RAPIDFlow: Recurrent Adaptable Pyramids with Iterative Decoding for Efficient Optical Flow Estimation

Henrique Morimitsu†, Xiaobin Zhu†, Roberto M. Cesar-Jr.‡, Xiangyang Ji*, and Xu-Cheng Yin†

***Abstract*— Extracting motion information from videos with optical flow estimation is vital in multiple practical robot applications. Current optical flow approaches show remarkable accuracy, but top-performing methods have high computational costs and are unsuitable for embedded devices. Although some previous works have focused on developing low-cost optical flow strategies, their estimation quality has a noticeable gap with more robust methods. In this paper, we develop a novel method to efficiently estimate high-quality optical flow in embedded devices. Our proposed RAPIDFlow model combines efficient NeXt1D convolution blocks with a fully recurrent structure based on feature pyramids to decrease computational costs without significantly impacting estimation accuracy. The adaptable recurrent encoder produces multi-scale features with a single shared block, which allows us to adjust the pyramid length at inference time and make it more robust to changes in input size. Also, it enables our model to offer multiple tradeoffs between accuracy and speed to suit different applications. Experiments using a Jetson Orin NX embedded system on the MPI-Sintel and KITTI public benchmarks show that RAPIDFlow outperforms previous approaches by significant margins at faster speeds. Our code is available at `https://github.com/hmorimitsu/ptlflow/tree/main/ptlflow/models/rapidflow`.**

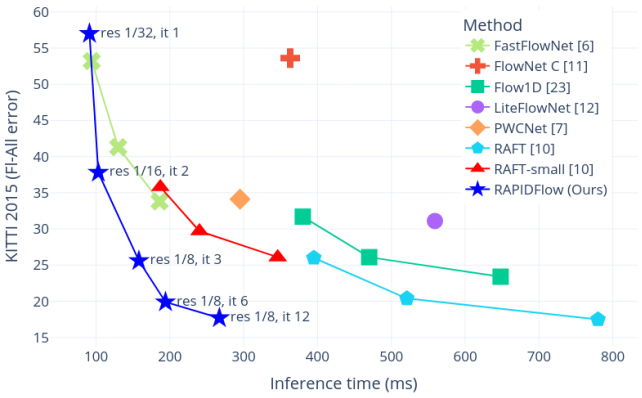

Fig. 1. Generalization results on KITTI 2015 train set vs. inference time. RAPIDFlow shows significantly lower estimation errors in less time. In particular, we outperform FastFlowNet [6] at similar speeds and compete with RAFT [10] while being almost three times faster. The speed is measured on a Jetson Orin embedded system. "res" denotes the maximum feature resolution, and "it" is the number of refinement iterations. FastFlowNet is tested at res $\in \{1/16, 1/8, 1/4\}$. Other iterative methods, such as RAFT, use res 1/8 and it $\in \{3, 6, 12\}$. All methods were tested with FP16 precision.

I. INTRODUCTION

Optical flow estimation aims to compute the per-pixel 2D motion between two consecutive video frames. It is a fundamental task that benefits various applications, such as action recognition [1], autonomous navigation [2], object discovery [3], segmentation [4], and 3D reconstruction [5].

Optical flow methods have greatly benefited from the rise of deep learning, and current approaches achieve remarkable performance improvements. Nonetheless, optical flow estimation is a dense prediction task and usually requires processing high-resolution inputs to produce accurate results. More reliable optical flow methods usually have a very high computational cost, which makes them unsuitable for deploying in embedded devices such as mobile robots. While some previous works have focused on producing more efficient optical flow methods for applications with restricted resources, they usually significantly impact the estimation quality. For example, FastFlowNet [6] significantly decreased the computational cost of PWCNet [7] to run on embedded devices. PatchFlow [8] proposed a two-stage approach where large inputs are divided into smaller patches to reduce the processing costs, making it more efficient than FastFlowNet, but at the cost of decreased accuracy. DIFT [9] explored ways of optimizing the widely adopted RAFT [10] method to mobile phones. Although these methods showed remarkable processing and speed improvements, their accuracy is not on par with more robust methods.

This paper proposes a new, efficient optical flow estimation method suitable for embedded devices. We design an iterative and fully recurrent encoder-decoder structure that allows us to offer multiple tradeoffs between speed and accuracy by dynamically changing the feature sizes and number of refinement iterations. Our recurrent encoder can also easily handle inputs of much larger sizes by changing the number of pyramid levels during inference, which makes it significantly more robust to input size changes than other competing methods. We combine the classical coarse-to-fine approach based on feature pyramids [7] with iterative refinement decoding [10] to produce accurate results with

† School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, China. {hmori,zhuxiaobin,xuchengyin}@ustb.edu.cn

‡ Institute of Mathematics and Statistics, University of São Paulo, São Paulo, Brazil. rmcesar@usp.br

* Department of Automation, Tsinghua University, Beijing, China. xyji@tsinghua.edu.cn

Corresponding author: Xiaobin Zhu

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). RMCJ is grateful to FAPESP (grants #2015/22308-2, #2022/15304-4), CNPq, CAPES, FINEP, and MCTI PPI-SOFTEX (TIC 13 DOU 01245.010222/2022-44).

© 2024 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

H. Morimitsu, X. Zhu, R. M. Cesar-Jr, X. Ji, and X.-C. Yin, "RAPIDFlow: Recurrent Adaptable Pyramids with Iterative Decoding for efficient optical flow estimation," in ICRA, 2024.

