017, pp. 1126–1135.

[83] Feihu Zhang, Oliver J Woodford, Victor Adrian Prisacariu, and Philip HS Torr, "Separable flow: Learning motion cost volumes for optical flow estimation", in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10807–10817.

[84] Shihao Jiang, Dylan Campbell, Yao Lu, Hongdong Li, and Richard Hartley, "Learning to estimate hidden motions with global motion aggregation", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 9772–9781.

[85] Blender, "Spring Open Movie", https://www.blender.org/press/spring-open-movie/.

[86] BlenderFoundation, "Sintel, the durian open movie project", http://durian.blender.org/.

[87] RH Hess, CL Baker, and J Zihl, "The" motion-blind" patient: low-level spatial and temporal filters", *Journal of Neuroscience*, vol. 9, no. 5, pp. 1628–1640, 1989.

[88] Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L Davidson, Sameh Khamis, Mingsong Dou, et al., "Holoportation: Virtual 3d teleportation in real-time", in *Proceedings of the 29th annual symposium on user interface software and technology*, 2016, pp. 741–754.

[89] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer, "D-nerf: Neural radiance fields for dynamic scenes", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10318–10327.

[90] Ang Cao and Justin Johnson, "Hexplane: A fast representation for dynamic scenes", *arXiv preprint arXiv:2301.09632*, 2023.

[91] Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa, "Monocular dynamic view synthesis: A reality check", in *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, Eds., 2022.

[92] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian, "Fast dynamic radiance fields with time-aware neural voxels", in *SIGGRAPH Asia 2022 Conference Papers*, 2022, pp. 1–9.

[93] Benjamin Attal, Jia-Bin Huang, Christian Richardt, Michael Zollhoefer, Johannes Kopf, Matthew O'Toole, and Changil Kim, "Hyperreel: High-fidelity 6-dof video with ray-conditioned sampling", *arXiv preprint arXiv:2301.02238*, 2023.

[94] Yu-Lun Liu, Chen Gao, Andréas Meuleman, Hung-Yu Tseng, Ayush Saraf, Changil Kim, Yung-Yu Chuang, Johannes Kopf, and Jia-Bin Huang, "Robust dynamic radiance fields", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 13–23.

[95] Chaoyang Wang, Lachlan Ewen MacDonald, Laszlo A Jeni, and Simon Lucey, "Flow supervision for deformable nerf", *arXiv preprint arXiv:2303.16333*, 2023.

[96] Chaoyang Wang, Ben Eckart, Simon Lucey, and Orazio Gallo, "Neural trajectory fields for dynamic novel view synthesis", *arXiv preprint arXiv:2105.05994*, 2021.

[97] Sameera Ramasinghe, Violetta Shevchenko, Gil Avraham, and Anton Van Den Hengel, "Bali-rf: Bandlimited radiance fields for dynamic scene modeling", *arXiv preprint arXiv:2302.13543*, 2023.

[98] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla, "Nerfies: Deformable neural radiance fields", in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5865–5874.

[99] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan, "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields", in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5855–5864.

[100] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan, "Ref-nerf: Structured view-dependent appearance for neural radiance fields", in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2022, pp. 5481–5490.

[101] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun, "Nerf++: Analyzing and improving neural radiance fields", *arXiv preprint arXiv:2010.07492*, 2020.

[102] James T. Kajiya and Brian P Von Herzen, "Ray tracing volume densities", in *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, 1984, p. 165–174.

[103] Nelson Max and Min Chen, "Local and global illumination in the volume rendering integral", Tech. Rep., Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2005.

[104] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz, "Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields", *arXiv preprint arXiv:2106.13228*, 2021.

[105] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt, "Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video", in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12959–12970.

[106] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B Tenenbaum, and Jiajun Wu, "Neural radiance flow for 4d view synthesis and video processing", in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE Computer Society, 2021, pp. 14304–14314.

[107] Mengyu Chu, Lingjie Liu, Quan Zheng, Erik Franz, Hans-Peter Seidel, Christian Theobalt, and Rhaleb Zayer, "Physics informed neural fields for smoke reconstruction with sparse data", *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, pp. 1–14, 2022.

