SKFlow: Optical Flow Estimation Using Selective Kernel Networks

MINGLIANG ZHAI\textsuperscript{1}, XUEZHI Xiang\textsuperscript{2}, (Member, IEEE), NING LV\textsuperscript{1}, SYED MASROOR ALI\textsuperscript{1}, AND ABDULMOTALEB EL SADDIK\textsuperscript{2}, (Fellow, IEEE)

\textsuperscript{1}The School of Information and Communication Engineering, Harbin Engineering University, Harbin 150001, China
\textsuperscript{2}The School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON K1N 6N5, Canada

Corresponding author: Xuezhi Xiang (xiangxuezhi@hrbeu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61401113, in part by the Natural Science Foundation of Heilongjiang Province of China under Grant LC201426, in part by the Fundamental Research Funds for the Central Universities of China under Grant 3072019CF0801, and in part by the Ph.D. Student Research and Innovation Fund of the Fundamental Research Funds for the Central Universities under Grant 3072019GIP0807.

\textbf{ABSTRACT} Leveraging on the recent developments in convolutional neural networks (CNNs), optical flow estimation from adjacent frames has been cast as a learning problem, with performance exceeding traditional approaches. The existing networks always use standard convolutional layers for extracting multi-level features with the fixed kernel size at each level. For enlarging the receptive field, some works introduce dilated convolution operation, which can capture more contextual information and can avoid the loss of motion details. However, these networks lack the ability to adaptively adjust its receptive field size and cannot aggregate multi-scale information with a selective mechanism. To address this problem, in this paper, we introduce selective kernel network into optical flow estimation, which can adaptively select different scale features and adjust their receptive field according to the global information. Specifically, we conduct the selective kernel mechanism on each level of pyramid, which can adaptively select multi-scale feature at each pyramidal level. The extensive analyses are conducted on MPI-Sintel and KITTI datasets to verify the effectiveness of the proposed approach. The experimental results show that our model achieves comparable results with the previous state-of-the-art networks while keeping a small model size.

\textbf{INDEX TERMS} CNNs, optical flow, selective kernel, multi-scale feature, receptive field.

I. INTRODUCTION

Optical flow estimation is a fundamental technique for numerous computer vision applications like visual tracking, action recognition and autonomous driving. Traditional approaches [1]–[5], also can be called knowledge-driven approaches, usually formulate optical flow as an energy function optimization problem, which design prior various constraints by considering prior knowledge. These methods dominate benchmark for many years. Nevertheless, optimizing such complex function are usually time consuming, and are too slow to be applied in real-time scene.

With the recent development of CNNs, remarkable progress has made in optical flow estimation. Compared to knowledge-driven approaches, CNNs-based methods have powerful learning ability to purify knowledge from large amount of data. Thus, these methods can be called data-driven approaches. Many methods [6]–[10] adopt encoder-decoder or spatial pyramid architectures for learning optical flow. FlowNet [6] is the first end-to-end network for learning optical flow, which adopts the encoder-decoder architecture. Based on [6], FlowNet2.0 further stacks several sub-networks to form a large network, which greatly boosts the accuracy of FlowNet. Despite FlowNet and FlowNet2.0 can produce reasonable and high-quality flow map, the model size is very large for real-world application.

To address this issue, several networks that have smaller model size than encoder-decoder architectures are proposed in [8], [10]–[12]. SpyNet [8] adopt spatial pyramid network for optical flow estimation, which conducts image warping at each level to disassemble large displacement into small displacement. Therefore, each pyramidal level only needs to calculate small displacement, which reduces the number of channels significantly. The SpyNet achieves smaller model...
size than FlowNet [6]. Based on [8], PWC-Net [11] and LiteFlowNet [10] use feature warping instead of image warping that reduces the motion distance on feature space directly. Specifically, PWC-Net [11] adopts cost volume for matching feature after feature warping. PWC-Net and LiteFlowNet achieve the state-of-the-art results among these light weight networks and obtain competitive accuracy compared to the heavy weight network, FlowNet2.0. Though the promising performance has been achieved, these methods only use standard convolution to extract features with the fixed receptive field, which hinders the capturing of rich contextual information. Especially, multi-scale contextual information is important for flow estimation. Recently, Zhai et al. [9] introduce dilated convolution into optical flow estimation and combine it with residual network, which not only can enlarge the receptive field to capture more detailed spatial information but also can decrease the loss of motion details. Nevertheless, existing networks exploit multi-scale features only by using different convolutional kernel or dilated convolution with different rate. Furthermore, these extracted multi-scale information can not be adaptively aggregated and selected.

Most recently, Li et al. [13] propose a novel network for image classification, called Selective Kernel Network, which can adaptively select receptive field and multi-scale features. The selective kernel mechanism mainly contains three parts, split, fusion and select. Firstly, the split component extracts initial multi-scale features with the different dilated rate. Then, the fusion part combines and incorporates multi-scale information, and calculates the selective weights. Finally, the select component integrates the information of different scale. The adaptive receptive fields are beneficial for feature learning and can exploit rich contextual information and details. These characteristics are useful for pixel-wise tasks, such as optical flow estimation. Attempting to incorporate the advantages of spatial pyramid network and selective kernel module, we propose to adaptively integrate and select different scale feature at each pyramidal level by introducing the selective kernel module.