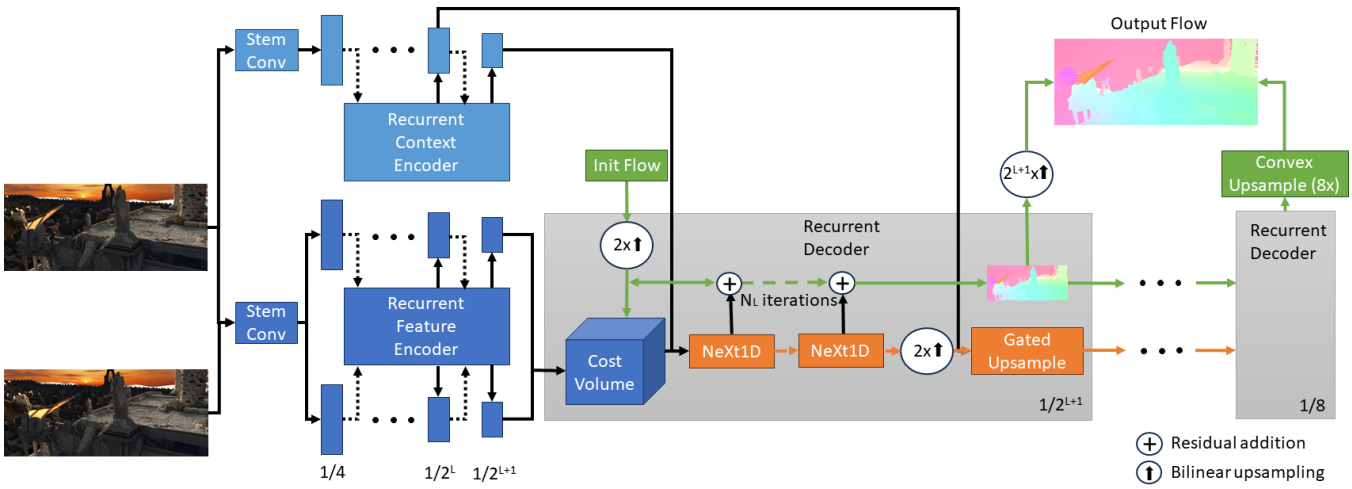

Fig. 2. An overview of RAPIDFlow's structure. A stem convolution block first projects the input images into the feature space. Our recurrent encoder then produces multi-scale feature pyramids by sharing the same encoder block across all scales. The decoder processes the feature pyramids in a coarse-to-fine approach to iteratively refine the flow estimation. The Gated Upsample module allows us to integrate more fine-grained contextual information and transfer the recurrent state across different pyramid levels.

reduced computational cost. We call the proposed method RAPIDFlow (Recurrent Adaptable Pyramids with Iterative Decoding) and deploy it on an NVIDIA Jetson Orin NX 16GB embedded system. All reported performance results throughout this paper are collected by running experiments on this embedded device. The results in Fig. 1 show that our method can be configured to be as fast as FastFlowNet [6] or competitive to RAFT [10].

In summary, our contributions are as follows:

1) We propose a new fully recurrent encoder-decoder structure that can be dynamically changed to produce state-of-the-art results at various tradeoffs between accuracy and inference speed.
2) We propose a novel recurrent feature encoder that uses a single shared block with efficient 1D layers (NeXt1D) to generate feature pyramids of variable levels, which makes our model more robust to changes in input size.
3) We propose an efficient recurrent decoder with NeXt1D layers that refines variable feature pyramids iteratively and produces precise optical flow while significantly reducing parameters and memory cost.

II. RELATED WORKS

A. Optical flow

FlowNet [11] was one of the first deep optical flow models. It adopted a simple encoder-decoder structure with multi-level flow prediction to improve robustness and signal propagation. The introduction of the correlation volume [7], [12] in optical flow models brought a significant improvement in the quality and efficiency of flow models. To reduce the cost of dense matching, HD^3 [13] proposed a hierarchical multi-scale approach to efficiently cover a wide matching search area. RAFT [10] proposed to change the feature pyramid design and use a single high-resolution feature map instead. This approach, combined with a global correlation volume to improve the matching in the high-resolution space, set a new standard for optical flow performance. Since then, multiple methods have adopted and improved this design. GMA [14] and SeparableFlow [15] improved the prediction for challenging regions by aggregating motion information. OCTC [16] added occlusion and transformation consistency into the training to improve the model's overall robustness. More recently, deep network methods started to add attention layers [17], [18], [19], [20], [21], [22] to include more contextual information into the model. Although these methods show remarkable estimation accuracy, they also impose outstanding computation requirements, which makes them unsuitable for applications with restricted resources.

B. Efficient models

Some recent works have focused on improving the efficiency of optical flow models by reducing their memory and processing requirements. FastFlowNet [6] decreased the encoder complexity by using simple pooling operations to create feature pyramids. Combining this encoder with a sparse correlation layer made it competitive to PWCNet [7] while running faster on embedded devices. However, their best results are still not on par with more current methods, such as RAFT [10]. Flow1D [23] vastly decreased RAFT's memory consumption by using only 1D correlation layers with attention, but this caused a noticeable decrease in estimation quality. SCM [24] and DIP [25] also tackled the memory problem with sparse and PatchMatch [26] strategies to decrease the search space. Nonetheless, they have high processing costs and impact the running speed significantly. DIFT [9] had explored optimization strategies to adapt RAFT for mobile phones. Although they showed remarkable speed improvements, the estimation quality was significantly impacted by the reduction of the model.

