StreamFlow: Streamlined Multi-Frame Optical Flow Estimation for Video Sequences

Shangkun Sun
SECE, Peking University
Peng Cheng Laboratory
sunshk@stu.pku.edu.cn

Jiaming Liu
Tiamat AI
james.liu.n1@gmail.com

Huaxia Li
Xiaohongshu Inc.
lihx0610@gmail.com

Guoqing Liu
Minieye Inc.
liugq@ntu.edu.sg

Thomas H Li
SECE, Peking University
thomas@pku.edu.cn

Wei Gao ✉
SECE, Peking University
Peng Cheng Laboratory
gaowei262@pku.edu.cn

Abstract

Prior multi-frame optical flow methods typically estimate flow repeatedly in a pair-wise manner, leading to significant computational redundancy. To mitigate this, we implement a Streamlined In-batch Multi-frame (SIM) pipeline, specifically tailored to video inputs to minimize redundant calculations. It enables the simultaneous prediction of successive unidirectional flows in a single forward pass, boosting processing speed by 44.43% and reaching efficiencies on par with two-frame networks. Moreover, we investigate various spatiotemporal modeling methods for optical flow estimation within this pipeline. Notably, we propose a simple yet highly effective parameter-efficient Integrative spatiotemporal Coherence (ISC) modeling method, alongside a lightweight Global Temporal Regressor (GTR) to harness temporal cues. The proposed ISC and GTR bring powerful spatiotemporal modeling capabilities and significantly enhance accuracy, including in occluded areas, while adding modest computations to the SIM pipeline. Compared to the baseline, our approach, StreamFlow, achieves performance enhancements of 15.45% and 11.37% on the Sintel clean and final test sets respectively, with gains of 15.53% and 10.77% on occluded regions and only a 1.11% rise in latency. Furthermore, StreamFlow exhibits state-of-the-art cross-dataset testing results on Sintel and KITTI, demonstrating its robust cross-domain generalization capabilities. The code is available here.

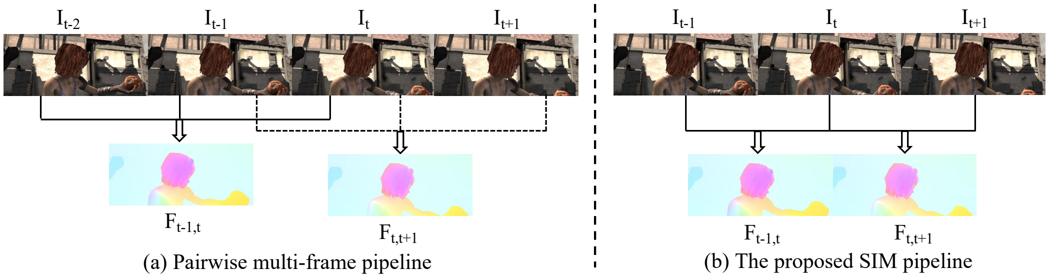
1 Introduction

Optical flow estimation, which aims to model the per-pixel correspondence between two consecutive frames, is a fundamental task in computer vision. It has various downstream video applications, such as video generation [22, 33, 20], video editing [6, 51], and video compression [25, 19]. In video streams, there is frequently a need for multi-frame input and continuous optical flow estimation.

✉Corresponding author.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

Nevertheless, previous multi-frame methods usually perform spatiotemporal modeling in a pairwise way, which leads to redundant computation, as depicted in Fig. 1 and Alg. 1. Given input frames I_{t-1}, I_t, I_{t+1}, previous pairwise methods such as VideoFlow [38] only output *one* forward flow $F_{t\to t+1}$. To derive the consecutive flow $F_{t-1\to t}$, one additional forward pass with the input of I_{t-2}, I_{t-1}, I_t is needed.



(a) Pairwise multi-frame pipeline

(b) The proposed SIM pipeline

Figure 1: Comparison between the pairwise and the proposed Streamlined In-batch Multi-frame (SIM) pipeline. Short dashed lines represent additional computations.

This gives rise to the design of what we refer to as in-batch estimation —a pipeline that simultaneously predicts all successive unidirectional flows within *one* forward pass, which greatly reduces the redundant computation. For instance, given input frames I_{t-1}, I_t, I_{t+1}, all consecutive flows $F_{t-1\to t}$ and $F_{t\to t+1}$ could be derived within *one* forward pass. It directly saves one forward pass time in the decoder. When T frames are input simultaneously in the batch, there is still only *one* forward pass needed to obtain $T-1$ flows, and more time is served on average as T increases.

Nevertheless, despite the acceleration brought by the in-batch estimation, its alteration of the pipeline renders it unable to directly apply past pairwise spatiotemporal modeling methods [16, 4, 38]. Notably, an effective spatiotemporal modeling approach is the key to resolving problems such as occlusions for multi-frame methods. Thus, this raises a new key question: *Under the constraint of in-batch estimation, how to perform effective spatiotemporal modeling while maintaining efficiency?*

In this work, we propose StreamFlow, a streamlined multi-frame optical flow estimation method tailored for video inputs. StreamFlow is made efficient through the Streamlined In-batch Multi-frame (SIM) pipeline, which avoids repetitive calculations when predicting unidirectional flows for videos. Furthermore, StreamFlow explores the challenge of effectively modeling spatiotemporal cues under the constraint of non-overlapping in-batch estimation. StreamFlow explores various spatiotemporal modeling methods and ultimately derives simple yet highly effective methods: a parameter-efficient Integrative spatiotemporal Coherence (ISC) module during encoding, and a lightweight Global Temporal Regressor (GTR) to decode all flows. Benefiting from these modules, StreamFlow achieves remarkable performance on Sintel and KITTI datasets *without self-supervised pre-training and the aim of bidirectional flows*. Moreover, StreamFlow attains state-of-the-art cross-dataset generalization results with comparable efficiency compared to two-frame methods, as illustrated in Figure 2.

In summary, our contributions are as follows: (1) We propose a Streamlined In-batch Multi-frame (SIM) pipeline for optical flow estimation, which eliminates the repetitive overlapping computation when computing forward flows for video inputs. (2) Under the constraint of a non-overlapping pipeline, we specifically designed the Integrative spatiotemporal Coherence (ISC) method to perform spatiotemporal modeling without additional parameters. (3) For the SIM pipeline, we devise a Global Temporal Regressor (GTR) during decoding to further exploit temporal cues with modest additional computation cost. (4) The proposed StreamFlow achieves superior performances on multiple benchmarks, particularly in occluded regions with comparable efficiency compared with two-frame methods, resulting in substantial improvements in optical flow estimation.

2 Related work

Two-frame optical flow. Optical flow estimation in the form of a supervised learning task has been performed by FlowNet [8] using Convolutional Neural Networks (CNN). The encoder-decoder architecture of FlowNet predicts flow from coarse to fine using the hierarchy of the flow pyramid. Thereafter, a number of refined coarse-to-fine approaches [12, 42, 43, 10, 11, 50, 54, 13] emerged.

The flow pyramid is constructed for the coarse-to-fine approach, which predicts the flow based on the flow guidance at a higher pyramid level. However, the flow guidance is often too coarse to capture small motions delicately and creates errors in later estimation. RAFT [45] recently introduced an iterative all-pairs flow transform technique, which enables the prediction of high-resolution flow and recurrent refinement of the residual flow estimation. It addresses the challenges of small motions and has consequently received high interest and performance in the field, inspiring numerous follow-up works [15, 28, 55, 21, 27, 56, 47]. To further address the occlusion issue, SKFlow [44] begins by expanding the spatial receptive field and designing effective large convolution kernel modules in the decoder of the flow network, with modest computational cost. Although StreamFlow is based on the SKFlow framework, it is designed to explore additional temporal cues and minimize redundant computations that are prevalent in earlier multi-frame methods. Their methodologies and foundational principles differ significantly.

