Recurrent Partial Kernel Network for Efficient Optical Flow Estimation

Henrique Morimitsu[1], Xiaobin Zhu[1*], Xiangyang Ji[2], Xu-Cheng Yin[1]

[1] School of Computer and Communication Engineering, University of Science and Technology Beijing, China
[2] Department of Automation, Tsinghua University, China
{hmori, zhuxiaobin, xuchengyin}@ustb.edu.cn, xyji@tsinghua.edu.cn

Abstract

Optical flow estimation is a challenging task consisting of predicting per-pixel motion vectors between images. Recent methods have employed larger and more complex models to improve the estimation accuracy. However, this impacts the widespread adoption of optical flow methods and makes it harder to train more general models since the optical flow data is hard to obtain. This paper proposes a small and efficient model for optical flow estimation. We design a new spatial recurrent encoder that extracts discriminative features at a significantly reduced size. Unlike standard recurrent units, we utilize Partial Kernel Convolution (PKConv) layers to produce variable multi-scale features with a single shared block. We also design efficient Separable Large Kernels (SLK) to capture large context information with low computational cost. Experiments on public benchmarks show that we achieve state-of-the-art generalization performance while requiring significantly fewer parameters and memory than competing methods. Our model ranks first in the Spring benchmark without finetuning, improving the results by over 10% while requiring an order of magnitude fewer FLOPs and over four times less memory than the following published method without finetuning. The code is available at github.com/hmorimitsu/ptlflow/tree/main/ptlflow/models/rpknet.

Introduction

Optical flow estimation is a fundamental computer vision problem and consists of computing the 2D motion vectors between all pixels of a pair of consecutive video frames. It has applications in multiple fields, such as action recognition, object tracking, video interpolation, and autonomous navigation.

Recent optical flow methods have been showing outstanding progress but at the cost of increasing the size and complexity of the models. On the one hand, the rise in computational cost hinders the applications of newer methods to situations with restricted computational resources. However, even more importantly, it has been shown that the generalization capacity of CNNs is directly impacted by the model size and number of training samples (Zhou and Feng 2018). Since it is not trivial to generate diverse optical flow training samples (Dosovitskiy et al. 2015; Menze and Geiger 2015), continuing to increase the model complexity may end up impacting the generalization capabilities of future models.

One of the main reasons for the substantial increase in optical flow model sizes is the adoption of more complex feature encoder networks. Some recent methods such as FlowFormer (Huang et al. 2022), MatchFlow (Dong, Cao, and Fu 2023), and CroCo (Weinzaepfel et al. 2022) finetune Transformer networks with up to hundreds of millions of parameters. Although these models perform well when finetuned and tested on the same datasets, their generalization capabilities are less evident. For example, testing FlowFormer without finetuning on the newer Spring benchmark (Mehl et al. 2023) shows a significant gap in performance to other top-performing methods.

In this paper, we go against the recent trend and propose a smaller, more efficient optical flow model. We design a new spatial recurrent encoder network that shares layers across multiple scales to improve feature representation with significantly fewer parameters. Unlike standard recurrent networks, we equip our encoder with a generalization of convolution layers, which we call Partial Kernel Convolution (PKConv). PKConv offers two main advantages: (1) process features with a variable number of channels with a single shared network, and (2) allow different parts of the convolution kernel to focus on features from specific levels. We also design a Separable Large Kernel (SLK) module that efficiently captures large context information using only 1D convolution layers to improve the model efficiency.

We incorporate our recurrent encoder into RPKNet (Recurrent Partial Kernel Network), a novel method that employs a fully recurrent encoder-decoder network to produce accurate optical flow estimations at reduced computational costs. Our spatial recurrent encoder produces multi-scale feature pyramids and then iteratively refines the optical flow prediction. We conduct extensive tests on public benchmarks and demonstrate that our method achieves top-performing generalization results while requiring a fraction of the parameters and computational cost of other state-of-the-art methods (Figure 1).

Our contributions can be summarized as three-folded:

1. We propose a spatial recurrent encoder with Partial Kernel Convolution (PKConv) layers that can efficiently extract discriminative multi-scale features with a single shared block.

*Corresponding author

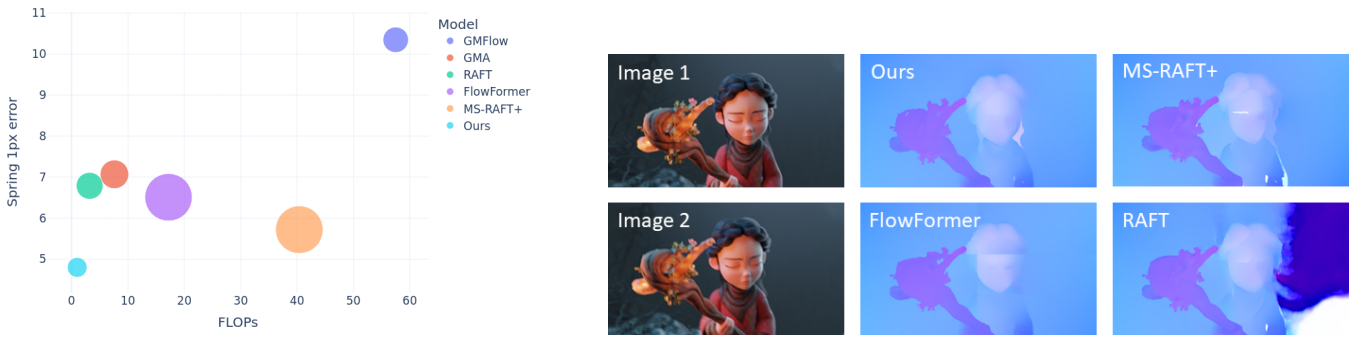

Figure 1: Generalization results on the Spring (Mehl et al. 2023) dataset. Left: We achieve the lowest error with the fewest FLOPs and model parameters (represented by the blob sizes) without using samples from Spring to train. Right: Our model leverages large contexts to produce consistent optical flow estimations on large $(1080p)$ images.

2. We design a compact Separable Large Kernel (SLK) module that only relies on 1D large convolutions to encode more contextual information.
3. We demonstrate that our model achieves strong generalization results with the lowest number of parameters and computational cost among all comparable models.