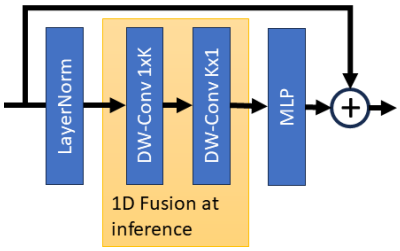

Fig. 3. Overview of the NeXt1D layer. We decrease learnable parameters by using only 1D depthwise convolution layers and fuse them during inference time to increase the model speed.

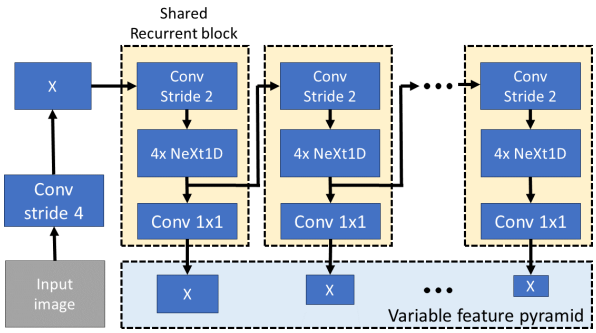

Fig. 4. The proposed recurrent encoder network. The encoder comprises a single layer that can be iteratively unfolded to create a feature pyramid.

III. METHOD

A. Problem formulation

Optical flow estimation can be seen as a matching problem. It aims to find a 2D correspondence (flow) map $F \in \mathcal{R}^{H \times W \times 2}$ between two images I_1 and I_2, where H and W respectively represent the height and width of the image. The introduced RAPIDFlow model architecture is based on RAFT [10], but we propose several improvements to decrease its computational cost without significantly impacting accuracy. We first replace the ResNet [27] encoder network with our single recurrent block that is repeatedly applied to generate multi-scale features. The main component of our encoder network is a module based on fused NeXt1D convolution layers to extract relevant features with low processing costs. We generate deep multi-scale feature pyramids to decrease feature sizes and improve the efficiency of the decoding step. We also leverage NeXt1D layers to fuse context features from previous pyramid levels. By varying the number of pyramid scales and refinement iterations, our method can offer multiple tradeoffs between accuracy and speed to suit various situations with a single model. The main components of RAPIDFlow are illustrated in Fig. 2.

B. Fused 1D convolution block (NeXt1D)

The Xception network [28] showed that decomposing the convolution operation into spatial (depthwise) and channel layers could significantly reduce the model size while retaining performance. Since then, this strategy has been further developed [29], [30], [31] to create even more efficient models. Although this decomposition decreases parameters significantly, the depthwise convolution kernel parameters still grow quadratically to the kernel size. Previous methods have adopted the 1D decomposition [10], [23], [32] to split the 2D kernel into 1D horizontal and vertical ones and decrease the parameters even more. However, although the 1D decomposition decreases parameters, it may also impact the model's speed due to the need to perform two operations in sequence. To maintain the advantages of reduced parameters of the 1D decomposition and increase parallelism, we fuse the 1D decomposition. Since depthwise convolution only operates at a single channel, we can fuse the vertical and horizontal 1D kernels w_v and w_h as follows:

$$Y = (X * w_v) * w_h = X * (w_v \otimes w_h), \qquad (1)$$

where $*$ denotes convolution and $\otimes$ the Kronecker product. Although the resulting fused kernel has K^2 elements per channel, the number of learnable parameters is only $2K$.

We build on the successful ConvNeXt [30] block design and combine fused 1D depthwise 1×7 kernels to further reduce its size with a Multi-Layer Perceptron (MLP) to encode channel relations. We call the resulting block NeXt1D. Our experiments show that using the fused 1D kernel increases our model speed by more than 25% compared to employing 1D decomposition directly. A diagram of the NeXt1D block is shown in Fig 3.

C. Recurrent encoder

Most optical flow models used a fixed CNN network as the feature encoder. This strategy has two main limitations: (1) the number of model parameters increases according to the network depth, and (2) the number of levels l for the feature pyramid cannot be changed after training. The second limitation is especially relevant since it also causes the pyramid feature spatial size to increase according to the input. This significantly increases the memory requirement and hinders generalization since the feature-matching search space changes according to the input size.

To overcome those challenges, we propose to use an encoder composed of a single recurrent block. Being recurrent, the number of pyramid levels L can be changed on the fly to suit the current input better. For our experiments, we add one more feature pyramid level each time the input size doubles in relation to the training size. Our encoder first adopts a stem convolution layer with stride 4 to reduce the input size and projects it into the latent feature space. Then, our recurrent block is repeatedly used to generate multi-scale features for building the pyramid. Fig 4 shows an overview of the operation of our recurrent encoder. It is worth emphasizing that the proposed encoder acts recurrently across spatial scales (multi-scale), distinct from the more traditional time recurrence (multi-frame).