Multi-frame optical flow. Exploiting temporal cues in optical flow estimation is an effective way to recover the occluded motion. Previous works [37, 46, 32, 1, 16, 4, 38, 26] propose various approaches to fuse temporal cues, such as leveraging previously predicted motion feature, optical flow, or contextual information. For instance, ContinualFlow [32] uses previous flow priors to estimate the current occlusion map. STaRFlow [1] passes extracted features in multiple scales, jointly with occlusion maps. [45] proposes a warm-start strategy to initialize the original flow with the past flow before prediction. MFCFlow [4] and MFRFlow [16] propose to leverage previously estimated motion features during decoding via feed-forward CNNs and self-similarity modeling, respectively. Nevertheless, these methods obtain a pairwise strategy when handling video sequences, which divide the input sequence into lots of overlapping groups and take huge repeated computations. SplatFlow [46] utilizes the differentiable splatting transformation to explore temporal cues. TransFlow [26] decodes all flows simultaneously and achieves impressive results. However, it needs self-supervised pre-training on the flow datasets to help the temporal modeling modules converge. Besides, its pure transformer architecture does not have advantages in time. VideoFlow [38] employs TROF and MOP modules to utilize multi-frame temporal cues and bidirectional optical flow to effectively mitigate occlusion issues. Nevertheless, it still follows the pairwise method to predict multiple unidirectional flows with the cost of predicting bidirectional flows. Differently, StreamFlow is proposed to avoid redundant, overlapping computation for consecutive unidirectional flow predictions while exploring efficient and effective temporal modules design under such a pipeline. It addresses the redundancy problems previous pairwise methods including VideoFlow encountered, and achieves excellent accuracy with latency similar to some two-frame methods.

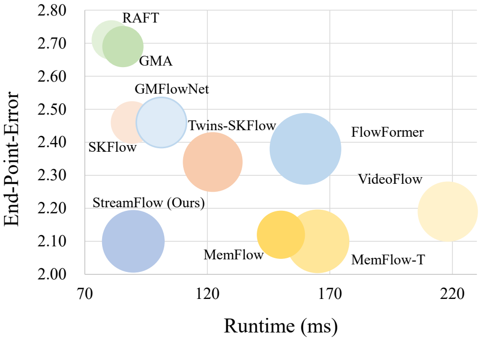


Figure 2: Comparison between performance and efficiency. A larger bubble denotes more parameters. Models are trained via the (C+)T schedule and tested on the Sintel final pass.

3 Methodology

In this Section, we introduce StreamFlow, an efficient and effective in-batch framework for multi-frame optical flow estimation. Extensive experiments guided our specific design of each module in StreamFlow. The key components of StreamFlow consist of three parts: (1) The Streamlined In-batch Multi-frame (SIM) pipeline for efficient multi-frame estimation, which contributes to the speed improvement of StreamFlow. (2) Integrative spatiotem-

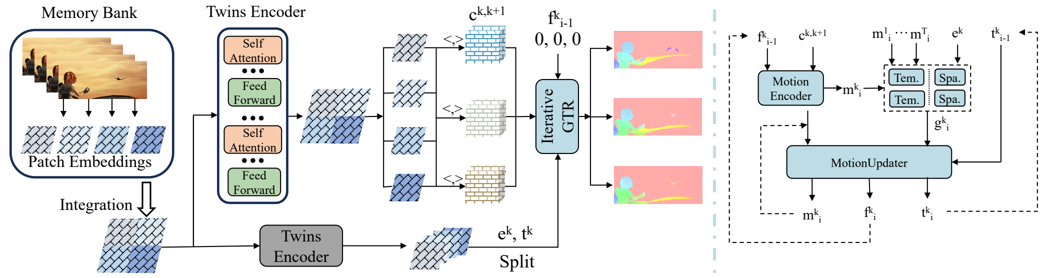
Algorithm 1 Pairwise Multi-frame Estimation

```
Input: frames $I_i$, size $N$, group size $T$
for $i = 1$ to $N - T + 1$ do
    $F_i$ =Model$(I_i, I_{i+1}, ..., I_{i+T-1})$.
end for
```

Algorithm 2 StreamFlow Multi-frame Estimation

```
Input: frames $I_i$, size $N$, group size $T$
Initialize $i = 1$, stride= $T - 1$.
repeat
    $F_i, ..., F_{i+stride-1} = Model(I_i, ..., I_{i+stride})$
    $i = i + stride$
until $i + stride > N$
$j = N - stride$
$F_j, F_{j+1}, ..., F_N = Model(I_j, I_{j+1}, ..., I_N)$.
```

poral Coherence (ISC) modeling, which is parameter-efficient and is specifically designed for spatiotemporal modeling in the encoder. (3) Global Temporal Regressor (GTR), which is quite lightweight and learns temporal relations during decoding. We will first give an overview of our methods in Section 3.1, and then introduce each module in Section 3.2, Section 3.3, and 3.4, respectively. In the end, we discuss the supervision in Section 3.5.



(a) Overall framework

(b) Iterative GTR

Figure 3: Overview of StreamFlow. (a) illustrates the overall framework and <,> denotes the dot-product operation. The computation of cost volume is limited to adjacent frames and is performed once in one forward pass. Flows are initialized to zeros. (b) depicts the details of the GTR decoder.

3.1 Overview

The overall framework of StreamFlow is illustrated in Figure 3. For the basic encoder and decoder, similar to VideoFlow [38], StreamFlow adopts the Twins transformer [5] as the encoder and utilizes the motion encoder and updater in SKFlow [44] during decoding. The overall iterative-refinement design that adopts an iterative decoder is the paradigm proposed in IRR [12] and followed by a lot of subsequent works [44, 41, 15, 44, 9, 39]. Input frames are first passed to two feature encoders that share the same architecture to extract the correlation feature and contextual feature, respectively. Then, the multi-scale all-pairs correlation volume is calculated based on the correlation feature. Namely, given feature embeddings $\mathbf{e}_1$ and $\mathbf{e}_2$ from the target frame and the reference frame, respectively:

$$\mathbf{c}^l(i,j,m,n) = \frac{1}{2^{2l}} \sum_u^{2^l} \sum_v^{2^l} \left\langle \mathbf{e}_1(i,j), \mathbf{e}_2(2^l m + u, 2^l n + v) \right\rangle, \tag{1}$$

where the derived $\mathbf{c}^l(i,j,m,n)$ is the average over the correlation in the local $2^l \times 2^l$ window. l denotes the l^{th} correlation level. u and v are the horizontal and vertical pixel motions, respectively. $\langle , \rangle$ refers to the dot product function. In summary, $\mathbf{c}^l(i,j,m,n)$ means the cost volume vector of $\mathbf{e}_1$ and $\mathbf{e}_2$ pooled with the $2^l \times 2^l$ kernel.

Then, the iterative decoder refines the flows via several updates. As depicted in Figure 3, flows are initialized to zeros. The derived multi-scale correlation volume, extracted context feature, and the initialized flows are passed to the decoder, and then the refinement is conducted.

3.2 Streamlined in-batch multi-frame pipeline

As illustrated in Figure 1, previous pairwise multi-frame networks typically perform redundant computations for video inputs, resulting in substantial computational overlap. We have briefly discussed this issue in the first two paragraphs of Section 1. Specifically, given N frames $\{I_1, I_2, ..., I_N\}$ and a group size of T ($N \geq T \geq 3$), pairwise methods need to form $N - T + 1$ groups, namely, $\{I_1, I_2, ..., I_T\}, \{I_2, I_3, ..., I_{T+1}\}, ..., \{I_{N-T+1}, ..., I_N\}$. In each group, pairwise methods use spatiotemporal cues from other frames to compute the optical flow only for the current frame. In this case, $T - 1$ times of forward process is performed to derive $T - 1$ flows. In contrast, StreamFlow introduces a Streamlined In-batch Multi-frame (SIM) Pipeline designed to minimize redundancy. For the same input, the SIM pipeline only forms $\lceil \frac{N-1}{T-1} \rceil$ groups, namely,

$\{I_1, I_2, ...I_T\}, \{I_T, I_{T+1}, I_{2T-1}\}, ..., \{I_{N-T+1}, ..., I_N\}$. In this case, frames are divided into non-overlapping groups (except for the first frame of each group). Within each group, the SIM pipeline calculates all uni-directional flows by modeling the spatiotemporal cues interconnecting all frames. In other words, only *one* forward pass is needed to derive $T-1$ flows. From this process, we could learn that the upper limit of the saved time for the SIM pipeline is approximately $\frac{\alpha}{T-1}$, where α represents the computational time of the decoder. To approach the theoretical limit as closely as possible, a lightweight yet effective method of spatiotemporal modeling is crucial. This is precisely the key contribution that StreamFlow adds to the SIM pipeline, which will be discussed in detail in Section 3.3 and Section 3.4. Additionally, when directly comparing StreamFlow with other multi-frame methods in practice, we find that the speed improvements with StreamFlow far exceed the theoretical limits calculated above. This is because StreamFlow inherently includes a memory bank mechanism in its encoder, whereas previous multi-frame methods required special additional implementation to utilize Memory banks for caching features. Nevertheless, even when compared to methods that include a memory bank, StreamFlow still achieves significant speed advantages due to its efficiency in the decoder, as demonstrated in Figure 2 and Appendix A.1.