Methodology

Optical Flow Estimation

Given a pair of consecutive images $I_{i\in\{1,2\}} \in \mathcal{R}^{h\times w\times c_0}$ from a video, the optical flow estimation problem consists of finding a dense set of 2D motion vectors $F \in \mathcal{R}^{h\times w\times 2}$ that represents how each pixel moves from I_1 to I_2, where (h, w, c_0) is the height, width, and number of input channels of the image. This is typically done using an encoder-decoder deep network structure where the encoder extracts single- or multi-scale feature maps $X_{s,i}$, where s represents the scale index. The decoder processes the feature maps to generate the output flow F.

We propose a new spatial recurrent encoder that leverages our proposed PKConv and SLK layers to extract multi-scale features of variable sizes at a reduced cost. Unlike most current models that decode a single high-resolution feature map (Teed and Deng 2020), we build a traditional feature pyramid with multi-scale features to estimate optical flow in a coarse-to-fine approach. We improve the RAFT decoder (Teed and Deng 2020) with a context fusion layer to handle feature pyramids efficiently and incorporate SLK GRU modules, which we show to bring a noticeable improvement without increasing the computational cost. Figure 2 shows a diagram of our method, and we provide more details about its components in the following sections.

Spatial Recurrent Encoder

Given an input X_0, a typical (non-recurrent) encoder extracts multi-scale features $X_{s>0} \in \mathcal{R}^{h_s\times w_s\times c_s}$ by applying a series of layer blocks $X_s = \phi_s(X_{s-1})$ (e.g., ϕ represents a residual block in ResNets (He et al. 2016a)). To save computation and increase the receptive field, each feature scale usually maintains the following properties: $h_s = \frac{h_{s-1}}{2}, w_s = \frac{w_{s-1}}{2}, c_s > c_{s-1}$.

Weight sharing is the core concept of the convolution operation, and it allows the kernel to learn more representative patterns by processing multiple views of the data. Following the same principle, we improve the representativity of the layer blocks by allowing them to process multiple scales. The naïve approach to implement weight sharing would be to repeat the same layer block ϕ at all scales as:

$$X_s = \phi(X_{s-1}) = \phi^s(X_0). \tag{1}$$

However, this simple approach has some drawbacks. First, since the same ϕ is applied at every scale, all features would have the same number of channels. Depending on the chosen c_s, this imposes a high computational cost on shallower, larger features or reduces the information stored in the deeper, smaller ones. Second, unlike in the convolution operation, a shared layer block is subject to changes in scales and semantic levels. Therefore, encoding all these changes in a globally shared block may be unreasonable. A better approach would be to share a part of the blocks at each scale and have more specialized parts to focus on learning more specific characteristics of different scale levels. Our proposed Partial Kernel Convolution (PKConv) operation is designed to tackle these two problems. More details about PKConv are provided in the next section. The third problem, as evidenced by Equation 1, is that later features (with larger s) are computed by repeatedly applying the same operation, which decreases the representation diversity. Drawing inspiration from RNNs, we include a ConvGRU (Ballas et al. 2016; Cho et al. 2014) to modulate the information flow and encourage each scale level to focus on different parts of the data. Our spatial encoder can then be defined in two steps:

$$\begin{aligned} H_s &= \text{ConvGRU}(H_{s-1}, X_{s-1}), \\ X_s &= \phi(H_s). \end{aligned} \tag{2}$$

The ConvGRU follows the standard definition:

$$\begin{aligned} R_s &= \sigma(\phi([H_{s-1}, X_{s-1}])), \\ Z_s &= \sigma(\phi([H_{s-1}, X_{s-1}])), \\ N_s &= \tanh(\phi([R_s \odot H_{s-1}, X_{s-1}])), \\ H_s &= (1 - Z_s) \odot N_s + Z_s \odot H_{s-1}, \end{aligned} \tag{3}$$

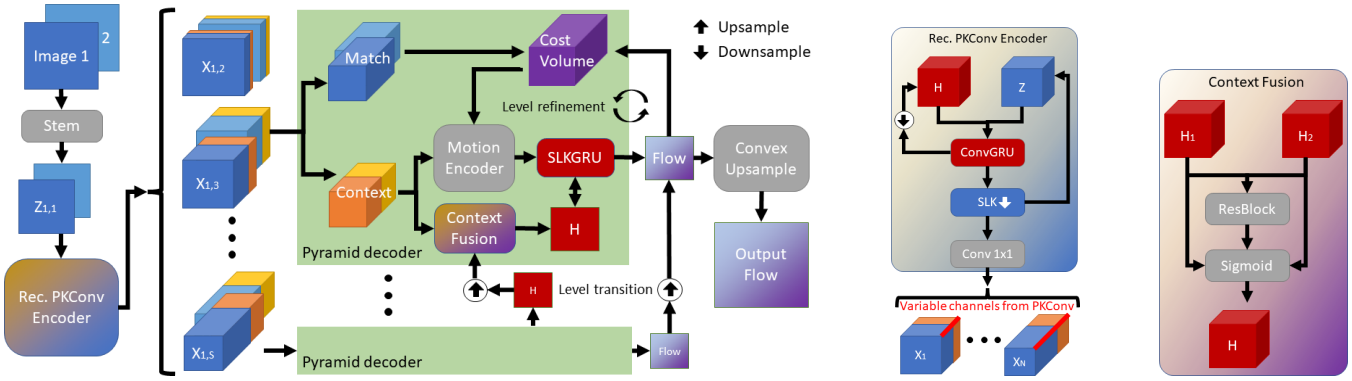

Figure 2: Overview of our model. (a) Input images are projected into feature space Z by a convolution stem and then converted into multi-scale features X by our proposed spatial recurrent encoder. (b) By combining PKConv layers with SLK modules, our encoder produces discriminative features of variable sizes with a single shared block. The pyramid decoder iteratively refines the flow estimation and uses a context fusion module (c) to transfer the hidden SLK GRU features H across pyramid levels. Best viewed in colors.

where $[\cdot]$ indicates concatenation, $\odot$ the Hadamard product, and σ the sigmoid function.

A representation of our complete encoder is shown in Figure 2, and it can be described as:

$$\begin{aligned} Z_{i,1} &= \phi_{\text{stem}}(I_i), \\ H_{i,s} &= \text{ConvGRU}^{\text{PK}}(H_{i,s-1}, Z_{i,s-1}), \\ Z_{i,s} &= \phi^{\text{PK}}_{\text{SLK}}(H_{i,s}), \\ X_{i,s} &= \phi^{\text{PK}}_{1\times1}(Z_{i,s}), \end{aligned} \tag{4}$$

where PK indicates PKConv layers, SLK is our proposed module, ϕ_{stem} is a single convolution layer with kernel size 7 and stride 2, and the final features $X_{i,s}$ are produced by a 1×1 convolution. We set $s \geq 2$ and initialize $H_{i,1} = \mathbf{0}$.