D. Pyramid decoder

The decoder design is similar to RAFT [10], but with essential improvements that significantly decrease the decoding costs. First, inspired by the findings of SKFlow [36], we replace the ConvGRU [37] with our more efficient NeXt1D

TABLE I

RESULTS ON SINTEL AND KITTI DATASETS GROUPED BY INFERENCE TIMES. TRAIN RESULTS ARE COLLECTED FROM MODELS WITHOUT FINE-TUNING, WHILE TEST RESULTS COME FROM THE OFFICIAL BENCHMARKS. FLOPS AND TIME RESULTS ARE BASED ON 1280×384 INPUTS MEASURED USING FP16 PRECISION ON A JETSON ORIN NX EMBEDDED SYSTEM. "RES." DENOTES THE MAXIMUM FEATURE RESOLUTION, AND "ITERS." IS THE NUMBER OF REFINEMENT ITERATIONS. SINTEL VALUES CORRESPOND TO EPE, WHILE KITTI 2015 TO FL-ALL. THE FIRST- AND SECOND-BEST RESULTS ARE HIGHLIGHTED IN BOLD AND UNDERLINED.

Method			Time (ms)	FLOPs	Params	Train set			Test set		
Name	Res.	Iters.				S.Clean	S.Final	K15	S.Clean	S.Final	K15
RAPIDFlow	1/32	1	**91**	12G	1.6M	**4.09**	**5.19**	57.0	-	-	-
FastFlowNet [6]	1/16	3	94	**8G**	**1.3M**	4.13	5.31	**53.2**	-	-	-
RAPIDFlow	1/16	2	**103**	18G	1.6M	**2.76**	**3.96**	**37.8**	-	-	-
FastFlowNet	1/8	4	130	**12G**	**1.3M**	3.36	4.61	41.3	-	-	-
RAPIDFlow	1/8	3	**158**	54G	1.6M	**2.03**	**3.36**	**25.5**	-	-	-
FastFlowNet	1/4	5	186	**27G**	1.3M	2.89	4.14	33.1	4.89	6.08	11.22
RAFT-small [10]	1/8	3	187	65G	**0.9M**	3.45	4.93	35.8	-	-	-
RAPIDFlow	1/8	6	**194**	**79G**	1.6M	1.69	3.00	19.9	-	-	-
RAPIDFlow	1/8	12	267	128G	1.6M	1.58	2.91	17.7	2.03	3.56	6.12
PWCNet [7]	1/4	5	295	180G	9.3M	2.55	3.93	33.6	3.45	4.59	7.72
PWCNet-IRR [33]	1/4	5	321	194G	3.3M	2.73	4.05	35.8	-	-	-
RAFT-small	1/8	12	346	185G	**0.9M**	2.19	3.54	26.1	-	-	-
FlowNet C [11]	1/4	5	363	94G	39.1M	4.31	5.87	-	6.85	8.51	-
Flow1D [23]	1/8	3	380	340G	5.7M	2.46	4.01	31.7	-	-	-
RAFT [10]	1/8	3	395	374G	5.2M	2.13	3.62	26.0	-	-	-
LiteFlowNet 3 [34]	1/4	5	471	185G	7.5M	-	-	-	2.99	4.45	7.34
LiteFlowNet [12]	1/8	5	559	320G	5.3M	2.48	4.04	28.5	4.53	5.38	9.38
Flow1D	1/8	12	648	703G	5.7M	1.98	3.27	22.9	2.23	3.80	6.27
RAFT	1/8	12	780	805G	5.2M	**1.43**	**2.71**	**17.4**	**1.60**	**2.85**	**5.10**
FlowNet 2 [35]	1/4	23	787	409G	162.5M	2.02	3.14	30.3	4.16	5.74	11.48
GMFlow [20]	1/4	2	**1860**	1109G	**4.7M**	1.08	2.48	23.4	1.74	2.90	9.32
SKFlow [36]	1/8	12	2719	1167G	6.2M	1.22	2.46	15.5	1.28	2.27	4.84
CRAFT [19]	1/8	12	2749	1809G	6.3M	1.27	2.79	17.5	1.45	2.42	4.79
GMFlowNet [21]	1/8	12	3232	**1083G**	9.3M	1.14	2.71	15.4	1.39	2.65	4.79
FlowFormer [17]	1/8	12	3653	1812G	16.1M	**0.95**	**2.35**	**14.7**	**1.14**	**2.18**	**4.68**

block. Second, we equip the decoder with a gated upsample module [38] but also use NeXt1D layers to reduce the cost of keeping the decoding state across different pyramid levels. Finally, since we start our decoding stage with small feature maps, we do not need RAFT's correlation pooling trick [10] to explore larger contexts, which leads to a noticeable reduction in decoder parameters.

Our pyramid decoder uses iterative refinement [33], [39] applied to feature pyramids to produce highly detailed flow estimations gradually. We uniformly distribute the total refinement iterations N across all pyramid levels as $N_L = \lceil \frac{N}{L} \rceil$. We use the NeXt1D block at each level to refine the previous flow estimation N_L times (see Fig. 2). Before moving to the next pyramid level, we forward the current decoder state alongside finer-grained context features through the gated upsample module to update the state.

We add a convex upsample module [10] at the last pyramid level to produce sharper flow fields at the original resolution. Our experiments use the convex upsample at the pyramid level with stride 8. The remaining levels are upsampled using bilinear interpolation.