3.3 Integrative spatiotemporal coherence

During the encoding process, we propose an Integrative spatiotemporal Coherence (ISC) modeling method, especially for the SIM pipeline. Our design principles for temporal modeling modules in the decoder encompass two facets: firstly, adherence to the design criteria of the SIM pipeline, with a focus on minimizing pair-wise overlap operations, such as the computation of cross-frame attention between every pair of consecutive frames. Secondly, the modules should be efficient enough and not impede the overall speed of the network.

Therefore, we design the ISC method, which introduces no additional parameters while learning spatiotemporal relations efficiently and effectively. The ISC method inherently takes the original modules in Twins. Specifically, after deriving patch embeddings from consecutive frames, patches from different frames are reorganized along the spatial dimension. Subsequently, it models the derived spatiotemporal graph using self-attention mechanisms and feed-forward layers in Twins, which can be formulated as,

$$\mathbf{x}_d^j = \mathcal{I}(\mathbf{x}_{1,d}^j, \mathbf{x}_{2,d}^j, ..., \mathbf{x}_{T,d}^j), \tag{2}$$

$$\mathbf{y}_d^j = f(\mathbf{q}(\mathbf{x}_d^j), \mathbf{k}(\mathbf{x}_d^j))\mathbf{v}(\mathbf{x}_d^j), \tag{3}$$

$$\mathbf{x}_d^j = \mathbf{x}_d^j + \mathbf{W_{proj}}\mathbf{y}_d^j, \tag{4}$$

where $f(\cdot, \cdot)$ is the attention function which conducts dot-product and softmax operation, $\mathbf{x}_{t,d}^j$ is the j^{th} vector along spatial dimension at channel d of the t^{th} frame. $\mathcal{I}$ denotes the *integration* operation, which integrates temporally contiguous multiple input embeddings into a large feature embedding along the spatial dimension. $\mathbf{q}, \mathbf{k}$ and $\mathbf{v}$ is the derived query, key, and value vector. $\mathbf{W_{proj}}$ is the projection matrix. By leveraging the derived spatiotemporal graph, the spatial and temporal relations are learned effectively, and no additional parameters are involved.

3.4 Global temporal regressor

As for the decoder, we propose a Global Temporal Regressor (GTR) to predict and refine flows. Compared with the previous widely used decoder [45, 15, 44, 28, 53, 27], GTR introduces the temporal modeling module to exploit temporal cues from consecutive frames. Different from VideoFlow [38] that concatenates motion features along a temporal dimension and implicitly learns temporal relations or TransFlow [26] that applies a transformer symmetric to the encoder, the core of GTR is super convolution kernels [44] and a lightweight temporal transformer block. The input correlation volume, initialized flows, and contextual features are first passed into a motion encoder to derive motion features and then extracted for spatiotemporal features, which can be formulated as:

$$\begin{aligned}
\mathbf{m}_i^k &= \mathcal{M}(\mathbf{f}_{i-1}^k, \mathbf{c}^{k,k+1}), && (5)\\
\mathbf{r}_i &= \mathcal{F}_{t=1}^{T}(\mathbf{m}_i^t), && (6)\\
\mathbf{s}_i^k &= \mathcal{A}(\mathbf{m}_i^k, \mathbf{e}^k), && (7)\\
\mathbf{g}_i^k &= \mathcal{C}(\mathbf{r}_i, \mathbf{s}_i^k), && (8)\\
\mathbf{t}_i^k, \mathbf{\Delta f}_i^k &= \mathcal{U}(\mathbf{m}_i^k, \mathbf{g}_i^k, \mathbf{t}_{i-1}^k), && (9)\\
\mathbf{f}_i^k &= \mathbf{f}_{i-1}^k + \mathbf{\Delta f}_i^k && (10)
\end{aligned}$$

where $\mathbf{m}_i^k$ is the derived motion feature of frame k at the i^{th} update and $\mathbf{f}_{i-1}^k$ denotes the flow of frame k after $i-1^{th}$ refinement. $\mathbf{c}^{k,k+1}$ denotes the correlation volume between frame k and $k+1$. $\mathcal{M}$ denotes the motion encoder which is the same as that in SKFlow [44]. $\mathbf{r}_i$ denotes the temporal feature embedding extracted from the motion features of all frames. Notably, the caching mechanism of the MemoryBank is employed, thus necessitating the calculation of $\mathbf{r}_i$ only once for different frames. $\mathcal{F}$ is a lightweight temporal-learning layer that consists of temporal attention and feed-forward layers. $\mathbf{e}^k$ refers to the feature embedding of frame k. Note that e^k and $c^{k,k+1}$ are not updated during the refinement. $\mathcal{A}$ denotes the spatial cross attention inspired by [15], but takes $\mathbf{m}_i^k$ and $\mathbf{e}^k$ as the input. $\mathcal{C}$ represents the concatenation operation and $\mathcal{U}$ refers to the motion updater. $\mathbf{t_i^k}$ denotes the extracted contextual information, which will be updated during each refinement. In practice, the decoder estimates the residual of flow $\mathbf{\Delta f}_i^k$. And the final flow $\mathbf{f}^k$ is updated via $\mathbf{\Delta f}_i^k$ during each refinement.

3.5 Supervision

StreamFlow adopts the overall loss in the same group as the total loss function. For each flow, StreamFlow adopts the same loss function as successful two-frame networks. Namely, the weighted sum for the predicted flows at different refinements. During both the training and the fine-tuning process, the supervision can be formulated as follows:

$$\mathcal{L} = \sum_{k=1}^{T-1} \sum_{i=1}^{N} \theta^{N-i} \, ||\mathbf{f}_i^k - \mathbf{f}_{gt}^k||_1, \tag{11}$$

where $\mathbf{f}_i^k$ refers to the flow of frame k at the i^{th} refinement. T and N are the number of frames and refinements, respectively. θ denotes the weights on corresponding estimated flows. $\mathbf{f}_{gt}$ is the ground truth flow and $||\cdot||_1$ means the l_1 distance between ground truth and our predicted flow. In practice, N is set to 12, θ is set to 0.8, the same as previous works [38, 45, 44, 15] for a fair comparison.

4 Experiments

Experimental setup. In this study, we evaluate StreamFlow on the Sintel [3], KITTI [31], and Spring [30] datasets, following previous works [44, 9, 7]. In previous works, models are initially pre-trained on the FlyingChairs [8] and FlyingThings [29] datasets using the "C+T" schedule and then are subsequently fine-tuned using the "C+T+S+K+H" schedule on Sintel and KITTI datasets. *After "C+T", the cross-dataset generalization tests are often performed, and the models from this stage are frequently used for various cross-domain video downstream tasks [18, 23]. In "+S+K+H", the evaluation typically focuses on intra-domain generalization capabilities.* In specific, for Sintel, models are trained on a combination of FlyingThings, Sintel, KITTI, and HD1K [17]. Models are then trained on the KITTI dataset for KITTI evaluation and on the Spring dataset for Spring evaluation.

Implementation details. Our StreamFlow method is built with PyTorch [34] library, and our experiments are conducted on the NVIDIA A100 GPUs. During training, we adopt the AdamW [24] optimizer and the one-cycle learning rate policy [40], following previous works [45, 15, 44]. The number of refinements in the decoder is set to 12, following previous works. Given the absence of multi-frame data information in the Chairs dataset, we follow VideoFlow [38] to directly train on the FlyingThings in the first stage. For the Spring dataset, we follow the settings of MemFlow [7] and fine-tune the model for 180k steps. The remaining training configurations are consistent with prior works [38, 44, 15, 45]. The temporal and non-temporal modeling modules are concurrently trained.