In this paper, we propose a novel selective kernel pyramid network for optical flow estimation, which can integrate multi-scale contextual information and adaptively adjust the receptive field to select useful kernels. In contrast to [13], we introduce the selective kernel mechanism into optical flow estimation and combine this module with spatial pyramid network. Furthermore, we design a cascaded selective flow estimator, which not only can generate multi-scale features but also can select the features of different scales through the self-adjustment receptive field. The rich scale and contextual information are adaptively encoded at each pyramidal level, which provides the selected multi-scale features for estimating more refined optical flow. In addition, to further improve the accuracy of flow estimation, we take advantages of feature warping and cost volume before encoding features at each level.

In summary, the main contributions of our proposed approach can be summarized as follows.

- We propose a novel selective kernel pyramid network for optical flow estimation, which can adaptively adjust the receptive field and select multi-scale features. To the best of our knowledge, we are the first to exploit the effectiveness of selective kernel mechanism for optical flow estimation.
- We use selective kernel module at each pyramidal level and extend original selective kernel mechanism to pyramidal selective kernel network.
- Extensive experiments on two publicly available datasets for optical flow evaluation, MPI-Sintel and KITTI demonstrate the effectiveness of the combination of spatial pyramid network and selective kernel network.

The follow sections are organized as follows. In section-2, we mainly review existing approaches for optical flow estimation and feature fusion networks. In section-3, we first introduce the selective kernel network and then describe the entire framework of our proposed approach. In section-4, we report the experimental results on public datasets in detail. In the last section, we give a conclusion of our method.

II. RELATED WORK

In this section, we first introduce the knowledge-driven approaches briefly. Then, we describe the data-driven approaches in detail. Finally, we introduce feature fusion networks used for computer vision tasks.

A. KNOWLEDGE-DRIVEN APPROACH

Early approaches for optical flow estimation usually involve the assumptions of brightness constancy and local smoothness, which typically models an energy function for optimization. Horn and Schunck [1] first formulate an energy function based on brightness constancy and spatial smoothness. However, the work [1] can only handle small displacement, and tends to fail when displacements are large. To address large displacement problem, Brox et al. [3] propose to use coarse-to-fine strategy, which conducts image warping operation and only needs to calculate the small displacement at each pyramidal level. Further, Xu et al. [14] introduce an extended coarse-to-fine scheme, which can reduce the reliance of initial flow values and can handle non-rigid motion with dense patch matching. Brox and Malik [4] integrate feature matching to address large displacements. Specially, they introduce a novel matching term into energy function. Bao et al. [15] present a fast edge-preserving patchmatch approach for large displacement optical flow, which can randomly propagate the self-similarity patterns and correspondence offsets. Tu et al. [16] propose local intensity fusion method for optical flow estimation, which fuses flow vectors with different approaches and different smoothness parameter settings. This operation is beneficial for handling large displacements. Hu et al. [17] present an efficient random search approach with the coarse-to-fine strategy for dealing with large displacements problem. Revaud et al. [18] propose a new edge-aware interpolation approach to interpolate dense
flow from sparse marches and use energy minimization to post-processing flow fields. Yang and Li [19] propose a piecewise parametric model for energy optimization, which uses super pixel segmentation to constrain flow fields. To solve the motion occlusion problem, Zhang et al. [20] propose a non-local TV-L1 model, which designs a linearizing iterative scheme combining median filtering and a coarse-to-fine image warping technique used for calculating large displacements. The above methods always use prior knowledge to define and further optimize an energy function. Although knowledge-driven approaches have achieved promising results and have dominated the benchmark for a long time, these methods only use priors to constrain the relationship between flow and images, and can not exploit knowledge from large amounts of data. Moreover, the optimization process is online and is time-consuming, which is a critical limitation for real-time application.

**B. DATA-DRIVEN APPROACH**

With the rapid development of deep convolutional neural networks (CNNs), several deep learning approaches have been proposed to solve optical flow estimation. FlowNet [6] is the first work for optical flow estimation, which adopts encoder-decoder architecture to extract the dense flow fields from adjacent images. In [6], they propose two novel networks, FlowNetS and FlowNetC. Specifically, FlowNetC conducts a correlation layer for calculating the feature matching. Vaquero et al. [21] introduce coarse and fine components into FlowNet, and cast flow estimation task to a joint classification and regression problem. However, the accuracy of [6], [21] can not match many knowledge-driven approaches. Based on [6], FlowNet2.0 stacks several FlowNet modules and uses image warping among these sub-networks, which refines the flow field after each stage. Although FlowNet2.0 can produce high-quality flow map and achieves the state-of-the-art results on several benchmarks, the model parameters and running time are huge for real-time applications. In addition, because each sub-network needs to be trained in sequential manner, the training process is complex and time-consuming. Many unsupervised approaches [22]–[27] also adopt the encoder-decoder architecture as a backbone to train optical flow on unlabeled data. USCNN [22] first design an unsupervised network for learning optical flow, which introduce brightness constancy loss function for guiding the training process. Further, Yu et al. [23] additionally use spatial smoothness constraint to exploit the spatial relationship of flow fields during training. Meister et al. [28] introduce a novel census loss function into optical flow estimation and stacks several sub-networks for iterative refinements similar to [7]. Zhu and Newsam [26] introduce dilated convolution into unsupervised network. Wang et al. [27] introduce occlusion estimation into learning flow, which first calculates forward and backward flow and uses consistency check function to estimate the occlusion region. The above approaches all adopt the encoder-decoder architecture as the backbone, which always estimate both large and small displacements simultaneously. Therefore, the network needs more feature channels and model parameters, which leads to less efficiency.