[108] Daniele Baieri, Stefano Esposito, Filippo Maggioli, and Emanuele Rodolà, "Fluid dynamics network: Topology-agnostic 4d reconstruction via fluid dynamics priors", *arXiv preprint arXiv:2303.09871*, 2023.

[109] Tianhao Wu, Fangcheng Zhong, Andrea Tagliasacchi, Forrester Cole, and Cengiz Oztireli, "Dˆ 2nerf: Self-supervised decoupling of dynamic and static objects from a monocular video", *Advances in Neural Information Processing Systems*, vol. 35, pp. 32653–32666, 2022.

[110] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su, "Tensorf: Tensorial radiance fields", in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*. Springer, 2022, pp. 333–350.

[111] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin, "Neuralangelo: High-fidelity neural surface reconstruction", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 8456–8465.

[112] J Soria, R Sondergaard, BJ Cantwell, MS Chong, and AE Perry, "A study of the fine-scale motions of incompressible time-developing mixing layers", *Physics of Fluids*, vol. 6, no. 2, pp. 871–884, 1994.

[113] LJ Hart-Smith, "Application of the strain invariant failure theory (sift) to metals and fiber–polymer composites", *Philosophical Magazine*, vol. 90, no. 31-32, pp. 4263–4331, 2010.

[114] Deqing Sun, Stefan Roth, and Michael J Black, "A quantitative analysis of current practices in optical flow estimation and the principles behind them", *International Journal of Computer Vision*, vol. 106, no. 2, pp. 115–137, 2014.

[115] Haian Jin, Isabella Liu, Peijia Xu, Xiaoshuai Zhang, Songfang Han, Sai Bi, Xiaowei Zhou, Zexiang Xu, and Hao Su, "Tensoir: Tensorial inverse rendering", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[116] Jae Shin Yoon, Kihwan Kim, Orazio Gallo, Hyun Soo Park, and Jan Kautz, "Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5336–5345.

[117] Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf, "Consistent video depth estimation", *ACM Transactions on Graphics (ToG)*, vol. 39, no. 4, pp. 71–1, 2020.

[118] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman, "Mip-nerf 360: Unbounded anti-aliased neural radiance fields", *CVPR*, 2022.

[119] Woobin Im, Sebin Lee, and Sung-eui Yoon, "Semi-supervised learning of optical flow by flow supervisor", in *The European Conference on Computer Vision (ECCV)*, 2022.

[120] Woobin Im, Geonho Cha, Sebin Lee, Jumin Lee, Ju-hyeong Seon, Dongyoon Wee, and Sung-eui Yoon, "Regularizing dynamic radiance fields with kinematic fields", *Under Review*, 2024.

Acknowledgments in Korean

쉽지 않았던 대학원 과정이었습니다. 이제 마칠 때가 되어 돌아보니, 그동안 저의 힘으로는 할 수 없는 일들이 정말 많았습니다. 연구의 방향을 찾지 못할 때도 있었고, 노력하는 만큼 결과가 나오지 않아 답답해하던 순간도 있었습니다. 그 순간마다 저를 일으키시고, 힘 주시고, 다시 걷게 하셨던 하나님께 감사드립니다.

힘든 이야기를 먼저 하기는 했지만, 사실 많은 날들을 즐겁게 지냈던 것 같습니다. 교수님, 동료들과 함께 학회 참석도 하고, 식사도 하고, 풋살도 하면서 지냈던 것이 떠오릅니다. 지식과 스킬이 쌓이고, 연구 업적이 늘어가는 것은 즐거운 일이었지만, 연구실에서 좋은 동료들과 함께하지 않았더라면 불행한 시간이었을 것이라 생각합니다. 이 자리를 빌려, 감사한 분들께 깊은 감사를 전하고자 합니다.

박사과정동안 저를 지도해주신 윤성의 교수님께 감사드립니다. 교수님의 통찰력있는 연구 지도는 저를 강한 연구자로 성장시키는 양분이 되었습니다. 무엇보다도, 교수님이 때때로 주시는 따뜻한 조언의 말씀은 제가 불평하며 나태해지려고 할 때 스스로를 돌아보는 좋은 길잡이가 되었습니다. 교수님을 본받아, 저도 훌륭한 연구자와 따뜻한 리더가 되겠습니다.