Training Data	Method	Sintel (train)		KITTI-15 (train)		Sintel (test)		KITTI-15 (test)
		Clean	Final	Fl-epe	Fl-all	Clean	Final	Fl-all
(C+)T	HD3 [52]	3.84	8.77	13.17	24.0	-	-	-
	PWC-Net [42]	2.55	3.93	10.35	33.7	-	-	-
	RAFT [45]	1.43	2.71	5.04	17.4	-	-	-
	CRAFT [41]	1.27	2.79	4.88	17.5	-	-	-
	AGFlow [28]	1.31	2.69	4.82	17.0	-	-	-
	Separable Flow [53]	1.30	2.59	4.60	15.9	-	-	-
	GMA [15]	1.30	2.74	4.69	17.1	-	-	-
	SKFlow [44]	1.22	2.46	4.27	15.5	-	-	-
	FlowFormer [9]	1.00	2.45	4.09	14.7	-	-	-
	GAFlow [27]	1.02	2.45	3.98	15.0	-	-	-
	TransFlow [26]	0.93	2.33	3.98	14.4	-	-	-
	VideoFlow-BOF [38]	1.03	2.19	3.96	15.3	-	-	-
	SplatFlow [46]	1.22	2.97	**3.70**	15.3	-	-	-
	Ours	**0.87**	**2.11**	3.85	**12.6**	-	-	-
(C+)T+S+K+H	IRR-PWC [12]	(1.92)	(2.51)	(1.63)	(5.3)	3.84	4.58	7.65
	MaskFlowNet [54]	-	-	-	-	2.52	4.17	6.10
	Separable Flow[53]	(0.69)	(1.10)	(0.69)	(1.6)	1.50	2.67	4.64
	PWC-Fusion [43]	-	-	-	-	3.43	4.57	7.17
	StarFlow [1]	-	-	-	-	2.72	3.71	7.65
	RAFT* [45]	(0.76)	(1.22)	(0.63)	(1.5)	1.61	2.86	5.10
	GMA* [15]	(0.62)	(1.06)	(0.57)	(1.2)	1.39	2.47	5.15
	GMFlowNet [55]	(0.59)	(0.91)	(0.64)	(1.5)	1.39	2.65	4.79
	AGFlow* [28]	(0.65)	(1.07)	(0.58)	(1.2)	1.43	2.47	4.89
	SKFlow* [44]	(0.52)	(0.78)	(0.51)	(0.9)	1.28	2.27	4.84
	FlowFormer [9]	(0.48)	(0.74)	(0.53)	(1.1)	1.16	2.09	4.68
	MFRFlow [16]	(0.64)	(1.04)	(0.54)	(1.1)	1.55	2.80	5.03
	MFCFlow [4]	(0.56)	(0.89)	(0.55)	(1.1)	1.49	2.58	5.00
	TransFlow [26]	(0.42)	(0.69)	(0.49)	(1.05)	1.06	2.08	4.32
	VideoFlow-BOF [38]	(0.37)	(0.54)	(0.52)	(0.85)	**1.00**	**1.71**	4.44
	SplatFlow [46]	(0.53)	(0.91)	(0.80)	(2.40)	1.12	2.07	4.61
	Ours	**(0.28)**	**(0.38)**	**(0.47)**	**(0.77)**	1.04	1.87	**4.24**

Table 1: Quantitative results on Sintel and KITTI. The average End-Point Error (EPE) is reported as the evaluation metric if not specified. * refers to the warm-start strategy [45] that use the previous flow for initialization. Bold and underlined metrics denote the method that ranks 1st and 2nd, respectively.

4.1 Quantitative results

From Table 1 and Table 4.1, we can learn that StreamFlow achieves advanced 0-shot performance on Sintel and KITTI. Compared to previous methods, StreamFlow reduces the 0-shot end-point error by 0.16 and 0.08 on the challenging Sintel clean and final pass, respectively. On KITTI, StreamFlow outperforms the previous state-of-the-art 0-shot results with 0.11 and 17.65% lower EPE and Fl-all metric. Besides, without self-supervised pre-training or bi-directional flows, StreamFlow attains commendable accuracy and efficiency on the challenging Sintel and KITTI test benchmarks using (C)+T+S+K+H schedule. On the challenging Spring [30] dataset, StreamFlow also achieves enhanced performance both before and after fine-tuning. **A detailed analysis on its shortcomings is in Section 5**.

4.2 Occlusion analysis

In this section, we validate if StreamFlow could help improve the performance on the occlusions. We compare StreamFlow with its base two-frame model Twins-SKFlow, which strengthens SKFlow [44] with the Twins [5] encoder. Evaluations are conducted on the matched and unmatched areas of the Sintel test dataset. The matched areas denote regions visible in adjacent frames and the unmatched areas refer to regions visible only in one of two adjacent frames. Our models are trained using the T+S+H+K schedule. We could learn that StreamFlow attains remarkable improvements on occluded areas, as shown in Table 3. We also visualize the performance on occluded regions, which are shown in the supplements. On the challenging Sintel final test set, StreamFlow attains the improvement of 10.77% and 11.83% on unmatched and matched regions, respectively. On the clean pass, StreamFlow improves the performance by 15.53%, 15.56%, and 15.45% on unmatched, matched, and overall

Method	1px												EPE	Fl	WAUC
	total	low-det.	high-det.	matched	unmat.	rigid	non-rig.	not-sky	sky	s0-10	s10-40	s40+			
RAFT† [45]	6.79	6.43	64.09	6.00	39.48	4.11	27.09	5.25	30.18	3.13	5.30	41.40	1.48	3.20	90.92
GMA† [15]	7.07	6.70	66.20	6.28	39.89	4.28	28.25	5.61	29.26	3.65	5.39	40.33	0.91	3.08	90.72
GMFlow† [49]	10.36	9.93	76.61	9.06	63.95	6.80	37.26	8.95	31.68	5.41	9.90	52.94	0.95	2.95	82.34
FlowFormer† [9]	6.51	6.14	64.22	5.77	37.29	3.53	29.08	5.50	21.86	3.38	5.53	35.34	0.72	2.38	91.68
MS-RAFT+† [14]	5.72	5.37	61.50	5.04	33.95	3.05	25.97	4.84	19.15	2.06	5.02	38.32	0.64	2.19	92.89
MemFlow† [7]	5.76	5.39	63.35	5.11	32.76	3.29	24.42	4.49	24.99	2.92	4.82	32.07	0.63	2.11	92.25
Ours†	**5.22**	**4.87**	**59.55**	**4.56**	**32.34**	**2.87**	**23.00**	**4.44**	**17.06**	**2.60**	**4.49**	**29.07**	**0.61**	**1.86**	**93.25**
CroCo-Flow [48]	4.57	4.21	**60.59**	3.85	**34.20**	2.19	22.50	4.48	**5.87**	**1.23**	**4.33**	33.13	0.50	1.51	93.66
MemFlow [7]	4.48	4.12	61.70	3.74	35.12	2.39	**20.31**	**3.93**	12.81	1.31	4.44	31.18	**0.47**	**1.42**	93.86
Ours	**4.15**	**3.79**	61.30	**3.42**	34.30	**1.99**	20.54	3.99	6.68	1.24	4.38	**27.94**	**0.47**	**1.42**	**94.40**

Table 2: Quantitative results on Spring test benchmark. Measures are from the official Spring website, including the total score, EPE, Fl, WAUC, and detailed metrics such as 1px outlier rate, etc. Important metrics are highlighted. † denotes 0-shot test using the checkpoint from “C+T+S+H+K".

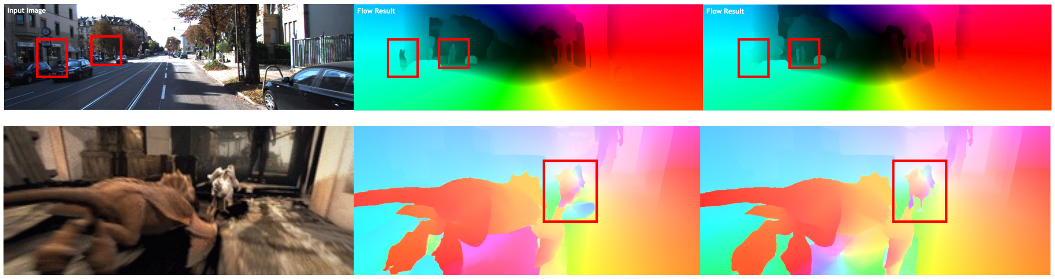
Method	Clean			Final		
	Unm.	Mat.	All	Unm.	Mat.	All
GMFlow [49]	10.56	0.65	1.74	15.80	1.32	2.90
GMFlowNet [55]	8.49	0.52	1.39	13.88	1.27	2.65
SKFlow [44]	7.24	0.55	1.28	11.51	1.46	2.28
FlowFormer [9]	7.16	0.42	1.16	11.30	0.96	2.09
TransFlow [26]	6.77	**0.36**	1.06	10.96	0.99	2.08
Baseline	7.60	0.45	1.23	11.70	0.93	2.11
Ours	**6.42**	0.38	**1.04**	**10.44**	**0.82**	**1.87**

Table 3: Occlusion analysis on Sintel test set. Unm. and Mat. denote performance on unmatched and matched areas, respectively. "Baseline" denotes our baseline method Twins-SKFlow.

regions, respectively. From the table, we can learn that StreamFlow improves not only the flow estimation in unmatched regions but also the estimation in matched regions.