Recently, spatial pyramid network is frequently used for learning optical flow. Several works [8], [10], [11], [29] design the light weight network based on the spatial pyramid network. The main advantage of spatial pyramid network is that it can infer the residual flow instead of original large displacement flow at each pyramidal level, which effectively reduces the model size and parameters. Ranjan and Black [8] are the first to design a spatial pyramid network for optical flow estimation, which uses image warping at each pyramidal level to split large displacements into small motions. Thus, they only need to estimate small motion at each pyramidal level. Its network is 96% smaller than that of FlowNet (encoder-decoder) and achieves close performance to FlowNet. Based on [8], PWC-Net [11] and LiteFlowNet [10] focus on improving precision with light weight architecture. PWC-Net uses feature warping instead of warping image directly to reduce feature-space distance and introduces cost volume into each pyramidal level. Further, they design a context network for post-processing. LiteFlowNet also conducts feature warping, and especially introduces a feature-driven local convolution to regularize the flow field. Both PWC-Net [11] and LiteFlowNet [10] achieve high precision with the premise of small model size and low computational cost. However, existing approaches always design feature extractor using the standard convolution, which contains the fixed receptive fields and ignore to aggregate and select multi-scale feature to capture more rich contextual information. Moreover, few works exploit the effectiveness of multi-scale fusion architecture for optical flow estimation.

**C. FEATURE FUSION NETWORK**

Feature fusion is widely used in convolutional neural network (CNNs) for several vision tasks, such as image classification [13], [30], [31], semantic segmentation [32], [33], person re-identification [34], optical flow estimation [6], image super-resolution [35], etc. Many methods [6], [13], [30], [32], [35] combine low-level features and high-level features by using skipping connection. However, the skipping connection just combines low-level and high-level features, and can not extract and fuse multi-scale features. Inception [30] designs a multi-scale fusion network for image classification, which contains multiple branches with different kernel size to capture multi-scale features. The work [34] also uses different receptive fields to obtain multi-scale features, and then uses the Inception architecture [30] to concatenate the multi-scale features. The work [33] propose an atrous spatial pyramid pooling module (ASPP) to fuse multi-scale feature information, which contains several dilated convolutional layers with different rate and a global average pooling layer to aggregate global spatial contexts. Then, the extracted features are concatenated directly for the following convolution. Although these multi-scale networks [30], [31], [33]–[35] can extract
FIGURE 1. The architecture of our proposed network. Our network is mainly composed of five parts, feature pyramid, feature warping, cost volume, cascaded selective flow estimator and context network. Given two frames, the network can output dense flow fields directly. The feature pyramid module contains two convolutional streams. The feature warping module can reduce feature-space distance between the two streams. The cascaded selective flow estimator contains selective kernel module and optical flow estimator.

FIGURE 2. The architecture of selective kernel network. The selective kernel network is mainly composed of three parts, split, fusion and select. Given input feature $F$, the module can adaptively adjust receptive field of kernel. The split part contains standard convolutional and dilated convolutional layers. The fusion part contains global average pooling and fully connect layer. The select part contains softmax layer and standard convolutional layers.

multi-scale features using different kernel size or dilated convolution, these networks fuse the features using direct way and can not adaptively select kernel and multi-scale features. Furthermore, the kernel size used in previous works, such as Inception [30], [31] and Deeplab [33], is fixed and can not be adjusted though a learning mechanism. Most recently, Li et al. [13] propose a novel architecture for image classification, selective kernel network, which adaptively adjust receptive field of convolutional kernel based on multi-scale features. Although the selective kernel mechanism is well used in image classification task, there is no work to explore the effectiveness on optical flow estimation. In this paper, we are the first to introduce selective kernel mechanism into optical flow estimation and try to selective multi-scale features adaptively in optical flow network.

III. APPROACH
In this section, we firstly introduce the selective kernel mechanism. Then, we give the entire network architecture of our proposed approach. Given two adjacent images $I_1$ and $I_2$ and ground truth $W'$, our network can learn and produce optical flow $W'$ directly. The entire network architecture is shown in Fig. 1. First, two frames are fed into feature pyramid module. The feature pyramid contains two streams, which can transform each image to a pyramid of multi-scale high-dimensional features. Then, the multi-scale features are fed into feature warping module, which can reduce feature-space distance between two streams. After calculating cost, the cascaded selective flow estimator can select multi-scale feature adaptively and then estimate dense optical flow at each pyramid level. Finally, we use a context network to refine the flow fields.