또, 본 학위 논문 내용에 대해 유익한 코멘트를 주시고, 심사위원으로 함께 해 주신 김태균 교수님, 조성호 교수님, 안성진 교수님, 성민혁 교수님께도 감사를 드립니다. 뛰어난 교수님들과 함께 연구하게 되어 영광이었고, 그로 인해 많이 성장하였습니다. 심사위원 교수님들께 다시 한 번 감사드립니다.

수년간 저와 함께해준 연구실 서브 그룹에 감사드립니다. 본인이 후크 선장이라 하지만 제 눈에는 피터팬인 창호, 굳은 일과 즐거운 모든 일을 함께 했던 세빈, 저를 믿고 그 어려운 디퓨전 연구를 개척한 주민, 학부 인턴시절부터 뭐든 잘 해내는 주형. 부족한 저를 믿고 따라 주어서 감사합니다! 모두 저의 좋은 스승이었고, 덕분에 연구실 생활이 즐거웠습니다. 모두 잘 될 거예요.

우리 비전팀과도 추억이 많네요. 사운드 쪽으로 재미있는 연구 하시던 게임 잘하는 태영이형, 논문 작성할 때 좋은 조언 많이 해주셨던 수민누나, 먼저 졸업했는데 네이버에서 한 번 더 함께했던 영기군에게 감사드립니다. 비전 재윤, 윤기, 준식이형과는 CVPR 출장을 함께 가기도 했고, 많은 일들을 함께 하면서 부대낀 기억이 납니다. 모두, 오랫동안 저와 함께 해주어서 감사합니다. 비디오 연구로 함께했던 치완이형과 서버실 관리로 함께 고생했던 우재군에게도 감사를 전합니다. Xu and Guoyuan, I respect your passion for research, and thank you! 함께한 시간이 길지 않았지만, 재미있는 연구를 함께했던 수현, 진환님에게도 감사드립니다. 가끔 오셔서 밥 잘 사주셨던 충수형님도 감사드립니다. 앞으로의 미래가 기대되는 우정, 영주, 태연, 준성에게도 감사드립니다. 비전팀 화이팅!

팔/다리 달린 로봇으로 멋있는 연구하시는 민철이형, 희찬이형, 아이 키우시면서 연구도 잘하시는 인규형, 주말마다 계시던 용선이형, 저에게 스타트업에 같이 가자던 멋진 동혁이형, 옆자리에서 빠른 졸업을 위해 고생하시는 김민철 형. 로봇팀의 기둥 민성군과 그를 따르는 태근, 제일, 찬미, 석륜, 지우. 우리 연구실의 핵심인 그래픽스 팀 친구들 재윤, 규범, 훈민, 인영, 그리고 YaXin. 한 분 한 분과 함께한 추억들이 새록새록 떠오르고, 또 각자에게 감사한 일들이 많은데 다 쓰지 못해 아쉬운 마음입니다.

제 연구를 지원해주신 네이버에도 감사드립니다. 부족한 저를 케어해 주시고 제 연구에 조언을 아끼지 않으시던 건호님과 항상 든든한 리더님 동윤님에게도 감사드립니다. 네이버 비디오팀에서도 따뜻하고 좋은 분들을 많이 만나서 감사했습니다.

오랫동안 대전에서 고생하는 저를 생각해 주는 가족들에게 감사한다는 말 전하고 싶습니다. 살아오면서 많은 중요한 선택의 순간이 있을 때마다 저를 믿어주셔서 감사드립니다. 가족들이 물심양면으로 저를 지원해주셔서 길고 험난했던 학위 과정을 마칠 수 있었습니다. 밥 잘 챙겨먹으라고 걱정해 주는 엄마와 매주 응원의 메시지를 보내주는 아빠에게 감사드립니다.

마지막으로, 지난 6년간 함께한 네스프레소 머신에게도 작별 인사를 전합니다. 또, 그동안 SGVR Cluster에서 저와 함께 고생했던 80여대의 GPU와, 악조건에도 고장나지 않고 묵묵히 잘 버텨준 연구실 웹서버에게도 감사를 드립니다.