4.3 Ablations

In this section, we verify the effectiveness of StreamFlow designs, as shown in Table 4. For a fair comparison, all models are trained with the same settings on the FlyingThings dataset. Then we evaluate each method on Sintel and KITTI. Below we will introduce each experiment in detail.



Input frame

Baseline (Twins-SKFlow)

StreamFlow (Ours)

Figure 4: Visualizations of results on Sintel and KITTI test sets. Differences are highlighted with red bounding boxes. StreamFlow achieves fewer artifacts on both synthetic and real-world scenes. More visualization results on DAVIS [35] and occluded regions are in the *supplements*.

SIM pipeline. We test the efficiency of the vanilla pairwise pipeline and our SIM pipeline. Pairwise methods utilize multi-frames to predict the flow of the current two frames and bring substantial redundant computation, while the SIM pipeline estimates multiple flows concurrently and minimizes the overlapping calculation. As shown in Table 4, the SIM pipeline brings great gain in efficiency. Notably, there might be information loss for the SIM pipeline due to different frame distances. For instance, given the frames I_{t-1}, I_t, I_{t+1}, the estimated flow $F_{t-1\to t}$ is typically less accurate than that derived from the sequence I_{t-2}, I_{t-1}, I_t. However, this issue tends to be alleviated with longer sequences. When the number of frames increases to 4, the impact on accuracy is significantly reduced. We will discuss this issue in more detail in the appendix.

Temporal modules. In this part, we explore the performance and efficiency of different temporal modeling methods in the flow encoder. Temporal attn refers to applying a temporal attention layer after each spatial self-attention modeling in Twins. Pseudo conv [36] denotes stacking 1D convolution layers in the temporal dimension to imitate 3D convolutions at minimal cost. We also apply 3D convolutions at the end of the flow encoder to learn temporal relations. As shown in Table 4, our ISC module achieves a good trade-off between efficiency and effectiveness. The improvements achieved by other methods are not as pronounced. We hypothesize that the limited volume of optical flow data impedes the efficient training of the spatiotemporal module from scratch to accomplish good optimization. For comparison, VideoFlow does not apply temporal modeling modules in the encoder, and TransFlow [26] applies self-supervised pre-training for better optimization.

Extra parameters. In this part, we aim to determine whether the performance gain is due to the additional parameters or the effective temporal modeling method. To this end, we introduce the additional parameters by widening the baseline network. Namely, we extract higher-dimension features along the spatial dimension and concatenate them with the original motion feature. All models in this section are equipped with the ISC module. "w/o" denotes the baseline Twins-SKFlow network. "w" means adding additional parameters. "Ours" denotes the method equipped with our temporal modeling modules. Results show the improvement achieved by simply adding more parameters is minor, and the performance gain is primarily attributed to the effectiveness of StreamFlow modules.

Experiment	Method	Sintel				KITTI		Param	Latency
		Clean	Final	Occ	Noc	Fl-epe	Fl-all	(M)	(ms)
SIM pipeline	w/o	1.03	2.34	7.69	0.35	4.64	14.70	12.49	122.18
	w/	1.03	2.34	7.69	0.35	4.64	14.70	12.49	**84.59**
Tem. modules	w/o	1.03	2.34	7.69	0.35	4.64	14.70	**12.49**	**84.59**
	Tem. attn	**0.96**	2.31	7.38	0.35	4.38	14.96	14.14	91.17
	Pse. 3D conv	1.05	2.36	7.60	0.38	4.46	15.20	13.48	87.41
	3D conv	0.98	2.34	7.63	0.33	4.57	15.59	16.03	93.05
	ISC	0.97	**2.29**	**7.11**	**0.32**	**4.14**	**14.16**	**12.49**	88.35
Extra params	w/o	0.97	2.29	7.11	0.32	4.14	14.16	**12.49**	**84.59**
	w/	0.98	2.24	7.33	**0.31**	4.15	13.94	13.77	89.29
	Ours	**0.93**	**2.15**	**7.06**	**0.31**	**3.92**	**12.36**	13.77	89.76
GTR module	w/o	0.97	2.29	7.11	0.32	4.14	14.16	**12.49**	**88.35**
	w/	0.93	**2.15**	**7.06**	**0.31**	**3.92**	**12.36**	13.77	89.76
ISC module	w/o	1.01	2.19	7.23	0.33	4.06	13.95	13.77	**86.02**
	w/	**0.93**	**2.15**	**7.06**	**0.31**	**3.92**	**12.36**	13.77	89.76
# Frames	3	0.93	2.15	7.06	0.31	3.92	**12.36**	**13.77**	89.76
	4	**0.87**	**2.11**	**6.24**	0.31	**3.85**	12.62	14.25	**85.53**

Table 4: Ablations on our proposed design. All models are trained using the "C+T" schedule. The number of refinements is 12 for all methods. The settings used in our final model are underlined.

GTR module. We also examined whether the GTR module could enhance flow predictions. "w/o" means applying vanilla SKFlow decoder while "w" denotes using GTR. All models in this part utilize the ISC module in the encoder. Table 4 demonstrates the necessity of incorporating the GTR, which could achieve stable improvement on multiple benchmarks. We could learn that GTR especially helps the estimation on the challenging final pass, with the performance gain of 0.14. In *supplements*, we give a more detailed discussion of its initialization, which is key to its training process.

ISC module. We verify the effectiveness of the proposed ISC module. All models in this part adopt GTR as the flow decoder. From Table 4, we could learn that the ISC module is efficient and effective in temporal modeling and greatly contributes to the improvement of the pipeline. It introduces no additional parameters and a modest increase in runtime, while significantly boosting the performance.

# Frames. We delve into the influence of different numbers of frames, as illustrated in Table 4. We set the number of frames to 4 due to limitations in GPU memory. From an efficiency standpoint, augmenting the number of frames results in a higher proportion of redundant computations eliminated by StreamFlow, consequently leading to a more substantial improvement in processing time. Although there is an increase in the parameter count for temporal modeling, the efficiency is further enhanced in the context of four input frames due to a reduced proportion of redundant computations, resulting in a shorter average prediction time per frame compared to the three-frame setting.

4.4 Qualitative results

We demonstrate visualization results on both synthetic (Sintel [3]) and real-world scenes (KITTI [31]), as shown in Figure 4. In the *supplements*, we also show the visualizations on the real-world dataset DAVIS [35]. Our models are pre-trained using the T+H+S+K schedule. We could learn that StreamFlow could still achieve remarkable qualitative results when generalized to real-world scenes.

4.5 Efficiency analysis

We evaluate the efficiency of the StreamFlow in terms of runtime and parameter counts. Our experiments were conducted on NVIDIA A100 GPUs. Models are trained using the (C+)T schedule and evaluated on the Sintel dataset. The runtime is measured as the average inference time per frame of five runs on the Sintel training set. We could learn StreamFlow achieves comparable efficiency with state-of-the-art two-frame methods while achieving superior performance. The key to maintaining high efficiency is its SIM pipeline. StreamFlow does not perform pairwise redundant computation and predicts all flows simultaneously. Another reason for the high speed is its CNN-based decoder. We could learn that StreamFlow is much faster than the pure two-frame transformer architecture FlowFormer. Besides, the specially designed lightweight temporal modules also contribute to the performance, simultaneously aiding in better results compared to the 2-frame baseline Twins-SKFlow.