A. SELECTIVE KERNEL MECHANISM
The selective kernel mechanism is designed to adaptively select different kernel of multi-scale features, which mainly contains three parts, split, fuse and select. As shown in Fig. 2, given a feature map $F \in \mathbb{R}^{H \times W \times C}$, we can obtain two branches features $F_1$ and $F_2$ by using two
convolutional layer with kernels of 3 × 3 and 5 × 5. Both \(F_1\) and \(F_2\) are followed by a batch norm layer and a ReLU layer. We use a ReLU layer to add the non-line relationship between each convolutional layer. In our experiments, instead of using 5 × 5 kernel size, we use the dilated convolutional layer with rate of 2, which can reduce the computational burden. The split process can be defined as

\[
F_1 = C_{3 \times 3}(F),
\]

\[
F_2 = D_{3 \times 3}(F),
\]

where \(C_{3 \times 3}(\cdot)\) denotes the standard convolutional layer with kernel of 3 × 3 and \(D_{3 \times 3}(\cdot)\) denotes the dilated convolutional layer with 3 × 3 kernel size and dilated size 2.

Then, we fuse two branches via element-wise summation, which can be defined as

\[
F_c = F_1 + F_2,
\]

where \(F_c\) denotes the fused feature. To aggregate global information, we use a global average pooling layer, which can generate channel-wise statistics \(F_c\). The process can be defined as

\[
x_c = g_{x_c} = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} x_c(i, j),
\]

where \(x_c(i, j)\) is the value at point \((i, j)\) of channel \(c\). \(g_{x_c}\) denotes the global average pooling layer. \(x_c\) denotes the channel features of \(F_c\). Then, a fully connect layer is used to learn the relationship between different scale features and to adaptively select kernel. In addition, the fully connect layer is followed by a batch norm layer and a ReLU layer. This process can be defined as

\[
F_f = FC(F_c),
\]

where \(FC(\cdot)\) denotes the fully connect operation. The number of channels is reduced to \(C/16\). Then, two softmax layers are applied to extract the weight of different scales. The soft attention vector can be defined as

\[
\alpha_c = \frac{e^{D_c F_f}}{e^{D_c F_f} + e^{B_c F_f}},
\]

\[
\beta_c = \frac{e^{B_c F_f}}{e^{D_c F_f} + e^{B_c F_f}},
\]

where \(D_c\) and \(B_c\) are the attention vector for \(F_1\) and \(F_2\). \(D_c\) denotes the \(c\)-th row of \(D\) and \(\alpha_c\) is the \(c\)-th element of \(\alpha\). \(B_c\) denotes the \(c\)-th row of \(B\) and \(\beta_c\) is the \(c\)-th element of \(\beta\). \(\alpha_c\) and \(\beta_c\) have the relationship shown as follow.

\[
\alpha_c + \beta_c = 1.
\]

The selected feature \(F_{nc}\) can be denoted as

\[
F_{nc} = \alpha_c \ast F_1 + \beta_c \ast F_2.
\]

Thus, we obtain the final feature \(F_n\), which contains multiple scale information.

**B. NETWORK ARCHITECTURE**

Our network is based on spatial pyramid network, which has been frequently applied in optical flow estimation. The main idea is warping image or feature at each pyramid level, which splits large displacement into small motion. At each pyramid level, the flow estimator only needs to infer residual flow, which greatly reduces the number of parameters. In this work, we combine spatial pyramid network with selective kernel unit, which can extract and select multi-scale features and then fuse the selected features.

Our proposed network is shown in Fig. 1, which mainly contains five parts, feature pyramid, feature warping, cost volume, cascaded selective flow estimator and context network. The input images are fed into feature pyramid from two branches with sharing weights. Each branch of feature pyramid contains twelve convolutional layers followed by ReLU function, and two layers is a stage. The number of feature maps at each stage is 196, 128, 96, 64, 32 and 16 respectively. Each stage contains two convolutional layers with the same channel number. The kernel sizes of all of these layers are set to 3 × 3.

At each pyramid level \(i\), a flow field \(W_i\) of the down-sampled images \(I_1\) and \(I_2\) is inferred from flow estimator. We introduce feature warping to reduce the feature distance except the top layer, which uses the up-sampled flow to warp the second feature \(f_i^{2} \) to the first feature \(f_i^{1}\). Note that the up-sampled flow in Fig. 1 just appears from the second pyramidal level, and the source of the sampled optical flow is the flow estimated at previous level. This process can be defined as

\[
f_i^w(x) = f_i^2(x + W_i),
\]

where \(f_i^w\) is the warped feature, and \(x\) is the pixel index. This allows our network to only calculate small displacement flow. To enable the end-to-end training, we use bilinear interpolation to conduct the warping process. The interpolation process can be defined as

\[
f_i^w(x) = \sum_{x_c} f(x_c) (1 - |x_w - x_c^i|)(1 - |y_w - y_c^i|),
\]

where \(x_w = x + W_i = (x_w, y_w)^T\) denotes the source coordinates in the input feature map \(f\). \(x = (x, y)^T\) denotes the target coordinates of the interpolated feature map \(f^w\).

Then, the warped feature \(f_i^w\) and the first image feature \(f_i^1\) are fed into cost volume layer, which can calculate the matching cost between \(f_i^w\) and \(f_i^1\). The cost volume layer is similar to the correlation layer proposed in [6], which can be defined as

\[
C = \frac{1}{N} f_i^1(x_1) ^T f_i^w(x_2),
\]

where \(f_i^1\) is the point position at \(F_1\) and \(x_2\) is the point position at \(f_i^w\). \(T\) is transpose operator.