Partial Kernel Convolution (PKConv)

The standard convolution operation can be defined as follows. Let $X \in \mathcal{R}^{h\times w\times c^i}$ be a $2D$ feature map of size $h \times w$ and c^i channels, $K \in \mathcal{R}^{c^o\times c^i\times k^h\times k^w}$ a $2D$ convolution kernel of size $k^h \times k^w$ with c^o output and c^i input channels. The convolution of X(with appropriate padding) with K generates a resulting feature map $Y \in \mathcal{R}^{h\times w\times c^o}$ as:

$$\begin{aligned} Y(i,j,k) &= (X * K)(i,j,k) = \\ &\sum_{m=0}^{k^h-1}\sum_{n=0}^{k^w-1}\sum_{p=0}^{c^i-1} \\ &\quad (X(i+\tilde{m}, j+\tilde{n}, p)K(k,p,m,n)), \end{aligned} \tag{5}$$

where $k \leq c^o$ and $\tilde{m}, \tilde{n}$ indicate subtracting half of the kernel size, i.e., $\tilde{m} = m - \lfloor \frac{k^h}{2} \rfloor$.

To reduce the model size and improve generalization, some models reuse the same convolution block multiple times (Jin et al. 2023; Sim, Oh, and Kim 2021; Teed and Deng 2020). However, the standard convolution operation requires the input and output to always have the same predefined number of channels, which hinders the ability to handle diverse inputs. We propose to overcome this problem by employing a generalization of standard convolution that we refer to as Partial Kernel Convolution (PKConv). In PKConv, the convolution kernel K is treated as a buffer that can be partially sampled according to the layer requirements. Hence, if we have inputs and outputs $\hat{X} \in \mathcal{R}^{h\times w\times \hat{c}^i}$ and $\hat{Y} \in \mathcal{R}^{h\times w\times \hat{c}^0}$ with $\hat{c}^i \leq c^i$ and $\hat{c}^o \leq c^o$, we define the PKConv operation as:

$$\begin{aligned} \hat{Y}(i,j,k) &= (\hat{X} *^{PK} K)(i,j,k) = \\ &\sum_{m\in\hat{\mathbf{M}}}\sum_{n\in\hat{\mathbf{N}}}\sum_{p=0}^{\hat{c}^i-1} \\ &\quad (\hat{X}\,(i+\tilde{m}, j+\tilde{n}, p)\,K(\hat{\mathbf{O}}(k), \hat{\mathbf{I}}(p), m, n)), \end{aligned} \tag{6}$$

where the bold letters indicate a set of sampling indices with cardinalities $|\hat{\mathbf{I}}| = \hat{c}^i$, $|\hat{\mathbf{O}}| = \hat{c}^o$, $|\hat{\mathbf{M}}| \leq k^h$, and $|\hat{\mathbf{N}}| \leq k^w$.

Our RPKNet model does not selectively sample the kernel size dimensions (k^h, k^w) and always uses the full extent. We sample the first element for the channels until we reach the required number for each layer. More formally, the sampling sets in our PKConv encoder at scale s are as follows: $\hat{\mathbf{I}} = (0, ..., \hat{c}^i_s - 1)$, $\hat{\mathbf{O}} = (0, ..., \hat{c}^o_s - 1)$, $\hat{\mathbf{M}} = (0, ..., k^h - 1)$, and $\hat{\mathbf{N}} = (0, ..., k^w - 1)$, with $\hat{c}_s \geq \hat{c}_{s-1}$. This sampling strategy offers two advantages: (1) balance computational cost and information storage by increasing the channels as we decrease the spatial dimensions, and (2) allow early weights to learn more global multi-scale patterns and later weights to focus on the high-level features only. A visual example of the PKConv operation is shown in Figure 3.

Partial normalization Most deep networks employ feature normalization layers (Ba, Kiros, and Hinton 2016; Ioffe and Szegedy 2015; Wu and He 2018) to improve training

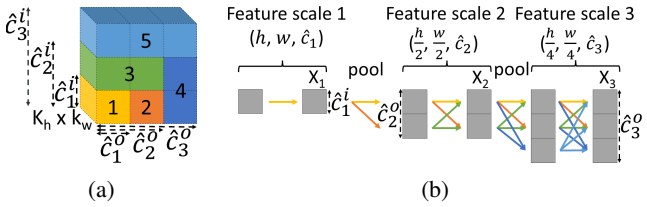

Figure 3: Example of multiple kernel samplings when using PKConv. (a) The entire volume represents the full convolution kernel, and different parts are sampled according to the feature scale. At a given scale s, the kernel will sample the first $\hat{c}_s$ values in the feature input and output dimensions (Y and X axes, respectively), with $\hat{c}_{s+1} \geq \hat{c}_s$. b) Visual representation of the sampled weights used at each scale level. Weights from subset 1 are shared among all levels and learn scale-agnostic patterns. Best viewed in colors.

and generalization. However, only some standard normalization techniques are compatible with the PKConv operation. In particular, strategies that normalize over all channels, such as Layer Normalization (Ba, Kiros, and Hinton 2016), are unsuitable since only a subset of channels may be used each time. We also experimentally found that Batch Normalization (Ioffe and Szegedy 2015) does not produce reliable results after PKConv layers, indicating that the channel statistics change according to the sampling sets. Group Normalization (Wu and He 2018) is compatible with PKConv since the channel groups can be synchronized with the sampling sets. Therefore, we adopt Group Normalization for PKConv layers.

Separable Large Kernel (SLK)

Recently, models with large kernel convolution have shown promising results in multiple applications (Ding et al. 2022a; Guo et al. 2023; Sun et al. 2022b). We further improve the efficiency of previous methods by utilizing only 1D large convolutions and skip connections. A diagram of our SLK module is shown in Figure 4. The main component of our module is integrated into the SLK Unit, and we complement it with residual scaling layers (Guo et al. 2023; Liu et al. 2022) and a Feed-Forward Network (Tan and Le 2021).

Skip connections It has been shown that using identity skip connection improves information flow and the performance of deep networks (He et al. 2016b; Veit, Wilber, and Belongie 2016). Based on these findings, we equip our SLK with skip connections of different lengths. In particular, we add redundant skip connections around each 1D large convolution block. This strategy, known as structural re-parameterization, has been shown to help the model during the training (Ding et al. 2021, 2022a). We adopt a similar design but simplify it by dropping the Batch Normalization (Ioffe and Szegedy 2015) layers. After training, these skip connections can be fused within the convolution layer (Ding et al. 2021).