5 Limitations

StreamFlow faces two primary challenges: (1) GPU memory usage in training. Although it is not an issue during inference (e.g., with a 432 × 1024 input, it needs about only 2.4 G and 3.3 G when the # frame is 3 and 4, respectively). But it is significantly increased during training. The storage of gradients and the batch size cause the GPU memory on a single card to reach approximately 40 G in a 4-frame setting. (2) Inter-group cues utilization. Limited to the SIM pipeline, StreamFlow is confined to using only intra-group information, and it does not utilize inter-group information during modeling. Despite achieving commendable results and surpassing some methods that incorporate inter-group information, bidirectional flows, or self-supervised pre-training strategies in cross-dataset tests, how to address this while maintaining good efficiency is a worthwhile issue to consider in future work.

6 Conclusion

In this work, we proposed StreamFlow, a multi-frame optical flow estimation approach proficient in estimating optical flows across multiple video frames using efficient spatiotemporal relationship mining. StreamFlow aims to estimate consecutive unidirectional optical flows with less overlapping computation. It proposes to estimate multi-frame optical flows via the proposed SIM pipeline and introduces efficient and effective ISC and GTR methods for temporal modeling under such circumstances. Extensive experiments on multiple challenging benchmarks demonstrate the efficiency and effectiveness of the proposed StreamFlow method.

Acknowledgements. This work was supported by The Major Key Project of PCL (PCL2024A02), Natural Science Foundation of China (62271013, 62031013), Guangdong Province Pearl River Talent Program (2021QN020708), Guangdong Basic and Applied Basic Research Foundation (2024A1515010155), Shenzhen Science and Technology Program (JCYJ20230807120808017), Shenzhen Fundamental Research Program (GXWD20201231165807007-20200806163656003), and Sponsored by CAAI-MindSpore Open Fund, developed on OpenI Community (CAAIXSJLJJ-2023-MindSpore07).

References

[1] Elaine Angelino, Daniel Yamins, and Margo Seltzer. Starflow: A script-centric data analysis environment. In *Provenance and Annotation of Data and Processes: Third International Provenance and Annotation Workshop, IPAW 2010, Troy, NY, USA, June 15-16, 2010. Revised Selected Papers 3*, pages 236–250. Springer, 2010.

[2] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *ICML*, volume 2, page 4, 2021.

[3] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *European conference on computer vision*, pages 611–625. Springer, 2012.

[4] Yonghu Chen, Dongchen Zhu, Wenjun Shi, Guanghui Zhang, Tianyu Zhang, Xiaolin Zhang, and Jiamao Li. Mfcflow: A motion feature compensated multi-frame recurrent network for optical flow estimation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5068–5077, 2023.

[5] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. *Advances in Neural Information Processing Systems*, 34:9355–9366, 2021.

[6] Yuren Cong, Mengmeng Xu, Shoufa Chen, Jiawei Ren, Yanping Xie, Juan-Manuel Perez-Rua, Bodo Rosenhahn, Tao Xiang, Sen He, et al. Flatten: optical flow-guided attention for consistent text-to-video editing. In *The Twelfth International Conference on Learning Representations*, 2023.

[7] Qiaole Dong and Yanwei Fu. Memflow: Optical flow estimation and prediction with memory. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19068–19078, 2024.

[8] 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 (ICCV)*, December 2015.

[9] Zhaoyang Huang, Xiaoyu Shi, Chao Zhang, Qiang Wang, Ka Chun Cheung, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Flowformer: A transformer architecture for optical flow. In *European Conference on Computer Vision*, pages 668–685. Springer, 2022.

[10] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8981–8989, 2018.

[11] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. A lightweight optical flow cnn—revisiting data fidelity and regularization. *IEEE transactions on pattern analysis and machine intelligence*, 43(8):2555–2569, 2020.

[12] Junhwa Hur and Stefan Roth. Iterative residual refinement for joint optical flow and occlusion estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5754–5763, 2019.

[13] 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 (CVPR)*, July 2017.

[14] Azin Jahedi, Maximilian Luz, Marc Rivinius, Lukas Mehl, and Andrés Bruhn. Ms-raft+: High resolution multi-scale raft. *International Journal of Computer Vision*, 132(5):1835–1856, 2024.

[15] 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*, pages 9772–9781, 2021.

[16] Yang Jiao, Guangming Shi, and Trac D Tran. Optical flow estimation via motion feature recovery. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 2558–2562. IEEE, 2021.

[17] 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*, pages 19–28, 2016.

[18] Tengchuan Kou, Xiaohong Liu, Wei Sun, Jun Jia, Xiongkuo Min, Guangtao Zhai, and Ning Liu. Stablevqa: A deep no-reference quality assessment model for video stability. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 1066–1076, 2023.

[19] Jiahao Li, Bin Li, and Yan Lu. Deep contextual video compression. *Advances in Neural Information Processing Systems*, 34:18114–18125, 2021.

[20] Wei Li, Zehuan Yuan, Xiangzhong Fang, and Changhu Wang. Moflowgan: Video generation with flow guidance. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2020.

[21] Yingping Liang, Jiaming Liu, Debing Zhang, and Ying Fu. Mpi-flow: Learning realistic optical flow with multiplane images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13857–13868, 2023.

[22] Mingxiang Liao, Hannan Lu, Xinyu Zhang, Fang Wan, Tianyu Wang, Yuzhong Zhao, Wangmeng Zuo, Qixiang Ye, and Jingdong Wang. Evaluation of text-to-video generation models: A dynamics perspective. *arXiv preprint arXiv:2407.01094*, 2024.

[23] Yaofang Liu, Xiaodong Cun, Xuebo Liu, Xintao Wang, Yong Zhang, Haoxin Chen, Yang Liu, Tieyong Zeng, Raymond Chan, and Ying Shan. Evalcrafter: Benchmarking and evaluating large video generation models. *arXiv preprint arXiv:2310.11440*, 2023.

[24] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

[25] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11006–11015, 2019.

[26] Yawen Lu, Qifan Wang, Siqi Ma, Tong Geng, Yingjie Victor Chen, Huaijin Chen, and Dongfang Liu. Transflow: Transformer as flow learner. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18063–18073, 2023.

[27] Ao Luo, Fan Yang, Xin Li, Lang Nie, Chunyu Lin, Haoqiang Fan, and Shuaicheng Liu. Gaflow: Incorporating gaussian attention into optical flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9642–9651, 2023.

[28] Ao Luo, Fan Yang, Kunming Luo, Xin Li, Haoqiang Fan, and Shuaicheng Liu. Learning optical flow with adaptive graph reasoning. *arXiv preprint arXiv:2202.03857*, 2022.

[29] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. arXiv:1512.02134.

[30] Lukas Mehl, Jenny Schmalfuss, Azin Jahedi, Yaroslava Nalivayko, and Andrés Bruhn. Spring: A high-resolution high-detail dataset and benchmark for scene flow, optical flow and stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4981–4991, 2023.

[31] Moritz Menze, Christian Heipke, and Andreas Geiger. Joint 3d estimation of vehicles and scene flow. In *ISPRS Workshop on Image Sequence Analysis (ISA)*, 2015.

[32] Michal Neoral, Jan Šochman, and Jiří Matas. Continual occlusion and optical flow estimation. In *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part IV 14*, pages 159–174. Springer, 2019.

[33] Haomiao Ni, Changhao Shi, Kai Li, Sharon X Huang, and Martin Renqiang Min. Conditional image-to-video generation with latent flow diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18444–18455, 2023.

[34] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[35] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv:1704.00675*, 2017.

[36] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *proceedings of the IEEE International Conference on Computer Vision*, pages 5533–5541, 2017.

[37] Zhile Ren, Orazio Gallo, Deqing Sun, Ming-Hsuan Yang, Erik B Sudderth, and Jan Kautz. A fusion approach for multi-frame optical flow estimation. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 2077–2086. IEEE, 2019.

[38] Xiaoyu Shi, Zhaoyang Huang, Weikang Bian, Dasong Li, Manyuan Zhang, Ka Chun Cheung, Simon See, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Videoflow: Exploiting temporal cues for multi-frame optical flow estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12469–12480, 2023.

[39] Xiaoyu Shi, Zhaoyang Huang, Dasong Li, Manyuan Zhang, Ka Chun Cheung, Simon See, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Flowformer++: Masked cost volume autoencoding for pretraining optical flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1599–1610, 2023.

[40] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, page 1100612. International Society for Optics and Photonics, 2019.

[41] Xiuchao Sui, Shaohua Li, Xue Geng, Yan Wu, Xinxing Xu, Yong Liu, Rick Goh, and Hongyuan Zhu. Craft: Cross-attentional flow transformer for robust optical flow. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 17602–17611, 2022.