As shown in Fig. 1, the matching cost and the upsamled flow are fed into flow estimator. In our network, the flow
estimator contains six levels corresponding to the feature pyramid part. At each pyramidal level of flow estimator, fully convolutional architecture followed by ReLU function are used to extract high-level features and further yield flow fields. The number of feature channels at each convolutional layers are 128, 128, 96, 64, and 32. In contrast to [11], we additionally introduce selective kernel network into flow estimator part. We fuse the selective kernel mechanism at the top three levels. After each convolution, the extracted features are fed into two convolutional layers with different scale features. Then, the high-level feature can be enhanced by introducing multi-scale contexts and embedding more spatial information into high-level features. Furthermore, the enhanced features are beneficial for producing more refined flow fields. We further adopt the dense connection [36] to further enhance the representation power of network.

In many knowledge-driven approaches, post-processing is an essential part for refining flow fields. A context network is used in our network, which is composed of successive standard and dilated convolutional layer with different rate. The number of channels are set to 128, 128, 96, 64, 32, respectively. The receptive field is enlarged for capturing more contextual information. The number of dilated convolution is 4, and the rates are set to 2, 4, 8, 16. The kernel size of each convolution is set to 3 × 3.

Loss function is an essential part for data-driven approaches. From Fig. 1, we can find that the ground truth \( W \) and final flow \( W \) are fed into loss function. For supervised networks, the end point error (EPE) is always adopted to guide the training process. In our network, we also use the EPE as loss function. The training loss can be defined as

\[
L_{\text{epe}} = \sum_{i=1}^{W} \sum_{j=1}^{H} \sqrt{(u_{i,j} - u'_{i,j})^2 + (v_{i,j} - v'_{i,j})^2},
\]

(13)

where \( u_{i,j} \) and \( v_{i,j} \) are the estimated flow with two directions, horizontal and vertical. \( u'_{i,j} \) and \( v'_{i,j} \) are the ground truth flow.

### IV. EXPERIMENTS

In this section, We first review existing datasets for flow training and evaluation. Then, we describe the training details for our method. Further, we mainly report the experimental results on several public benchmarks, such as MPI-Sintel [37], KITTI2012 [38] and KITTI2015 [39], and compare our network to recent knowledge-driven and data-driven approaches.

### A. DATASETS

Our network was first trained on FlyingChairs [6] dataset and then fine-tuned on FlyingThings3D [40] dataset. The FlyingChairs and FlyingThings3D datasets are synthetic. We further tested our network on MPI-Sintel, KITTI2012, KITTI2015 and Middlebury datasets. Among them, MPI-Sintel is a synthetic dataset. KITTI2012 and KITTI2015 are real-world datasets. The details of these datasets are listed on Table 1.

1) **FLYINGCHAIRS**

is a large synthetic dataset, which is generated by applying affine transformations to real images and synthetically rendered chairs. It totally contains 22232 image pairs for training and 640 image pairs for test. The resolution of each image is 448×320. In addition, it contains dense ground truth for training and test.

2) **FLYINGTHINGS3D**

is also a large synthetic dataset for flow and disparity training and test. In our work, we use this dataset for optical flow task. It contains 25000 stereo frames with dense ground truth data and contains 2247 different scenes. The resolution of each image is 768×384.

3) **MPI-SINTEL**

is a synthetic dataset for training and test of optical flow, which contains 1064 frames for training and 564 frames
for test. This dataset contains two versions, Clean and Final. The Clean version contains realistic illuminations and reflections. The Final version additionally adds rendering effects like motion, defocus blurs and atmospheric effects.

4) KITTI2012 is a real-world dataset with autonomous driving scene, which consists of 194 image pairs for training with sparse ground truth and 195 image pairs for test.

5) KITTI2015 is also a real-world dataset with autonomous driving scene, which contains 200 image pairs with sparse ground truth for training and 200 image pairs for test.

6) MIDDLEBURY is a small dataset for optical flow estimation, which contains 8 scenes for training and 12 scenes for test.

B. TRAINING DETAILS
Our experiments were conducted on a NVIDIA 1080Ti GPU using the Caffe framework [41]. We trained our network using Adam optimizer [42] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We first trained our network on FlyingChairs dataset with batch size of 8. We randomly cropped 448 x 320 patches from original images. The initial learning rate was set to 1e-4 and then it decays half every 200k iterations after the first 400k iterations. And then we fine-tuned our model on FlyingThings3D dataset with batch size of 4. We randomly cropped 768 x 384 patches from original images. The initial learning rate was set to 1e-5 and then divided it by 2 after the first 200k iterations. To avoid over-fitting, we used data augmentation scheme same as that in [6]. We further fine-tuned our network on MPI-Sintel dataset with 150k iterations. The batch size was set to 4, and the cropping size was set to 768 x 384. We first set the learning rate to 3e-5, and then divided it by 2 after the first 50k iterations. Further, we set the learning rate to 2e-5 and trained our network with 150k iterations. The learning rate was divided by 2 after the first 50k iterations.

C. COMPARISON TO STATE-OF-THE-ART
We compared our model with the recent state-of-the-art methods on MPI-Sintel, KITTI and Middlebury in Table 2, Table 3 and Table 4. These approaches can be roughly divided into four categories, knowledge-driven approaches, unsupervised approaches, heavy weight networks (supervised approaches) and light weight networks (supervised approaches). We further mark the best results in each category. Note that we mainly compare our network with the supervised approaches for fair comparison. Note that ‘C’ denotes the model only trained on FlyingChairs, and ‘C+T’ denotes the model trained on FlyingChairs and FlyingThings. ‘C+S+T’ denotes the model trained on FlyingChairs, FlyingThings and MPI-Sintel.