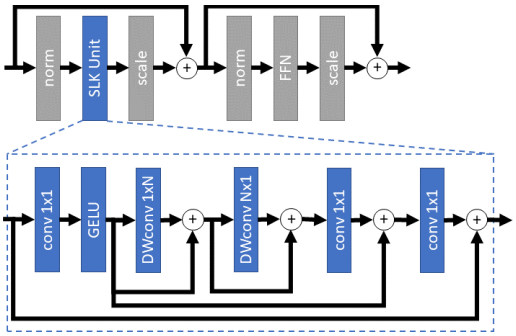

Figure 4: Overview of our SLK module. The main SLK Unit block only requires 1D large kernels and 1×1 convolution layers with skip connections.

Feature Pyramid Decoder

We adapt the iterative decoder from RAFT due to the success of iterative refinement strategies for fine-grained prediction tasks (Teed and Deng 2020; Zhang et al. 2023a). However, RAFT's decoder is designed for a single feature map, which is unsuitable for feature pyramids. Therefore, we add a context fusion module to handle the transition between pyramid levels. To account for the domain shift between levels, we forward the GRU state through a residual block and fuse it with more fine-grained context features from the encoder (Figure 2c).

Our encoder produces $S = 5$ level feature pyramid with feature strides 2^s. However, the decoder only processes the smallest three levels ($s \in \{3, 4, 5\}$) to reduce the processing costs. The total number of refinement iterations N is uniformly spread across the levels. For example, if $N = 12$, we first do four refinements at the smallest resolution, move to the next pyramid level, and proceed with the refinement. Predictions at the original resolution are produced using a convex upsampling layer (Teed and Deng 2020).

SLK GRU

The GRU is the main component of the decoder, as it aggregates all the input features and produces a central state feature, which is used to decode the optical flow and keep the history of previous iterations. Nonetheless, most optical flow models adopt a single ConvGRU (Ballas et al. 2016; Teed and Deng 2020) for this task. Unlike the original GRU (Cho et al. 2014), which encodes global relationships via fully connected layers, the ConvGRU extracts minimal context with a single 3×3 convolution layer for each gate. We improve the GRU by replacing the convolution layers with our SLK (equivalent to replacing ϕ with ϕ_{SLK} layers in Equation 3). The resulting SLK GRU requires even fewer parameters than the ConvGRU and significantly boosts the estimation quality.

Loss Function

Our loss function is similar to RAFT, consisting of the $L1$ difference between the predicted flow and the ground truth. The main difference is that we use different upsampling

Method	FLOPs	Mem.	Param.	KITTI 2015		Sintel clean		Sintel final		Spring
	(T)	(GB)	(M)	train	test	train	test	train	test	test
Ours	1.0	**2.4**	**2.8**	**13.0**	4.64	**1.12**	1.31	**2.45**	2.65	**4.80**
PWCNet (Sun et al. 2018b)	**0.7**	3.8	9.3	33.7	7.72	2.55	3.45	3.93	4.60	82.26
LiteFlowNet (Hui, Tang, and Loy 2018)	1.3	3.7	5.3	29.3	10.4	2.52	4.86	4.05	6.09	-
HD3 (Yin, Darrell, and Yu 2019)	1.3	4.2	39.5	23.9	6.50	3.84	4.79	8.77	4.67	-
MaskFlowNet (Zhao et al. 2020)	1.4	3.9	20.6	23.1	6.10	2.25	2.52	3.61	4.17	-
FlowNet 2 (Ilg et al. 2017)	1.6	6.5	162.5	28.2	11.4	2.02	3.96	3.14	6.02	-
Flow1D (Xu et al. 2021)	3.2	3.2	5.7	22.9	6.27	1.98	2.23	3.27	3.80	-
RAFT (Teed and Deng 2020)	3.8	10.8	5.1	17.4	5.10	1.44	1.60	2.71	2.85	6.79
GMFlowNet (Zhao et al. 2022)	5.0	21.9	9.3	15.4	4.79	1.14	1.39	2.71	2.65	-
GMA (Jiang et al. 2021a)	7.6	18.7	5.8	17.1	5.25	1.30	1.38	2.74	2.47	7.07
DIP (Zheng et al. 2022)	7.8	5.4	5.3	13.7	**4.21**	1.30	1.43	2.82	2.83	-
SKFlow (Sun et al. 2022b)	7.9	18.7	6.3	15.5	4.84	1.22	**1.29**	2.46	**2.27**	-
MatchFlow (Dong, Cao, and Fu 2023)	9.9	19.1	15.4	13.6	4.72	1.14	1.33	2.61	2.64	-

Table 1: Results with the official metrics on KITTI 2015 (Fl-All), MPI-Sintel (EPE), and Spring (1px) datasets. Train results are collected from models without finetuning, while test results come from the official benchmarks. The best results are highlighted in bold, while the second best are underlined. Memory and FLOP results are based on an input of (1920×1080) using each method's default number of iterations.

functions ψ_s for each pyramid level s. At the largest decoding pyramid level ($s = 3$), we use convex upsampling (Teed and Deng 2020), while the remaining level ($s > 3$) adopt a simple bilinear upsampling. If we let $N_l = \lceil \frac{N}{3} \rceil$ be the number of iterations per feature level, we can define the loss as:

$$\mathcal{L} = \sum_{s=5}^{3} \sum_{j=0}^{N_l-1} \gamma^{N-(5-s)N_l-j-1} \|F_{gt} - \psi_s(F_{s,j})\|_1, \quad (7)$$

where $\gamma < 1$ is a scalar to give less importance to early iterations.

Experimental Results

Implementation Details

We follow the same training routine as RAFT (Teed and Deng 2020), using the AdamW optimizer (Loshchilov and Hutter 2019) combined with the OneCycle learning rate schedule (Smith and Topin 2019). We use the extended training (Sun et al. 2022a; Xu et al. 2022) with 100k iterations on the FlyingChairs dataset (Dosovitskiy et al. 2015) followed by 1M iterations on the FlyingThings3D (Mayer et al. 2016). For the Sintel (Butler et al. 2012) and Spring (Mehl et al. 2023) benchmarks, we use 250k iterations to finetune the model in a mixed dataset combining FlyingThings3D, KITTI (Geiger, Lenz, and Urtasun 2012; Menze and Geiger 2015), HD1K (Kondermann et al. 2016), and Sintel samples. For KITTI, we start from the Sintel model and further finetune it in the KITTI 2015 dataset for 5k iterations. We use 12 refinement iterations for training and ablation experiments and 32 for the public benchmark evaluation. Each model is trained once using 1234 as the random seed on two NVIDIA RTX3090 GPUs. We calculate FLOPs using the PyTorch profiler.