E. Loss function

The adopted loss term $\mathcal{L}$ is similar to feature pyramid architectures based on RAFT [10], [38], [40], where different upsampling functions ϕ_l are used depending on the pyramid level l. We apply convex upsampling for outputs with stride 8 and bilinear upsampling otherwise to resize the network outputs to the original resolution and then compute the L1 norm to the groundtruth:

$$\mathcal{L} = \sum_{l=1}^{L} \sum_{n=1}^{N_L} \gamma^{(L-l+1)N_L - n} \| \phi_l(F_{l,n}) - F_{GT} \|_1, \quad (2)$$

where $\gamma < 1$ is a scalar to decrease the significance of earlier predictions, $F_{l,n}$ is one of the intermediate predictions of our model, and F_{GT} is the groundtruth optical flow.

IV. EXPERIMENTAL RESULTS

A. Implementation details

The training pipeline starts by pretraining on the FlyingChairs dataset [11] followed by the FlyingThings3D [41]. We use a batch size of 8 and pretrain for 10 epochs in the former dataset and a batch of 4 and 10 epochs for the latter. For the MPI-Sintel benchmark [42], we fine-tune the model for 4 epochs in a mixed dataset combining FlyingThings3D, KITTI 2015 [43], HD1K [44], and Sintel samples. For KITTI, we start from the Sintel model and further fine-tune it for 300 more epochs on the KITTI dataset. The model is trained on one NVIDIA RTX3090 GPU, with the same

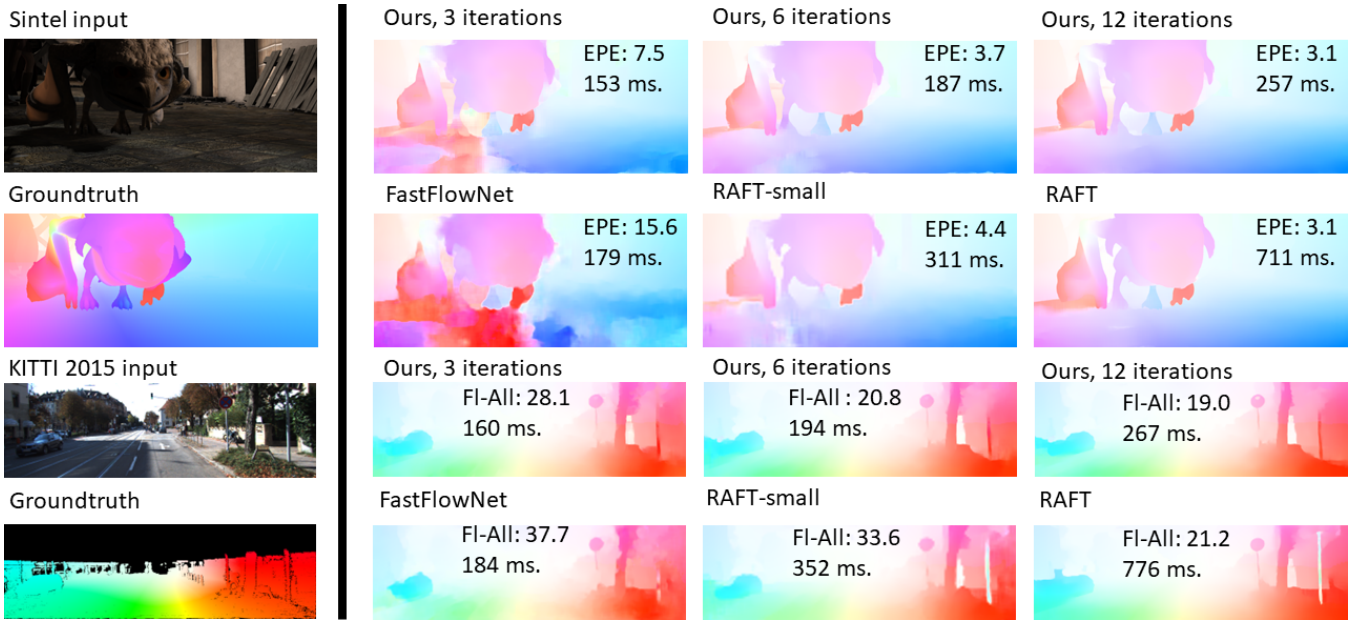

Fig. 5. Qualitative results of comparable methods at different time requirements. Our method can be as fast as FastFlowNet while producing noticeably better results. We can also increase the number of refinements and achieve results comparable to RAFT while running at almost three times its speed.

optimization and parameters as RAFT [10]. More specific details are available in the source code.

B. Comparison with the state-of-the-art

1) Quantitative results: We compare RAPIDFlow against other methods in Tab. I. We test five variants of our method that offer different accuracy and speed tradeoffs (see Tab. II). We follow the standard evaluation metrics and adopt the average End-Point Error (EPE) for Sintel and the percentage of outliers (Fl-All) for KITTI 2015 results. We achieve the best results in most time categories while running at the fastest speeds. RAFT-small and FastFlowNet still have a size advantage, but RAPIDFlow runs faster and has better accuracy. One of the reasons we outspeed FastFlowNet despite the higher FLOPs is that FastFlowNet makes heavy use of poolings, which are not counted as FLOPs but still take processing time.

Our method offers very competitive tradeoffs between accuracy and speed. We achieve an average of only a 15% accuracy gap compared to RAFT while running almost three times faster. For comparison, PWCNet is slightly slower than our method, but its accuracy is 76% worse than RAFT. Even Flow1D still shows higher errors than RAPIDFlow despite having larger computational costs. More recent methods (bottom of Tab. I) can achieve even better results, but they have significantly higher computational costs and are less suitable for embedded devices.