[42] 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*, pages 8934–8943, 2018.

[43] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Models matter, so does training: An empirical study of cnns for optical flow estimation. *IEEE transactions on pattern analysis and machine intelligence*, 42(6):1408–1423, 2019.

[44] Shangkun Sun, Yuanqi Chen, Yu Zhu, Guodong Guo, and Ge Li. Skflow: Learning optical flow with super kernels. *Advances in Neural Information Processing Systems*, 35:11313–11326, 2022.

[45] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020.

[46] Bo Wang, Yifan Zhang, Jian Li, Yang Yu, Zhenping Sun, Li Liu, and Dewen Hu. Splatflow: Learning multi-frame optical flow via splatting. *International Journal of Computer Vision*, pages 1–23, 2024.

[47] Yihan Wang, Lahav Lipson, and Jia Deng. Sea-raft: Simple, efficient, accurate raft for optical flow. In *European Conference on Computer Vision*, pages 36–54. Springer, 2025.

[48] Philippe Weinzaepfel, Thomas Lucas, Vincent Leroy, Yohann Cabon, Vaibhav Arora, Romain Brégier, Gabriela Csurka, Leonid Antsfeld, Boris Chidlovskii, and Jérôme Revaud. Croco v2: Improved cross-view completion pre-training for stereo matching and optical flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17969–17980, 2023.

[49] Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, and Dacheng Tao. Gmflow: Learning optical flow via global matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8121–8130, 2022.

[50] Gengshan Yang and Deva Ramanan. Volumetric correspondence networks for optical flow. *Advances in neural information processing systems*, 32, 2019.

[51] Shuai Yang, Yifan Zhou, Ziwei Liu, and Chen Change Loy. Rerender a video: Zero-shot text-guided video-to-video translation. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–11, 2023.

[52] Zhichao Yin, Trevor Darrell, and Fisher Yu. Hierarchical discrete distribution decomposition for match density estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6044–6053, 2019.

[53] 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*, pages 10807–10817, 2021.

[54] Shengyu Zhao, Yilun Sheng, Yue Dong, Eric I Chang, Yan Xu, et al. Maskflownet: Asymmetric feature matching with learnable occlusion mask. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6278–6287, 2020.

[55] Shiyu Zhao, Long Zhao, Zhixing Zhang, Enyu Zhou, and Dimitris Metaxas. Global matching with overlapping attention for optical flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17592–17601, 2022.

[56] Shili Zhou, Ruian He, Weimin Tan, and Bo Yan. Samflow: Eliminating any fragmentation in optical flow with segment anything model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 7695–7703, 2024.

A Appendix

A.1 Comparison of latency using memory bank

As discussed in Section 3.2, to better demonstrate that the temporal advantage of StreamFlow is not solely due to the memory bank, this section explores the efficiency comparison between StreamFlow and other methods when using a memory bank. Given that the model's runtime is closely related to the coding implementation, this comparison prioritizes officially open-sourced multi-frame optical flow methods. However, as of the writing of this paper, the choices for leading open-source multi-frame methods are quite limited, and thus VideoFlow [38] was selected for comparison. The experimental setup and the machine are consistent with those described in Section 4.5, and the measured time is the average of five tests. The input is resized to 432×1024, and the model is trained via (C+)T manner. As shown in Table A.1, it can be observed that StreamFlow still exhibits good efficiency in time. This is because, in addition to the memory bank, it further optimizes the average estimation time in the decoder.

Method	Sintel (clean)	Sintel (final)	Fl-EPE	Fl-all	Latency	Hardware
VideoFlow-BOF [38]	1.03	2.19	3.96	15.3	122.37ms	A100-40G
StreamFlow (Ours)	**0.87**	**2.11**	**3.85**	**12.6**	**85.53ms**	A100-40G

Table 5: Comparison of latency using memory bank.

A.2 Impact of the frame distance

Within the SIM pipeline, the impact of different frame distance $\mathbf{D}$ indeed exists. For instance, the flow $F_{t-1,t}$ derived from I_{t-2}, I_{t-1}, I_t tends to be more accurate than that from I_{t-1}, I_t, I_{t+1} due to the different distance $\mathbf{D}_{t-1,t-2}$ and $\mathbf{D}_{t-1,t+1}$. As the frame distance increases, the information provided may decrease, as confirmed by the results in Table 7. However, this impact might not necessarily grow larger with more frames, which could be due to the affected frames mainly being distributed at the head or the tail of a group. We could define the longest frame distance that provides effective information as m. As the length of the group increases, there will be more frames in the middle of the group (i.e., frames I_t within the interval $[I_{t-m}, I_{t+m}]$ all lying within the group), and fewer frames distributed on both ends. As shown in Table 7, the impact was weakened with 4 frames. Maybe the future study on an appropriate choice of m is helpful for future multi-frame optical flow work.

A.3 Detailed metrics on the Sintel test set

Table A.4 presents a more detailed comparison on the Sintel test set between StreamFlow and its two-frame baseline. The detailed metrics including all/matched/unmatched EPE, d0-10, d10-60, d60-140, s0-10, s10-40, s40+ are from the official Sintel website. We could learn that StreamFlow performs exceptionally well in unmatched areas, validating its effectiveness in addressing occlusion issues. Compared to the baseline method, it improved by 15.53% and 10.77% in the Matched areas on the clean pass and final pass, respectively, and by 15.56% and 11.83% in the Unmatched areas. Within different ranges of occlusion boundaries, it showed improvements of 15.13% and 10.30% at d0-10, 15.79% and 13.51% at d10-60, and 10.00% and 12.50% at d60-140 on the clean pass and final pass, respectively. For pixels moving at different speeds, it improved by 16.00% and 14.58% at s0-10, 10.39% and 10.14% at s10-40, and 14.41% and 10.49% at s40+ on the clean pass and final pass, respectively. It is shown that StreamFlow has outperformed its baseline on both the clean pass and final pass, with a more pronounced improvement on the clean pass. The enhancements are particularly noticeable in areas of occlusion and for objects moving at high speeds.

A.4 Relationship between image size and GPU memory

In this section, we show the GPU usage of StreamFlow. During the inference, without the optimization in the third-party package, StreamFlow takes about 2.4 G and 3.3 G for 432 × 1024 inputs with the number of frame set to 3 and 4, respectively. With PyTorch 2.2 and flash-attention, using 12 refinements and 4 frames, the GPU memory usage for StreamFlow is shown in Table **??**. Specifically, when the image size increases by 4 times, the GPU memory usage increases to nearly 4.4 times. When the image size is increased by 9 times, GPU memory usage grows to nearly 20 times. For most current scenarios, StreamFlow still maintains a relatively moderate use of GPU memory. The variation in GPU memory usage may be influenced by underlying optimizations in the framework. We believe the memory usage could be further optimized in the future.

A.5 Qualitative analysis on real-world scenes

In this section, we facilitate our visualizations and evaluations using two prominent real-world datasets, namely DAVIS [35]. The DAVIS dataset, short for Densely Annotated VIdeo Segmentation, is a widely recognized benchmark in the field of computer vision. It comprises high-quality video sequences captured in diverse scenarios, encompassing a broad range of challenging visual conditions such as occlusions, motion blur, and dynamic object interactions. The dataset provides pixel-level annotations for every frame, facilitating precise evaluation and comparison of various video segmentation methods. The visualizations on the DAVIS dataset are shown in Figure 5. Our model is pretrained using the "T" and "T+S+H+K" schedule and then fine-tuned on KITTI [31]. "T" denotes the FlyingThings [29] dataset and "T+S+H+K" refers to the combination of the FlyingThings, Sintel [3], HD1K [17], and KITTI datasets. Then we infer our models on the DAVIS dataset. The number of refinements is set to 12. The number of input frames for each non-overlapping group is 3. We could learn that StreamFlow demonstrates remarkable adaptability across real-world datasets, showing its robust performance in challenging scenes for optical flow estimation. This is particularly evident in scenarios such as the occlusion of the bear's hind legs in the first row, first column, and the small motion of the small tennis ball in the last column. Additionally, it can be observed that in the motion captured in the first row, second, and third columns, the hind legs of the camel and the leg movements of the dancer are also vividly delineated. These instances reaffirm its efficacy in diverse and demanding environments for optical flow estimation.