### TABLE 2. Performance comparison on MPI-Sintel dataset. Average EPE denotes average endpoint error. We divided these methods into four categories for fair comparison. The best number for each category is highlighted in bold.

| Method               | MPI-Sintel Clean | MPI-Sintel Final |
|----------------------|------------------|------------------|
|                      | Train | Test | Train | Test |
| HS [1]               | 7.23  | 8.73 | 8.38  | 9.61 |
| Classic-NL [2]       | 6.03  | 7.96 | 7.99  | 9.15 |
| EpicFlow [18]        | 2.27  | 4.12 | 3.56  | 6.29 |
| PCA-Layers [5]       | 3.22  | 5.73 | 4.52  | 7.89 |
| LDOP [4]             | 4.64  | 7.56 | 5.96  | 9.12 |
| DSTFlow [24]         | 6.93  | 10.40| 7.62  | 11.11|
| USCNN [22]           | -     | -    | 8.88  | -    |
| UnFlow-CSS [28]      | -     | 9.37 | 7.91  | 10.22|
| Densef [25]          | -     | -    | -     | 10.07|
| OccAwareFlow [27]    | 5.23  | 8.02 | 6.34  | 9.08 |
| FlowNetS [6]         | 4.50  | 7.42 | 5.45  | 8.43 |
| FlowNetC [6]         | 4.51  | 7.28 | 5.87  | 8.81 |
| CaF-Full+41c [21]    | 6.51  | 9.42 | 7.28  | 10.18|
| FlowNet2-S (C+T) [7] | 3.79  | -    | 4.99  | -    |
| FlowNet2-C (C+T) [7] | 3.04  | -    | 4.60  | -    |
| FlowNet2-SS (C+T) [7]| 2.10  | -    | 3.23  | -    |
| FlowNet2.0 (C+T+S) [7]| 2.02 | 3.96 | 3.54  | 6.02 |
| SpyNet (C) [8]       | 4.12  | 6.97 | 5.37  | 8.43 |
| Motion-Cue (C) [29]  | 4.08  | -    | 5.26  | -    |
| RecSpyNet (C) [12]   | -     | 6.79 | 9.38  | -    |
| LiteFlowNet (C+T) [10]| 2.48 | -    | 4.04  | -    |
| PWC-Net (C) [11]     | 3.33  | -    | 4.59  | -    |
| PWC-Net (C+T) [11]   | 2.55  | -    | 3.93  | -    |
| Ours (C) [11]        | 2.97  | -    | 4.24  | -    |
| Ours (C+T) [11]      | 2.34  | 4.79 | 3.33  | 6.92 |

### TABLE 3. Performance of the fine-tuned models on MPI-Sintel training set. Average EPE denotes average endpoint error. We divided these approaches into two categories for fair comparison. The best number for each category is highlighted in bold.

| Method               | MPI-Sintel Clean | MPI-Sintel Final |
|----------------------|------------------|------------------|
|                      | Average EPE | Average EPE |
| DSTFlow-ft [24]      | 6.16      | 6.81      |
| Occ-Aware-ft [27]    | 4.03      | 5.95      |
| FlowNetS-ft [6]      | 3.66      | 4.44      |
| FlowNetC-ft [6]      | 3.78      | 5.28      |
| SpyNet-ft [8]        | 3.17      | 4.32      |
| RecSpyNet-ft [12]    | -         | 4.27      |
| PWC-Net-ft [11]      | 1.70      | 2.21      |
| Ours-ft              | 1.66      | 2.17      |

1) MPI-SINTEL
Table 2 reports the comparison of most existing approaches on MPI-Sintel dataset (Clean and Final version). We used average end point error (average EPE) as the error measure standard. As we can see, among these knowledge-driven methods, EpicFlow [18] and PCA-Layers [5] achieve relatively lower average EPE than other approaches HS [1], Classic-NL [2], LDOF [4]. Compared to our results, EpicFlow [18] obtains better results on Sintel Clean (training and test) and Sintel Final test sets. We also find that “Ours (C)” performs better than HS [1], Classic-NL [2], PCA-Layers [5] and LDOF [4] on Sintel Clean training and Sintel Final training sets. Moreover, “Ours (C+T)” achieves better performance than HS [1], Classic-NL [2], PCA-Layers [5] and LDOF [4] on all cases of Sintel...
TABLE 4. Performance comparison on KITTI2012 and 2015 datasets. Average EPE denotes average endpoint error. Fl-all denotes the ratio of pixels where flow estimate is wrong by both $\geq 3$ pixels and $\geq 5\%$.