2) Qualitative results: We compare some qualitative results of RAPIDFlow against other state-of-the-art methods with comparable performance in Fig. 5. Compared to FastFlowNet, our method avoids large estimation errors and produces sharper results. RAFT-small suffers from a lack of details at the motion boundaries due to using eight-times bilinear upsampling. With more iterations, we can even produce some results comparable to RAFT.

3) Memory scale: Memory usage is a crucial factor in developing methods for limited-resource devices. To compare the memory usage with previous methods, we analyze how the memory requirements increase according to changes in the input size. We measure memory allocation using the pynvml [45] package to compute the difference before and after loading a model in the device. We adopt RAFT's efficient correlation layer to avoid building the full correlation volume. The results in Fig. 6 show that our method has one of the lowest memory requirements.

4) Scale robustness: One of the advantages of our recurrent design is its ability to adapt to inputs of various scales by changing the pyramid levels. Here, we conduct a study to check how a change in the scale (size) of the input affects the predicted flow. Our study consists of applying bilinear upsampling to multiply the input image by a given scale. The enlarged image is put through the network, and then the output flow is downsampled back to compare to the groundtruth resolution. RAPIDFlow adds one more pyramid level each time the input scale doubles.

As Tab. III shows, our approach maintains the most stable performance even when increasing the input by up to four times. At larger scales, we achieve the best results in all tests. Considering the degradation caused by increasing the scale from 1 to 4, our results only drop by 11% on Sintel and 2% on KITTI, significantly better than the second most scalable method, PWCNet, which shows drops of 63% and 46%, respectively. Notably, RAFT and Flow1D (a variant of RAFT) show the largest degradations by the changes in scale. This likely happens because the high-resolution feature maps they use for the matching stage are more influenced by

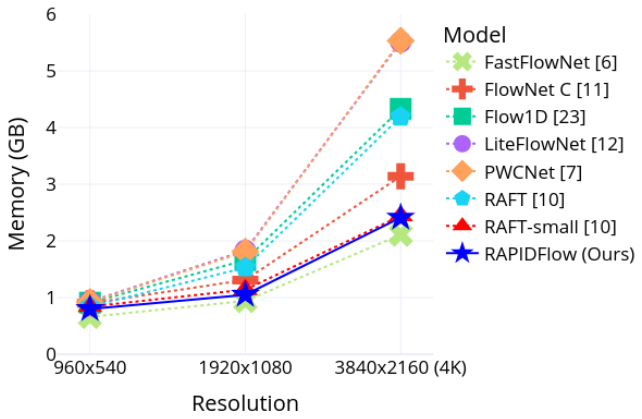

Fig. 6. Memory consumption according to the input size. Our method has low memory requirements, comparable to RAFT-small and FastFlowNet. The memory was measured using FP16 precision and RAFT's local correlation operation.

TABLE II

CONFIGURATIONS OF THE TESTED VARIATIONS OF RAPIDFLOW.

Res.	Iters. (N)	Pyr. strides	Num. levels (L)	Level iters. (N_L)
1/32	1	32	1	1
1/16	2	32, 16	2	1
1/8	3	32, 16, 8	3	1
1/8	6	32, 16, 8	3	2
1/8	12	32, 16, 8	3	4

changes in the input size than deeper pyramidal approaches.

C. Ablation study

We evaluate the importance of our proposed components by analyzing their impact on the results. All variants are trained on the FlyingChairs and FlyingThings3D datasets. The following sections discuss the contributions of each component in more detail.

1) Recurrent encoder: We first compare our recurrent encoder with a regular ResNet [27] (as in RAFT). The first comparison in Tab. IV shows that using a ResNet backbone causes a noticeable drop in performance and an increase in model size, highlighting the proposed encoder's efficiency. We also check the impact of removing the recurrency from the encoder (layers are not shared across scales). As the second comparison in Tab. IV shows, sharing the layers decreases the model size by more than 25%, with a minimal impact on the accuracy. More importantly, the recurrent version adopted in RAPIDFlow can dynamically change the feature pyramid depth, making the model more robust to changes in the input size (see Tab. III).

2) Gated upsampling: The third comparison in Tab. IV shows that maintaining the complete decoder state across all pyramid levels contributes to better estimation results.

3) Decoder 1D block: The fourth comparison in Tab. IV shows that the NeXt1D block efficiently replaces the ConvGRU (as proposed by RAFT). RAPIDFlow produces similar results to the ConvGRU variant but with 20% less parameters and FLOPs.

TABLE III

EPE RESULTS USING INPUTS OF DIFFERENT SCALES. $\Delta 1$ DENOTES THE DIFFERENCE TO THE $1\times$ RESULTS. THE FIRST- AND SECOND-BEST RESULTS ARE HIGHLIGHTED IN BOLD AND UNDERLINED.