Comparison With the State-of-the-Art

Table 1 shows how our results compare to the state-of-the-art. We compare our approach with recent models using up to ten times more FLOPs than ours.

Cross-dataset generalization on train sets We test our model generalization by evaluating it on the train sets from Sintel and KITTI 2015 after training on the FlyingThings3D dataset. Table 1 shows that our method outperforms all other methods on the train sets while having the smallest size and computational costs. These results indicate that larger models do not necessarily find better general patterns on the limited amount of training optical flow samples, further supporting the impact of model size on generalization performance (Zhou and Feng 2018). Even MatchFlow, which uses almost ten times more FLOPs, five times more parameters, and additional pretraining from the MegaDepth dataset (Li and Snavely 2018), still is outperformed by our method.

Cross-dataset generalization on the Spring test set Spring is a suitable benchmark for generalization tests since it provides high-resolution $1080p$ samples with $4K$ optical flow annotations, substantially different from other datasets used for training. To be fair to the current submissions, we do not finetune our model on samples from the Spring dataset; instead, we use the same model finetuned for the Sintel dataset. Our model ranks first among published methods without finetuning in the official benchmark and outperforms other very large models, such as MS-RAFT+ (Jahedi et al. 2022) and FlowFormer (Huang et al. 2022) (see Figure 1). Remarkably, our model has over five times fewer parameters and uses at least $15\times$ fewer FLOPs than those two models, further highlighting our generalization capability.

Finetuning test Following previous works, we finetuned our model on the Sintel and KITTI 2015 datasets and

Method	FLOP	Param.	KITTI 2015	Sintel clean	Sintel final
Ours	1.0	2.8	13.0	1.12	2.45
RAFT	3.8	5.1	17.4	1.44	2.71
FlowFormer-S	14.2	6.1	16.6	1.20	2.64
FlowFormer	17.2	16.0	14.7	0.95	2.35

Table 2: Comparison of generalization results of our method against other model reduction approaches.

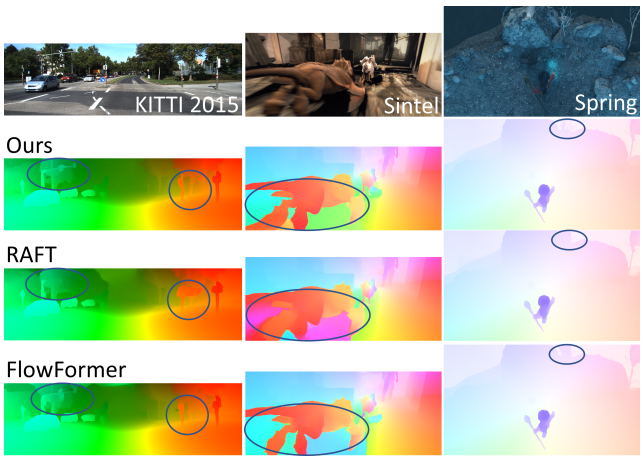

Figure 5: Sample qualitative results on the tested datasets. Best viewed in colors.

submitted our results to the official benchmarks. We also achieve strong performance on finetuning, outperforming most larger comparable models. On KITTI 2015, our model outperforms almost all other methods except for DIP. On the MPI-Sintel benchmark, other larger methods, such as SKFlow, show a more significant advantage. We believe this is caused by the extensive finetuning that most models use for this benchmark, which may contribute to larger models to fit a more specialized distribution for this task.

Model efficiency We compare the models by calculating their memory consumption, FLOPs count, and speed on large $1080p$ images (Spring size). Table 1 shows that our model has the lowest parameter count and memory consumption among all compared methods. It also requires at least three times fewer FLOPs than other top-performing methods. While methods like PWCNet use even fewer operations, our results are significantly more accurate than those. We also check the inference time for some models, with ours taking around 580 ms, offering a good compromise between RAFT[1] (890 ms) and Flow1D (375 ms).

Model Reduction Evaluation

Our model can be interpreted as a reduced version of RAFT (Teed and Deng 2020). Table 2 shows that, despite being smaller, our model outperforms RAFT in all tests. FlowFormer (Huang et al. 2022) also presented a reduced FlowFormer-S version, which still requires fourteen times more FLOPs than ours and is outperformed in all datasets. These results further demonstrate the advantages of our proposed efficient architecture for reducing computational costs without impacting performance.

[1]With the memory-efficient alternate correlation layer

Experiment	Method	Sintel clean	Sintel final	KITTI 2015
Encoder	ResNet	1.40	2.68	5.40
	PKConv	**1.28**	**2.60**	**4.23**
Decoder	2 ConvGRU	1.39	2.73	5.81
	2 SLK	1.35	2.68	4.55
	1 SLKGRU	1.35	2.61	4.68
	2 SLKGRU	**1.28**	**2.60**	**4.23**
Kernel size	3	1.39	2.67	4.88
	7	1.29	2.62	4.70
	15	1.34	2.64	4.48
	23	**1.28**	**2.60**	**4.23**
	31	1.32	**2.60**	4.81
Re-param.	No	1.32	**2.58**	4.58
	Yes	**1.28**	2.60	**4.23**
Feat/Ctxt net	Individual	1.34	2.64	4.48
	Joint	**1.28**	**2.60**	**4.23**

Table 3: Ablation results (EPE) of the different modules. The underline indicates the option adopted by our model.

Qualitative Results

We qualitatively compare our model against the RAFT baseline and the large attention model FlowFormer in Figure 5. As the Sintel example shows, the large kernels of our SLK can capture long-range context to enforce consistency between disconnected regions. FlowFormer can also leverage its global attention with a similar effect, but our approach has a much lower computational cost. Our model also preserves fine details and produces accurate estimations for smaller objects, outperforming the competing methods in many cases. On the other hand, the larger FlowFormer structure still generally produces slightly sharper results and shows more robustness to adverse image conditions, especially on the Sintel dataset.

Ablation Study

We conduct ablation studies about the different modules we propose in our method. Table 3 shows the results using a training schedule of 250k iterations on the FlyingThings3D dataset. More details are provided below.