| Method            | KITTI2012 | KITTI2015  |
|-------------------|-----------|------------|
|                   | Average EPE | Average EPE | Fl-all |
|                   | Train    | Test      | Train    | Test      |
| HS [1]            | 11.7     | -         | 41.80%   | -         |
| Classic-NL [2]    | 16.4     | -         |         |           |
| EpicFlow [18]     | 3.09     | 3.8       | 9.27     | 27.10%    |
| PCA-Layers [5]    | 5.99     | 5.2       | 12.74    |           |
| LDODF [4]         | 10.94    | 12.4      | 18.19    |           |
| BackToBasic [23]  | 11.30    | 9.9       |         |           |
| DSTFlow [24]      | 16.98    | 24.3      | 24.3     | 11.6      |
| UnFlow-CSS [28]   | 3.29     | -         | 8.10     |           |
| DenseF [25]       | -        |           | 11.6     |           |
| OccAwareFlow [27] | 12.95    | -         | 21.30    |           |
| FlowNetS (C) [6]  | 8.26     | 9.1       | -        | 51.00%    |
| FlowNetC (C) [6]  | 9.35     | -         | -        | -         |
| FlowNet2-S (C+T) [7] | 7.26     | -         | 14.28    | -         |
| FlowNet2-C (C+T) [7] | 5.79     | -         | 11.49    | -         |
| FlowNet2-CSS (C+T) [7] | 3.55     | -         | 8.94     | -         |
| FlowNet2.0 (C+T+S) [7] | 4.01     | -         | 10.08    | -         |
| SpyNet (C) [8]    | 9.12     | -         | -        | -         |
| Motion-Cue (C) [29] | 7.92     | -         | -        | -         |
| RecSpyNet (C) [12] | 10.02    | 13.7      | -        | 40.90%    |
| LiteFlowNet (C+T) [10] | 4.0      | -         | 10.39    | -         |
| PWC-Net (C) [11]  | 5.14     | -         | 13.20    |           |
| PWC-Net (C+T) [11] | 4.14     | -         | 10.35    |           |
| Ours (C)          | 4.51     | -         | 13.10    |           |
| Ours (C+T)        | 3.66     | 4.73      | 9.98     | 32.54%    |

2) KITTI

We evaluated our model on KITTI2012 and KITTI2015 (training and test) sets. From Table 4, for knowledge-driven approaches, we can find that EpicFlow [18] achieves the lowest error on KITTI 2012 and 2015 datasets. EpicFlow [18] introduces an edge-aware interpolation approach, which can preserve motion boundary and can produce sharp and accurate flow field. Moreover, energy minimization is used for post-processing. Nevertheless, the running time of EpicFlow [18] is increased due to interpolation and post-processing. Among unsupervised methods, UnFlow-CSS [28] outperforms other methods by a large margin. However, UnFlow-CSS has a large model size and slow runtime for real-time application due to the stacking operation. For heavy supervised networks, FlowNet2-CSS [7] achieves better performance than other networks on KITTI2012 and 2015 sets. The result of “Ours (C+T)” is close to FlowNet2-CSS and outperforms FlowNetS [6], FlowNetC [6], FlowNet2-S [7] and FlowNet2-C [7] on KITTI2012 and 2015 sets by a large margin. Compared to light weight networks SpyNet [8], Motion-Cue [29], RecSpyNet [12] and LiteFlowNet [10], “Ours (C)” and “Ours (C+T)” outperform [8], [12], [29] by a large margin. Specially, compared to PWC-Net (C) [11], “Ours (C)” reduces the average EPE about 0.48 and 0.37 on KITTI2012 training and KITTI2015 training sets respectively.

3) MIDDLEBURY

In Table 5, we evaluated our model on Middlebury training datasets. We can find that “Ours (C+T)” outperforms FlowNetS [6], FlowNetC [6], PWC-Net (C) [11],
FIGURE 3. Examples of flow fields from different methods on training set of MPI-Sintel (Final version). In each row left to right: Image1, image2, ground truth flow and 3 predictions: FlowNetC [6], PWC-Net (C-T) [11] and “Ours (C-T)”.  

TABLE 5. Performance comparison on Middlebury training set. Average EPE denotes average endpoint error. The best number for each category is highlighted in bold.

| Method                  | Middlebury Average EPE |
|-------------------------|-------------------------|
| FlowNetS [6]            | 1.09                    |
| FlowNetC [6]            | 1.15                    |
| LiteFlowNet (C+T) [10]  | 0.39                    |
| PWC-Net (C) [11]        | 0.48                    |
| PWC-Net (C+T) [11]      | 0.41                    |
| Ours (C)                | 0.45                    |
| Ours (C+T)              | 0.37                    |

TABLE 6. Model size and running time comparison. The running time is tested on KITTI2012 training set with same GPU. Note that the model size is the memory occupation on computer, and the running time is average time on all frames.

| Method                  | Model size (MB) | Running time (ms) |
|-------------------------|-----------------|-------------------|
| FlowNetS [6]            | 154.5           | 135               |
| FlowNetC [6]            | 156.4           | 174               |
| UnFlow-CSS [28]         | 473.6           | 429               |
| FlowNet2.0 [7]          | 638.5           | 659               |
| PWC-Net [11]            | 41.1            | 103               |
| Ours                    | 44.2            | 119               |

PWC-Net (C+T) [11] and LiteFlowNet (C+T) [10]. The result of LiteFlowNet (C+T) [10] is close to “Ours (C+T)”.  