Method	Sintel Final			KITTI 2015	
	$1\times$	$2\times$	$4\times(\Delta 1)$	$1\times$	$4\times(\Delta 1)$
FlowNet C [11]	5.6	7.5	11.2 (99%)	15.4	24.5 (59%)
FastFlowNet [6]	4.2	4.7	8.3 (99%)	12.7	24.9 (96%)
PWCNet [7]	4.1	4.5	<u>6.7</u> (63%)	10.6	<u>15.5</u> (46%)
LiteFlowNet [12]	3.9	4.8	6.9 (75%)	11.0	22.9 (108%)
Flow1D [23]	3.4	5.9	20.5 (494%)	7.6	35.7 (366%)
RAFT [10]	**2.7**	<u>3.8</u>	11.0 (305%)	**5.4**	31.8 (479%)
RAPIDFlow	<u>2.8</u>	**2.8**	**3.1** (11%)	<u>5.8</u>	**5.9** (2%)

TABLE IV

TOP: ABLATION RESULTS USING DIFFERENT ENCODER AND DECODER CONFIGURATIONS. BOTTOM: RESULTS OF RAFT USING THE SAME NUMBER OF CHANNELS OF OUR MODEL.

Method	Params	FLOPs	S.Fin (EPE)	K15 (Fl-All)
RAPIDFlow	1.6M	128G	2.89	17.7
ResNet enc.	2.2M	128G	3.05	20.8
Our enc. w/o rec.	2.3M	128G	2.87	17.2
w/o Gated Upsample	1.5M	127G	2.96	18.0
with ConvGRU dec.	2.0M	161G	2.84	17.8
RAFT-reduced	2.6M	478G	3.15	18.2

4) RAFT-reduced: We demonstrate the effectiveness of RAPIDFlow's design by comparing its results against a variation of RAFT using the same number of feature channels per layer (RAFT-reduced). The last results in Tab. IV show that RAFT-reduced still has noticeably larger computational requirements than ours, mainly due to the lack of pyramids and the use of correlation pooling [10]. We also achieve better results in both benchmarks, with a noticeable difference in the Sintel evaluation.

V. CONCLUSIONS

We presented a fully recurrent encoder-decoder structure to estimate optical flow with reduced computational costs. Our RAPIDFlow model has a recurrent encoder that can be dynamically changed during inference to produce feature pyramids of various lengths to better adapt to the input size. The experiments on an embedded system show that our model can be configured as fast as FastFlowNet or be close to RAFT while running at significantly faster speeds. Our reported results do not include optimizations such as integer quantization or specialized, efficient layers for embedded devices. Therefore, careful tuning could make our model even more efficient for deployment. For example, optimizing RAPIDFlow using the default configuration of TensorRT [46] decreases the inference time by 27% from 267 to 193 ms. Our computational cost could also be further reduced by adopting sparse operations [6], [24] and improving the efficiency of the correlation sampling [9] stage. These directions will be explored in future works.

REFERENCES

[1] S. Sun, Z. Kuang, L. Sheng, W. Ouyang, and W. Zhang, "Optical flow guided feature: A fast and robust motion representation for video action recognition," in *CVPR*, 2018, pp. 1390–1399.

[2] H. Liu, T. Lu, Y. Xu, J. Liu, W. Li, and L. Chen, "CamLiFlow: Bidirectional camera-LiDAR fusion for joint optical flow and scene flow estimation," in *CVPR*, 2022, pp. 5781–5791.

[3] T. Kipf, G. F. Elsayed, A. Mahendran, A. Stone, S. Sabour, G. Heigold, R. Jonschkowski, A. Dosovitskiy, and K. Greff, "Conditional object-centric learning from video," in *ICLR*, 2022.

[4] Y.-H. Tsai, M.-H. Yang, and M. J. Black, "Video segmentation via object flow," in *CVPR*, 2016, pp. 3899–3908.

[5] Z. Teed and J. Deng, "DROID-SLAM: Deep visual SLAM for monocular, stereo, and RGB-D cameras," in *NeurIPS*, 2021, pp. 16 558–16 569.

[6] L. Kong, C. Shen, and J. Yang, "Fastflownet: A lightweight network for fast optical flow estimation," in *ICRA*, Mar. 2021.

[7] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume," in *CVPR*, 2018, pp. 8934–8943.

[8] A. Alhawwary, J. Mustaniemi, and J. Heikkilä, "PatchFlow: A two-stage patch-based approach for lightweight optical flow estimation," in *ACCV*, 2022.

[9] R. Garrepalli, J. Jeong, R. C. Ravindran, J. M. Lin, and F. M. Porikli, "DIFT: Dynamic iterative field transforms for memory efficient optical flow," in *CVPR Workshops*, 2023.

[10] Z. Teed and J. Deng, "RAFT: recurrent all-pairs field transforms for optical flow," in *ECCV*, 2020, pp. 402–419.

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

[12] T.-W. Hui, X. Tang, and C. C. Loy, "LiteFlowNet: a lightweight convolutional neural network for optical flow estimation," in *CVPR*, 2018, pp. 8981–8989.

[13] Z. Yin, T. Darrell, and F. Yu, "Hierarchical discrete distribution decomposition for match density estimation," in *CVPR*, 2019, pp. 6037–6046.

[14] S. Jiang, D. Campbell, Y. Lu, H. Li, and R. Hartley, "Learning to estimate hidden motions with global motion aggregation," in *ICCV*, 2021, pp. 9752–9761.

[15] F. Zhang, O. J. Woodford, V. A. Prisacariu, and P. H. S. Torr, "Separable Flow: Learning motion cost volumes for optical flow estimation," in *ICCV*, 2021, pp. 10 787–10 797.

[16] J. Jeong, J. M. Lin, F. Porikli, and N. Kwak, "Imposing consistency for optical flow estimation," in *CVPR*, 2022, pp. 3171–3181.