A.6 Qualitative analysis on occluded regions

In this section, we focus on the performance of the occluded regions. As discussed in previous works [15, 44], here we term occlusions as areas where pixels appear in the current frame while disappearing in the next frame. We visualize the flow-error map on occluded regions of the Sintel dataset with the official occlusion masks. All models are trained using the (C+)T schedule. As shown in Figure 6, significant occluded areas are highlighted using red boxes. A darker color in the flow-error map denotes a more significant error. We could learn that StreamFlow achieves better overall performance, and attains leading performance on the occluded regions.

Image Size	GPU Memory
360×640	1.19 G
720×1280	5.20 G
1080×1920	24.11 G

Table 6: Relationship between input size and GPU memory usage during inference. The number of frames is set to 4.

Method	Sintel	
	clean	final
T3	0.93	2.15
T3†	0.90	2.13
T4	0.87	**2.11**
T4†	**0.86**	**2.11**

Table 7: Impact of frame distance. † denotes using nearer frames.

Data	Method	Unm.	Mat.	All	d0-10	d10-60	d60-140	s0-10	s10-40	s40+
Clean	Baseline	7.60	0.45	1.23	1.19	0.38	0.20	0.25	0.77	7.01
	Ours	**6.42**	**0.38**	**1.04**	**1.01**	**0.32**	**0.18**	**0.21**	**0.69**	**6.00**
Final	Baseline	11.70	0.93	2.11	2.33	0.74	0.40	0.48	1.38	11.92
	Ours	**10.44**	**0.82**	**1.87**	**2.09**	**0.64**	**0.35**	**0.41**	**1.24**	**10.67**

Table 8: Results on Sintel test set. Unm. and Mat. denote performance on unmatched and matched areas, respectively. "Baseline" denotes our baseline method Twins-SKFlow.

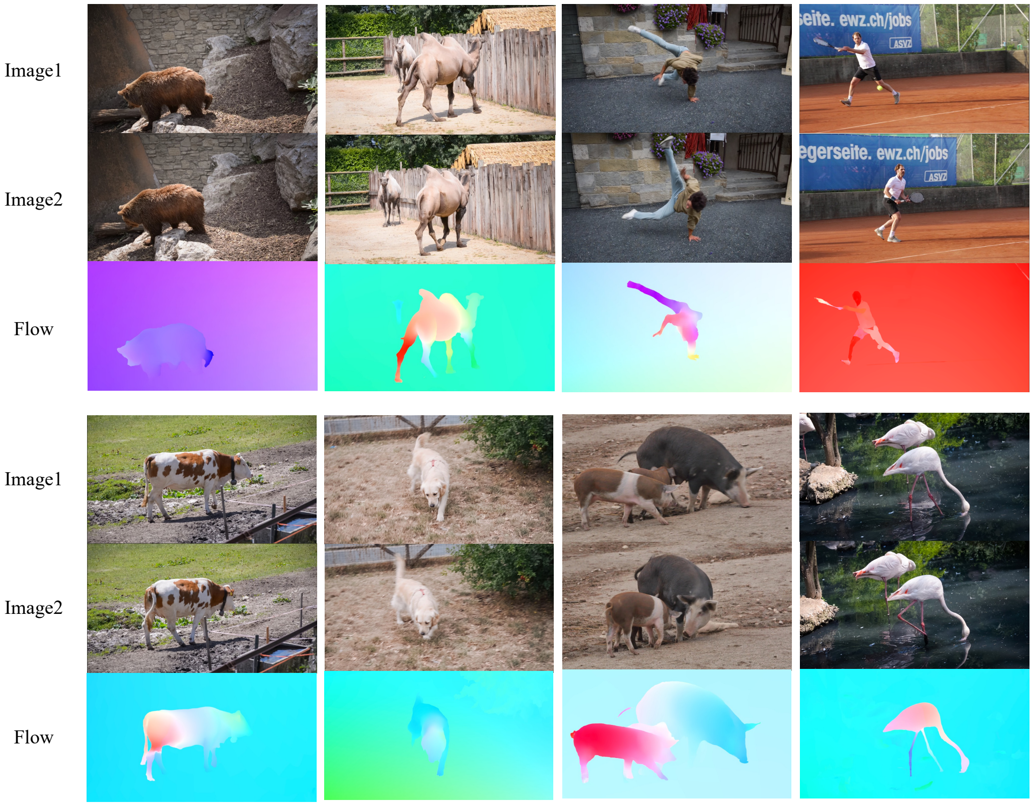


Figure 5: Visualizations of predicted flows on DAVIS [35]. StreamFlow demonstrates robust generalization to other real-world datasets, performing well in challenging scenarios for optical flow estimation, as evidenced by instances such as the occluded hind legs of the bear in the first column and the small tennis ball in the last column.

A.7 Initialization of GTR

In this section, we investigate the impact of different GTR initialization methods. Previous works in spatio-temporal modeling such as [2] have suggested initializing the temporal modules with zero values. We employed two distinct initialization approaches, namely zero initialization and PyTorch's default initialization, and the corresponding results are presented in Table 9. Following training on the FlyingThings dataset, the model was tested on the Sintel and KITTI datasets. It is evident from the results that zero initialization could contribute to a better overall performance.

Method	Sintel (Clean)	Sintel (Final)	KITTI (EPE)	KITTI (Fl-all)
Default	**0.91**	2.20	4.05	13.44
Zero-init	0.93	**2.15**	**3.92**	**12.36**

Table 9: Comparison of different ways of initialization. All models are trained under the FlyingThings.

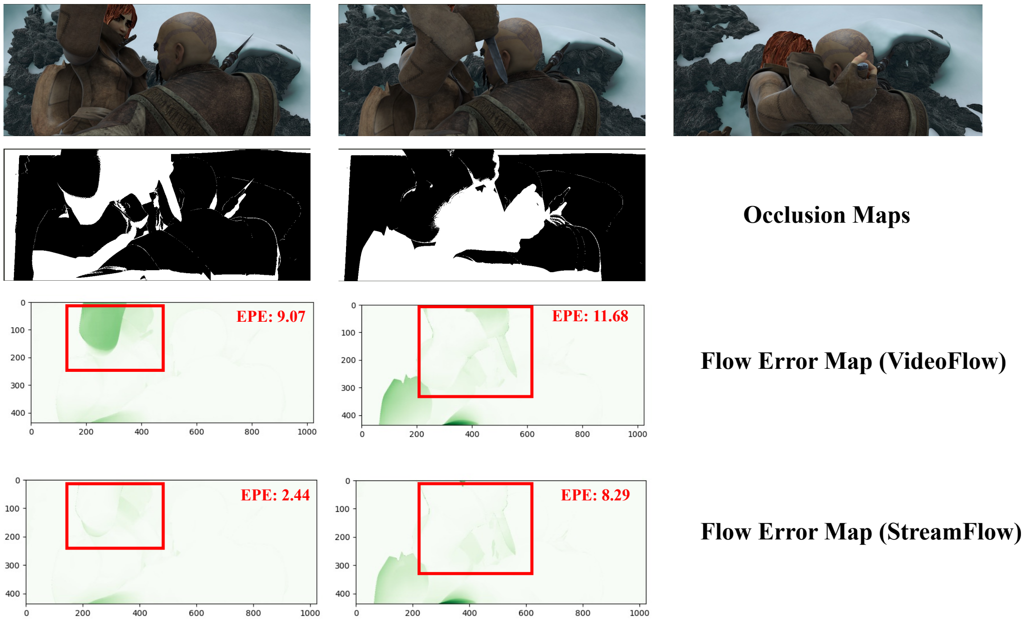


Figure 6: Visualizations of the performance on the occluded regions. StreamFlow achieves comparable performance even with advanced methods. All models are trained on the FlyingThings dataset. A darker color in the flow error map denotes a higher estimation error compared with ground truth.

B NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Please refer to the last paragraph in Section 1.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Please refer to Section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Please refer to Section 3.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Please refer to Section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code, pre-trained models, and visualization scripts can be seen in paperswithcode.com and github.com now.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Please refer to Section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The test results on Sintel and KITTI datasets are from online benchmarks that do not include information about error bars.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Please refer to Section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have thoroughly obeyed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The aim of this work is to enhance the utilization of machine learning to estimate more accurate inter-frame motion. We feel this work would not bring direct societal consequences.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This work poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets in this work are cited and used properly.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: This paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.