D. MODEL SIZE AND RUNNING TIME

We measure runtime of each network on a computer equipped with an NVIDIA 1080Ti GPU. The running time is tested on KITTI2012 training set. Table 6 reports the running time and model size of each network. Note that the running time is the average time of the entire KITTI2012 training sequence. We can see that the model sizes of FlowNetC [6] and FlowNetC [6] are about three times larger than our model. Moreover, the running time is slower than our model. We also find that the model size of UnFlow-CSS [28] is about 11 times bigger than ours, and the running time of UnFlow-CSS [28] is about 4 times slower than ours. Specially, the model size of FlowNet2.0 [7] is about 13 times larger than our model and the running time of FlowNet2.0 [7] is about 6 times slower than our model. As a result of the introduction of selective kernel module, the model size and running time of our model are slightly increased compared to the original PWC-Net.
E. ABLATION STUDY

In order to study the importance of selective kernel module in our method, we trained and evaluated a series of models. Table 7 shows the ablation study on MPI-Sintel training set (Final version) and KITTI2012 training set. The first two rows show the two models only trained on FlyingChairs. We can find that the average EPE would be increased about 0.35 and 0.63 on MPI-Sintel and KITTI2012 datasets if without the selective kernel module. The last two rows show the two models trained on FlyingChairs and FlyingThings. We can find that the average EPE would be increased about 0.6 and 0.48 on MPI-Sintel and KITTI2012 datasets if without the selective kernel module. The above experiments verify that the selective kernel module is significant and useful for optical flow estimation.

V. CONCLUSION

In this paper, we introduce selective kernel mechanism into optical flow estimation. Based on selective kernel module, we present a novel selective kernel pyramid network, which not only can extract multi-scale feature at each pyramidal level but also can adaptively aggregate multi-scale information. The selective kernel module can adjust the receptive field and can learn the joint feature of both local feature and the surrounding context. Experiments conducted on MPI-Sintel and KITTI datasets validate our proposal and analysis. Moreover, our network achieves comparable performance to several recent state-of-the-art networks on public datasets. In future, we will extend our network to perform optical flow estimation in spatio-temporal domain.

REFERENCES

[1] B. K. P. Horn and B. G. Schunck, “Determining optical flow,” Artif. Intell., vol. 17, nos. 1–3, pp. 185–203, Aug. 1981.
[2] D. Sun, S. Roth, and M. J. Black, “Secrets of optical flow estimation and their principles,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2010, pp. 2432–2439.
[3] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, “High accuracy optical flow estimation based on a theory for warping,” in Proc. Eur. Conf. Comput. Vis. (ECCV), May 2004, pp. 25–36.
[4] T. Brox and J. Malik, “Large displacement optical flow: Descriptor matching in variational motion estimation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 33, no. 3, pp. 500–513, Mar. 2011.
[5] J. Wulff and M. J. Black, “Efficient sparse-to-dense optical flow estimation using a learned basis and layers,” in Proc. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 120–130.
[6] A. Dosovitskiy, P. Fischery, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, “FlowNet: Learning optical flow with convolutional networks,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 2758–2766.
[7] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, “FlowNet 2.0: Evolution of optical flow estimation with deep networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 1647–1655.
IEEE Access

MINGLIANG ZHAI was born in Xining, China, in 1994. He received the B.Eng. degree from Jilin University, China, in 2016. He is currently pursuing the Ph.D. degree with Harbin Engineering University. His research interests include image processing, computer vision, and pattern recognition.

XUEZHI XIANG was born in Harbin, China, in 1979. He received the B.Eng. degree in information engineering, and the M.Sc. and Ph.D. degrees in signal and information processing from Harbin Engineering University, China, in 2002, 2004, and 2008, respectively. He was a Postdoctoral Fellow with the Harbin Institute of Technology, from 2009 to 2011. From 2011 to 2012, he was a Visiting Scholar with the University of Ottawa. Since 2010, he has been an Associate Professor with the School of Information and Communication Engineering, Harbin Engineering University. He has authored over 40 articles. His research interests include image processing, computer vision, and pattern recognition. Dr. Xiang is also a member of the Association for Computing Machinery and a Senior Member of the China Computer Federation.
NING LV was born in Yingkou, China, in 1994. She received the B. Eng. degree in communication engineering from Shandong University, China, in 2016. She is currently pursuing the Ph.D. degree with Harbin Engineering University. Her research interests include computer vision and pattern recognition.

SYED MASROOR ALI was born in Karachi, Pakistan, in 1979. He received the B.S.Eng. degree from Sir Syed University, Karachi, in 2004, and the M.S. degree in mobile computing and information systems from Hamdard University, Karachi, in 2009. He is currently pursuing the Ph.D. degree with the School of Information and Communication Engineering, Harbin Engineering University, Harbin, China. His research interest includes computer vision, flow estimation and visualization, and image processing.

ABDULMOTALEB EL SADDIK (F’09) is currently the Distinguished University Professor and the University Research Chair with the School of Electrical Engineering and Computer Science, University of Ottawa. He is also a Senior Associate Editor among others of the ACM Transactions on Multimedia Computing, Communications, and Applications (ACM TOMCCAP currently TOMM), the IEEE Transactions on Multimedia (IEEE TMM), and the Guest Editor for several IEEE Transactions and journals. He has authored or coauthored four books and more than 550 publications and chaired more than 50 conferences and workshop. His research interests include multimodal interactions with sensory information in smart cities. He has received research grants and contracts totaling more than $18 M. He has supervised more than 120 researchers and received several international awards, among others, are the ACM Distinguished Scientist, a Fellow of the Engineering Institute of Canada, the Canadian Academy of Engineers, the IEEE I&M Technical Achievement Award and the IEEE Canada Computer Medal.