[17] Z. Huang, X. Shi, C. Zhang, Q. Wang, K. C. Cheung, H. Qin, J. Dai, and H. Li, "FlowFormer: A transformer architecture for optical flow," in *ECCV*, 2022, pp. 668–685.

[18] A. Luo, F. Yang, X. Li, and S. Liu, "Learning optical flow with kernel patch attention," in *CVPR*, 2022, pp. 8896–8905.

[19] X. Sui, S. Li, X. Geng, Y. Wu, X. Xu, Y. Liu, R. S. M. Goh, and H. Zhu, "CRAFT: Cross-attentional flow transformer for robust optical flow," in *CVPR*, 2022, pp. 17 581–17 590.

[20] H. Xu, J. Zhang, J. Cai, H. Rezatofighi, and D. Tao, "GMFlow: Learning optical flow via global matching," in *CVPR*, 2022, pp. 8121–8130.

[21] S. Zhao, L. Zhao, Z. Zhang, E. Zhou, and D. N. Metaxas, "Global matching with overlapping attention for optical flow estimation," in *CVPR*, 2022, pp. 17 571–17 580.

[22] H. Zhou, X. Zhu, J. Zhu, Z. Han, S.-X. Zhang, J. Qin, and X.-C. Yin, "Learning correction filter via degradation-adaptive regression for blind single image super-resolution," in *ICCV*, 2023, pp. 12 331–12 341.

[23] H. Xu, J. Yang, J. Cai, J. Zhang, and X. Tong, "High-resolution optical flow from 1D attention and correlation," in *ICCV*, 2021, pp. 10 478–10 487.

[24] S. Jiang, Y. Lu, H. Li, and R. Hartley, "Learning optical flow from a few matches," in *CVPR*, 2021, pp. 16 592–16 600.

[25] Z. Zheng, N. Nie, Z. Ling, P. Xiong, J. Liu, H. Wang, and J. Li, "DIP: Deep inverse patchmatch for high-resolution optical flow," in *CVPR*, 2022, pp. 8915–8924.

[26] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "PatchMatch: a randomized correspondence algorithm for structural image editing," *ToG*, vol. 28, no. 3, p. 24, 2009.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CVPR*, pp. 770–778, 2016.

[28] F. Chollet, "Xception: deep learning with depthwise separable convolutions," in *CVPR*, 2017, pp. 1251–1258.

[29] M.-H. Guo, C.-Z. Lu, Z.-N. Liu, M.-M. Cheng, and S.-M. Hu, "Visual attention network," *Computational Visual Media*, vol. 9, no. 4, 2023.

[30] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A ConvNet for the 2020s," in *CVPR*, 2022, pp. 11 966–11 976.

[31] X. Ding, X. Zhang, Y. Zhou, J. Han, G. Ding, and J. Sun, "Scaling up your kernels to 31×31: Revisiting large kernel design in CNNs," in *CVPR*, 2022, pp. 11 953–11 965.

[32] S.-X. Zhang, X. Zhu, J.-B. Hou, C. Yang, and X.-C. Yin, "Kernel proposal network for arbitrary shape text detection," *TNNLS*, vol. 34, pp. 8731–8742, 2023.

[33] J. Hur and S. Roth, "Iterative residual refinement for joint optical flow and occlusion estimation," in *CVPR*, 2019, pp. 5754–5763.

[34] T.-W. Hui and C. C. Loy, "LiteFlowNet3: Resolving correspondence ambiguity for more accurate optical flow estimation," in *ECCV*, 2020, p. 169–184.

[35] 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 *CVPR*, vol. 2, 2017, pp. 2462–2470.

[36] S. Sun, Y. Chen, Y. Zhu, G. Guo, and G. Li, "SKFlow: Learning optical flow with super kernels," in *NeurIPS*, 2022.

[37] N. Ballas, L. Yao, C. J. Pal, and A. C. Courville, "Delving deeper into convolutional networks for learning video representations," in *ICLR*, 2016.

[38] H. Morimitsu, X. Zhu, X. Ji, and X.-C. Yin, "Recurrent partial kernel network for efficient optical flow estimation," in *AAAI*, 2024.

[39] S.-X. Zhang, X. Zhu, L. Chen, J.-B. Hou, and X.-C. Yin, "Arbitrary shape text detection via segmentation with probability maps," *TPAMI*, vol. 45, pp. 2736–2750, 2023.

[40] A. Jahedi, L. Mehl, M. Rivinius, and A. Bruhn, "Multi-Scale RAFT: Combining hierarchical concepts for learning-based optical flow estimation," in *ICIP*, 2022, pp. 1236–1240.

[41] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *CVPR*, 2016, pp. 4040–4048.

[42] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *ECCV*, 2012, pp. 611–625.

[43] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *CVPR*, 2015, pp. 3061–3070.

[44] D. Kondermann, R. Nair, K. Honauer, K. Krispin, J. Andrulis, A. Brock, B. Gussefeld, M. Rahimimoghaddam, S. Hofmann, C. Brenner, *et al.*, "The HCI benchmark suite: Stereo and flow ground truth with uncertainties for urban autonomous driving," in *CVPR Workshops*, 2016, pp. 19–28.

[45] NVIDIA, "pynvml," https://pypi.org/project/pynvml/.

[46] ——, "TensorRT," https://github.com/NVIDIA/TensorRT.