PKConv encoder We compare our encoder against a similarly-sized ResNet (He et al. 2016a). Our recurrent encoder with PKConv layers and SLK blocks shows a noticeable improvement in all benchmarks.

Decoder unit We check how our SLK blocks compare to the commonly used ConvGRU (Ballas et al. 2016; Teed and Deng 2020). Similar to SKFlow (Sun et al. 2022b), replacing the ConvGRUs with our SLK module (without GRU) already brings a noticeable improvement to the results. However, using SLK GRUs improves the results even further, demonstrating the benefits of GRUs for iterative tasks.

Kernel size We experiment with 1D kernels of different sizes for the SLK module. Our SLK module structure is robust; even a small kernel provides competitive results. However, increasing the size of the kernel improves the results, achieving the best performance with a kernel size of 23.

Re-parameterization We use structural re-parameterization by adding skip connections around our 1D convolution layers. The results show that this strategy benefits our model and improves the training.

Joint encoder Our model replaces the traditional dual encoder design (Teed and Deng 2020) with a single joint encoder. Our results show that the joint design improves estimation quality.

Related Works

Optical Flow

FlowNet (Dosovitskiy et al. 2015), especially FlowNet2 (Ilg et al. 2017), set a new standard using deep networks to estimate optical flow. Early deep models for optical flow estimation were primarily based on the traditional coarse-to-fine strategy and employed feature pyramids to gradually refine the estimation (Hofinger et al. 2020; Hui, Tang, and Loy 2018; Ranjan and Black 2017; Sun et al. 2018b). Later, RAFT (Teed and Deng 2020) proposed to remove the pyramid and work with a single high-resolution feature map, creating a new standard for optical flow models. Multiple advances have been proposed since then to improve the cost volume representation (Xiao et al. 2020; Zhang et al. 2021), leverage temporal cues (Chen et al. 2023; Ferede and Balasubramanian 2023; Lu et al. 2023; Shi et al. 2023a), and improve features (Dong, Cao, and Fu 2023; Luo et al. 2022; Shi et al. 2023b; Sui et al. 2022). These improvements, however, also contributed to an increase in the computational cost of the models. Some recent works (Jiang et al. 2021b; Kong, Shen, and Yang 2021; Xu et al. 2021; Zheng et al. 2022) have been focusing on decreasing the cost of optical flow models, but they impose a noticeable drop in accuracy or running speed.

Recurrent Models for Optical Flow

Current optical flow models often use a recurrent decoder. IRR (Hur and Roth 2019) first used a recurrent CNN decoder, while most current models (Huang et al. 2022; Jiang et al. 2021a; Zheng et al. 2022) uses a GRU-based block (Teed and Deng 2020) to refine the prediction iteratively. Recurrent units have been adopted in encoders as well in the temporal dimension to handle multiple frames of videos (Ding et al. 2022b; Lu et al. 2023). However, recurrent encoders in the spatial dimension have yet to be well-explored for optical flow. Some works adopted it for video interpolation to generate multi-scale features, but that required either producing features of a fixed size (Sim, Oh, and Kim 2021) or treating the entire optical flow network as a recurrent unit (Jin et al. 2023; Zhang, Zhao, and Wang 2020), which decreases the flexibility of the model.

Large Context Networks

Considering large contexts is often beneficial to optical flow estimation. Dilated convolution (Sun et al. 2018a) has been proposed to capture more long-range relationships. More recently, it has become standard to add attention layers (Vaswani et al. 2017) to improve feature representation (Du et al. 2022; Sui et al. 2022; Xu et al. 2021, 2022; Zhao et al. 2022; Zheng et al. 2023), but this increases the computational cost significantly. More recently, works in other fields have shown that large kernel convolution can outperform Transformer models at a lower computational cost (Ding et al. 2022a; Guo et al. 2022). SKFlow (Sun et al. 2022b) has also shown that large convolution layers produce state-of-the-art optical flow estimation results. However, SKFlow is built on top of GMA (Jiang et al. 2021a), a global attention model, and thus still imposes a high computational cost.

Convolution Operations

Variations of convolution and channel manipulation operations have been proposed before for different purposes. Partial Convolution (Liu et al. 2018) was proposed as an alternative to feature padding to ignore some parts of the kernel by masking the areas out of the image boundaries. However, it ignored changes in the number of channels and increased the convolution costs by calculating masks. Sparse Convolution (Engelcke et al. 2016) can also select parts of the kernel. However, the kernel sampling is guided by the sparsity of the 3D cloud point inputs to save computation in empty areas. This is a different motivation than our PKConv, where each part of the kernel encodes different types of information. Convolution with adaptable kernels has been explored to handle more diverse features (Jia et al. 2016; Zhang et al. 2023b), but these methods require complementary networks to generate the kernels.

Discussion

We have proposed RPKNet, an efficient optical flow model that leverages recurrent Partial Kernel Convolution layers (PKConv) and Separable Large Kernels (SLK) to produce high-quality optical flow estimation at a lower computational cost. Our model has shown outstanding generalization capability by achieving the best performance without finetuning, using fewer parameters and less memory than all competing models.

Limitations It has been shown (Ding et al. 2022a) that current software and hardware are not optimized for large kernel operations. Also, if computational resources are not a concern, large and more expensive methods like FlowFormer (Huang et al. 2022) can still outperform ours on standard benchmarks.

Acknowledgments

This research is supported by the National Key Research and Development Program of China (2020AAA0109700), the Beijing Natural Science Foundation (IS23060), the Fundamental Research Funds for the Central Universities (FRF-TP-22-048A1), the National Science Fund for Distinguished Young Scholars(62125601), and the National Natural Science Foundation of China (62076024, 62172035).

References

Ba, J.; Kiros, J. R.; and Hinton, G. E. 2016. Layer Normalization. *ArXiv*.

Ballas, N.; Yao, L.; Pal, C. J.; and Courville, A. C. 2016. Delving Deeper into Convolutional Networks for Learning Video Representations. In *ICLR*.

Butler, D. J.; Wulff, J.; Stanley, G. B.; and Black, M. J. 2012. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 611–625.

Chen, Y.; Zhu, D.; Shi, W.; Zhang, G.; Zhang, T.; Zhang, X.; and Li, J. 2023. MFCFlow: A Motion Feature Compensated Multi-Frame Recurrent Network for Optical Flow Estimation. In *WACV*, 5057–5066.