Curriculum Vitae in Korean

이 름: 임 우 빈

생 년 월 일: 1993년 08월 04일

웹 사 이 트: iwbn.github.io

학 력

2012. 3. – 2016. 2. 연세대학교 컴퓨터과학과 (학사)

2016. 3. – 2018. 2. 한국과학기술원 전산학부 (석사)

2018. 3. – 2024. 8. 한국과학기술원 전산학부 (박사)

경 력

2023. 2. – 2023. 8. CLOVA Video, 네이버 클라우드 (인턴)

프 로 젝 트

2018. 3. – 2024. 7. SGVR 연구실 홈페이지 리뉴얼 및 관리 (sgvr.kaist.ac.kr)

2023. 9. – 2024. 7. GPU Cluster 구축 및 관리 (sgvr.kaist.ac.kr/ml-research-environment)

연 구 업 적

(Full list: iwbn.github.io/#publication)

1. **Woobin Im**, Geonho Cha, Sebin Lee, Jumin Lee, Ju-hyeong Seon, Dongyoon Wee, and Sung-Eui Yoon. *Regularizing Dynamic Radiance Fields with Kinematic Fields*, Under Review, 2024.
2. Xu Yin, **Woobim Im**, Dongbo Min, Yuchi Huo, Fei Pan, and Sung-Eui Yoon. *Fine-grained Background Representation for Weakly Supervised Semantic Segmentation*, IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), 2024.
3. Juhyeong Seon, **Woobin Im**, Sebin Lee, Jumin Lee, Sung-Eui Yoon. *Extending Segment Anything Model into Auditory and Temporal Dimensions for Audio-Visual Segmentation*, International Conference on Image Processing (ICIP), 2024.
4. Jumin Lee, Sebin Lee, Changho Jo, **Woobin Im**, Ju-hyeong Seon, and Sung-Eui Yoon. *SemCity: Semantic Scene Generation with Triplane Diffusion* Conference on Computer Vision and Pattern Recognition (CVPR), 2024.
5. Sebin Lee, **Woobin Im**, and Sung-Eui Yoon. *Multi-resolution distillation for self-supervised monocular depth estimation*, Pattern Recognition Letters, 2023.

6. Jumin Lee, **Woobin Im**, Sebin Lee, and Sung-Eui Yoon, *Diffusion Probabilistic Models for Scene-Scale 3D Categorical Data*, Workshop on Image Processing and Image Understanding (IPIU), 2023.
7. **Woobin Im**, Sebin Lee, and Sung-Eui Yoon. *Semi-Supervised Learning of Optical Flow by Flow Supervisor*, European Conference on Computer Vision (ECCV), 2022.
8. Changho Jo, **Woobin Im**, and Sung-Eui Yoon. *In-N-Out: Towards Good Initialization for Inpainting and Outpainting*, British Machine Vision Conference (BMVC), 2021.
9. **Woobin Im**, Tae-Kyun Kim, and Sung-Eui Yoon. *Unsupervised Learning of Optical Flow with Deep Feature Similarity*, European Conference on Computer Vision (ECCV), 2020.
10. Abhilasha Nanda, **Woobin Im**, Key-Sun Choi, and Hyun Seung Yang. *Combined Center Dispersion Loss Function for Deep Facial Expression Recognition*, Pattern Recognition Letters, 2020.
11. **Woobin Im**, Sungeun Hong, Sung-Eui Yoon, and Hyun S. Yang. *Scale-Varying Triplet Ranking with Classification Loss for Facial Age Estimation*, Asian Conference on Computer Vision (ACCV), 2018.
12. Sungeun Hong, **Woobin Im**, and Hyun S. Yang. *CBVMR: Content-Based Video-Music Retrieval Using Soft Intra-Modal Structure Constraint*, Proceedings of the ACM international conference on Multimedia Retrieval (ICMR), 2018.
13. Sungeun Hong, Jongbin Ryu, **Woobin Im**, and Hyun S. Yang. *D3: Recognizing dynamic scenes with deep dual descriptor based on key frames and key segments*, Neurocomputing, 2018
14. Sungeun Hong, **Woobin Im**, Jongbin Ryu, and Hyun S. Yang, *SSPP-DAN: Deep Domain Adaptation Network for Face Recognition with Single Sample Per Person*, International Conference on Image Processing, 2017.