Cho, K.; van Merrienboer, B.; Çaglar Gülçehre; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *EMNLP*, 1724–1734.

Ding, X.; Zhang, X.; Ma, N.; Han, J.; Ding, G.; and Sun, J. 2021. RepVGG: Making VGG-style ConvNets Great Again. In *CVPR*, 13728–13737.

Ding, X.; Zhang, X.; Zhou, Y.; Han, J.; Ding, G.; and Sun, J. 2022a. Scaling Up Your Kernels to 31×31: Revisiting Large Kernel Design in CNNs. In *CVPR*, 11953–11965.

Ding, Z.; Zhao, R.; Zhang, J.; Gao, T.; Xiong, R.; Yu, Z.; and Huang, T. 2022b. Spatio-Temporal Recurrent Networks for Event-Based Optical Flow Estimation. In *AAAI*, 525–533.

Dong, Q.; Cao, C.; and Fu, Y. 2023. Rethinking Optical Flow from Geometric Matching Consistent Perspective. In *CVPR*, 1337–1347.

Dosovitskiy, A.; Fischer, P.; Ilg, E.; Hausser, P.; Hazirbas, C.; Golkov, V.; Van Der Smagt, P.; Cremers, D.; and Brox, T. 2015. FlowNet: learning optical flow with convolutional networks. In *ICCV*, 2758–2766.

Du, Y.; Chen, Z.; Jia, C.; Yin, X.; Zheng, T.; Li, C.; Du, Y.; and Jiang, Y.-G. 2022. SVTR: Scene Text Recognition with a Single Visual Model. In *IJCAI*, 884–890.

Engelcke, M.; Rao, D.; Wang, D. Z.; Tong, C. H.; and Posner, I. 2016. Vote3Deep: Fast object detection in 3D point clouds using efficient convolutional neural networks. In *ICRA*, 1355–1361.

Ferede, F. A.; and Balasubramanian, M. 2023. SSTM: Spatiotemporal Recurrent Transformers for Multi-frame Optical Flow Estimation. *ArXiv*.

Geiger, A.; Lenz, P.; and Urtasun, R. 2012. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *CVPR*, 3354–3361.

Guo, M.-H.; Lu, C.; Hou, Q.; Liu, Z.; Cheng, M.-M.; and Hu, S. 2022. SegNeXt: Rethinking Convolutional Attention Design for Semantic Segmentation. In *NeurIPS*.

Guo, M.-H.; Lu, C.-Z.; Liu, Z.-N.; Cheng, M.-M.; and Hu, S.-M. 2023. Visual Attention Network. *Computational Visual Media*, 9(4).

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep Residual Learning for Image Recognition. *CVPR*, 770–778.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Identity mappings in deep residual networks. In *ECCV*, 630–645.

Hofinger, M.; Bulò, S. R.; Porzi, L.; Knapitsch, A.; and Kontschieder, P. 2020. Improving optical flow on a pyramidal level. In *ECCV*, 770–786.

Huang, Z.; Shi, X.; Zhang, C.; Wang, Q.; Cheung, K. C.; Qin, H.; Dai, J.; and Li, H. 2022. FlowFormer: A Transformer Architecture for Optical Flow. In *ECCV*, 668–685.

Hui, T.-W.; Tang, X.; and Loy, C. C. 2018. LiteFlowNet: a lightweight convolutional neural network for optical flow estimation. In *CVPR*, 8981–8989.

Hur, J.; and Roth, S. 2019. Iterative residual refinement for joint optical flow and occlusion estimation. In *CVPR*, 5754–5763.

Ilg, E.; Mayer, N.; Saikia, T.; Keuper, M.; Dosovitskiy, A.; and Brox, T. 2017. FlowNet 2.0: evolution of optical flow estimation with deep networks. In *CVPR*, volume 2, 2462–2470.

Ioffe, S.; and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*, 448–456.

Jahedi, A.; Mehl, L.; Rivinius, M.; and Bruhn, A. 2022. Multi-Scale RAFT: Combining Hierarchical Concepts for Learning-based Optical Flow Estimation. In *ICIP*, 1236–1240.

Jia, X.; Brabandere, B. D.; Tuytelaars, T.; and Gool, L. V. 2016. Dynamic Filter Networks. In *NeurIPS*, 667–675.

Jiang, S.; Campbell, D.; Lu, Y.; Li, H.; and Hartley, R. 2021a. Learning to Estimate Hidden Motions with Global Motion Aggregation. In *ICCV*, 9752–9761.

Jiang, S.; Lu, Y.; Li, H.; and Hartley, R. 2021b. Learning Optical Flow From a Few Matches. In *CVPR*, 16592–16600.

Jin, X.; Wu, L.; Shen, G.; Chen, Y.; Chen, J.; Koo, J.; and hee Hahm, C. 2023. Enhanced Bi-directional Motion Estimation for Video Frame Interpolation. In *WACV*, 5038–5046.

Kondermann, D.; Nair, R.; Honauer, K.; Krispin, K.; Andrulis, J.; Brock, A.; Gussefeld, B.; Rahimimoghaddam, M.; Hofmann, S.; Brenner, C.; et al. 2016. The HCI Benchmark Suite: Stereo and Flow Ground Truth With Uncertainties for Urban Autonomous Driving. In *CVPR Workshops*, 19–28.

Kong, L.; Shen, C.; and Yang, J. 2021. FastFlowNet: A Lightweight Network for Fast Optical Flow Estimation. In *ICRA*.

Li, Z.; and Snavely, N. 2018. MegaDepth: Learning Single-View Depth Prediction from Internet Photos. In *CVPR*, 2041–2050.

Liu, G.; Reda, F. A.; Shih, K. J.; Wang, T.-C.; Tao, A.; and Catanzaro, B. 2018. Image Inpainting for Irregular Holes Using Partial Convolutions. In *ECCV*, 89–105.

Liu, Z.; Mao, H.; Wu, C.-Y.; Feichtenhofer, C.; Darrell, T.; and Xie, S. 2022. A ConvNet for the 2020s. In *CVPR*, 11966–11976.

Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. In *ICLR*.

Lu, Y.; Wang, Q.; Ma, S.; Geng, T.; Chen, V. Y.; Chen, H.; and Liu, D. 2023. TransFlow: Transformer as Flow Learner. In *CVPR*, 18063–18073.

Luo, A.; Yang, F.; Li, X.; and Liu, S. 2022. Learning Optical Flow with Kernel Patch Attention. In *CVPR*, 8896–8905.

Mayer, N.; Ilg, E.; Häusser, P.; Fischer, P.; Cremers, D.; Dosovitskiy, A.; and Brox, T. 2016. A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. In *CVPR*, 4040–4048.

Mehl, L.; Schmalfuss, J.; Jahedi, A.; Nalivayko, Y.; and Bruhn, A. 2023. Spring: A High-Resolution High-Detail Dataset and Benchmark for Scene Flow, Optical Flow and Stereo. In *CVPR*, 4981–4991.

Menze, M.; and Geiger, A. 2015. Object scene flow for autonomous vehicles. In *CVPR*, 3061–3070.

Ranjan, A.; and Black, M. J. 2017. Optical flow estimation using a spatial pyramid network. In *CVPR*, 4161–4170.

Shi, X.; Huang, Z.; Bian, W.; Li, D.; Zhang, M.; Cheung, K. C.; See, S.; Qin, H.; Dai, J.; and Li, H. 2023a. VideoFlow: Exploiting Temporal Cues for Multi-frame Optical Flow Estimation. In *ICCV*, 12469–12480.

Shi, X.; Huang, Z.; Li, D.; Zhang, M.; Cheung, K. C.; See, S.; Qin, H.; Dai, J.; and Li, H. 2023b. FlowFormer++: Masked Cost Volume Autoencoding for Pretraining Optical Flow Estimation. In *CVPR*, 1599–1610.

Sim, H.; Oh, J.; and Kim, M. 2021. XVFI: eXtreme Video Frame Interpolation. In *ICCV*, 14469–14478.

Smith, L. N.; and Topin, N. 2019. Super-convergence: very fast training of neural networks using large learning rates. In *Defense + Commercial Sensing*.

Sui, X.; Li, S.; Geng, X.; Wu, Y.; Xu, X.; Liu, Y.; Goh, R. S. M.; and Zhu, H. 2022. CRAFT: Cross-Attentional Flow Transformer for Robust Optical Flow. In *CVPR*, 17581–17590.

Sun, D.; Herrmann, C.; Reda, F.; Rubinstein, M.; Fleet, D. J.; and Freeman, W. T. 2022a. Disentangling Architecture and Training for Optical Flow. In *ECCV*, 165–182.

Sun, D.; Yang, X.; Liu, M.-Y.; and Kautz, J. 2018a. Models matter, so does training: an empirical study of CNNs for optical flow estimation. *TPAMI*, 42(6): 1408–1423.

Sun, D.; Yang, X.; Liu, M.-Y.; and Kautz, J. 2018b. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *CVPR*, 8934–8943.

Sun, S.; Chen, Y.; Zhu, Y.; Guo, G.; and Li, G. 2022b. SKFlow: Learning Optical Flow with Super Kernels. In *NeurIPS*.

Tan, M.; and Le, Q. V. 2021. EfficientNetv2: smaller models and faster training. In *ICML*.

Teed, Z.; and Deng, J. 2020. RAFT: recurrent all-pairs field transforms for optical flow. In *ECCV*, 402–419.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention Is All You Need. In *NeurIPS*, 6000–6010.

Veit, A.; Wilber, M. J.; and Belongie, S. J. 2016. Residual Networks Behave Like Ensembles of Relatively Shallow Networks. In *NeurIPS*.

Weinzaepfel, P.; Arora, V.; Cabon, Y.; Lucas, T.; Brégier, R.; Leroy, V.; Csurka, G.; Antsfeld, L.; Chidlovskii, B.; and Revaud, J. 2022. Improved Cross-view Completion Pre-training for Stereo Matching. *ArXiv*.

Wu, Y.; and He, K. 2018. Group normalization. In *ECCV*, 1–17.

Xiao, T.; Yuan, J.; Sun, D.; Wang, Q.; Zhang, X.; Xu, K.; and Yang, M.-H. 2020. Learnable Cost Volume Using the Cayley Representation. In *ECCV*, 483–499.

Xu, H.; Yang, J.; Cai, J.; Zhang, J.; and Tong, X. 2021. High-Resolution Optical Flow from 1D Attention and Correlation. In *ICCV*, 10478–10487.

Xu, H.; Zhang, J.; Cai, J.; Rezatofighi, H.; and Tao, D. 2022. GMFlow: Learning Optical Flow via Global Matching. In *CVPR*, 8121–8130.

Yin, Z.; Darrell, T.; and Yu, F. 2019. Hierarchical Discrete Distribution Decomposition for Match Density Estimation. In *CVPR*, 6037–6046.

Zhang, F.; Woodford, O. J.; Prisacariu, V. A.; and Torr, P. H. S. 2021. Separable Flow: Learning Motion Cost Volumes for Optical Flow Estimation. In *ICCV*, 10787–10797.

Zhang, H.; Zhao, Y.; and Wang, R. 2020. A Flexible Recurrent Residual Pyramid Network for Video Frame Interpolation. In *ECCV*, 474–491.

Zhang, S.-X.; Zhu, X.; Chen, L.; Hou, J.-B.; and Yin, X.-C. 2023a. Arbitrary Shape Text Detection via Segmentation With Probability Maps. *TPAMI*, 45: 2736–2750.

Zhang, S.-X.; Zhu, X.; Hou, J.-B.; Yang, C.; and Yin, X.-C. 2023b. Kernel Proposal Network for Arbitrary Shape Text Detection. *IEEE Transactions on Neural Networks and Learning Systems*, 34: 8731–8742.

Zhao, S.; Sheng, Y.; Dong, Y.; Chang, E. I.; Xu, Y.; et al. 2020. MaskFlownet: asymmetric feature matching with learnable occlusion mask. In *CVPR*, 6277–6286.

Zhao, S.; Zhao, L.; Zhang, Z.; Zhou, E.; and Metaxas, D. N. 2022. Global Matching with Overlapping Attention for Optical Flow Estimation. In *CVPR*, 17571–17580.

Zheng, T.; Chen, Z.; Fang, S.; Xie, H.; and Jiang, Y.-G. 2023. CDistNet: Perceiving Multi-Domain Character Distance for Robust Text Recognition. *IJCV*, 1–19.

Zheng, Z.; Nie, N.; Ling, Z.; Xiong, P.; Liu, J.; Wang, H.; and Li, J. 2022. DIP: Deep Inverse Patchmatch for High-Resolution Optical Flow. In *CVPR*, 8915–8924.

Zhou, P.; and Feng, J. 2018. Understanding Generalization and Optimization Performance of Deep CNNs. In *ICML*.