NeuFlow v2: Push High-Efficiency Optical Flow To the Limit

Zhiyong Zhang[1], Aniket Gupta[2], Huaizu Jiang[2], Hanumant Singh[1]

***Abstract*— Real-time high-accuracy optical flow estimation is crucial for various real-world applications. While recent learning-based optical flow methods have achieved high accuracy, they often come with significant computational costs. In this paper, we propose a highly efficient optical flow architecture that addresses both high accuracy and computational cost concerns. The model builds upon NeuFlow v1, incorporating new components such as an extremely lightweight backbone and a simple RNN module that improve refinement accuracy while reducing computational demands. We evaluate our approach on the Jetson Orin Nano and RTX 2080, demonstrating significant efficiency improvements across multiple GPU platforms. Our model achieves a 10×-70× speedup compared to several state-of-the-art methods while maintaining comparable accuracy on the KITTI and Sintel datasets. Our approach reaches approximately 10 FPS on edge computing platforms, marking a significant breakthrough in deploying complex computer vision tasks such as SLAM on small robots. The full training and evaluation code is available at https://github.com/neufieldrobotics/NeuFlow_v2.**

I. Introduction

The development of high-accuracy optical flow estimation algorithms has seen substantial progress in recent years [1]. Starting from FlowNet [2], learning-based methods for optical flow have shifted towards feature learning for matching, moving away from traditional hand-crafted features like those in Lucas-Kanade [3] or SIFT [4] [5]. Despite these advances, early optical flow methods struggled with significant challenges such as large displacement and generalizing to real-world data [6]. More recent deep learning approaches have mitigated these issues, albeit with an increased computational cost [7], [8].

The problem of model generalization arises in the early stages of optical flow estimation with deep learning [6]. Due to the difficulty in collecting ground truth optical flow data in the real-world, simulation is predominantly used to generate sufficient training data [2], [9]. However, training with simulation data can lead to overfitting due to unrealistic illumination, reflections, and monotonous scenes [10], [11]. Early optical flow methods that rely on CNNs struggle with handling large displacements [12], and their one-shot architectures do not generalize well to real-world data [2], [12], [13], [14].

Starting with RAFT [6], iterative refinements have partially mitigated the generalization issue while also capturing larger motions [15]. Recent research has further improved accuracy and generalization by incorporating the latest modules, such as transformer [16], Partial Kernel Convolution [17], Super Kernel [18], and more. However, these recent methods generally require more computation time compared to earlier methods due to the iterative refinement process [19]. Some models need over 30 iterations to generate a stable optical flow [6], while others reduce the number of iterations but increase the computational load of each iteration [17], [20].

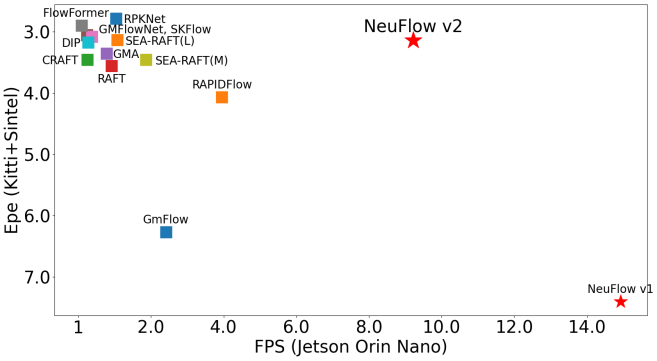

Fig. 1: End Point Error (EPE) of KITTI and Sintel datasets vs. Frames Per Second (FPS) throughput on an edge computing platform (Jetson Orin Nano). Individual points represent a broad class of optical flow methods. Our algorithm is comparable in accuracy but significantly more efficient, approaching an order of magnitude improvement in computational complexity. All models were trained solely on the FlyingThings and FlyingChairs datasets.

This paper presents an enhanced version of NeuFlow v1 [21], maintaining a similiar inference speed while nearing state-of-the-art accuracy across multiple datasets. To address the large displacement problem, we use transformer-based [22] cross-attention and global matching [19] to estimate an initial optical flow, unrestricted by the receptive field. The iterative refinement module has been shown to improve model generalization, and we employ our simple RNN module for local refinement. To further enhance model efficiency, we developed a new simple backbone, which is even more lightweight than NeuFlow v1. The overall accuracy vs. inference time comparison is shown in Fig. 1. NeuFlow v1 had low accuracy on the KITTI dataset [23], so NeuFlow v2 was developed to achieve better model generalization and perform well on real-world data as shown in Fig. 2.

The main contributions of this paper are as follows:

1. Simple Backbone: A simple CNN-based backbone that extracts low-level features from multi-scale images. Instead

[1]Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115. zhang.zhiyo@northeastern.edu ha.singh@northeastern.edu [2]Khoury College of Computer Sciences, Northeastern University, Boston, MA 02115. h.jiang@northeastern.edu HJ is supported by the National Science Foundation under Award IIS-2310254.

Fig. 2: NeuFlow v2 Examples: We run NeuFlow v2 in real-world scenarios to showcase the model's generalization capabilities.

of commonly used architectures like ResNet [24] or Feature Pyramid Network [25], this lightweight backbone is found to be sufficient for getting accurate Optical Flow.

2. Light-weight and Efficient Iterative Refinement Module: A simple recurrent network module capable of outputting the hidden state and decoded refined optical flow. Instead of using time-consuming modules like LSTM [26] or GRU [27], we propose a simpler RNN module that is more lightweight and achieves higher accuracy for local refinement.

II. Related Work

FlowNet [2] was the first deep learning-based optical flow estimation method, introducing two variants: FlowNetS and FlowNetC, along with the synthetic FlyingChairs dataset for end-to-end training and benchmarking. An improved version, FlowNet 2.0 [13], fused cascaded FlowNets with a small displacement module, decreasing the estimation error by more than 50% while being marginally slower.

Following FlowNet 2.0 [13], researchers developed more lightweight optical flow methods. SPyNet [14] is 96% smaller than FlowNet in terms of model parameters. PWC-Net [12] is 17 times smaller than FlowNet 2. LiteFlowNet [28] is 30 times smaller in model size and 1.36 times faster in running speed compared to FlowNet 2. LiteFlowNet 2 [29] improved optical flow accuracy on each dataset by around 20% while being 2.2 times faster. LiteFlowNet 3 [30] further enhanced flow accuracy. RapidFlow [31] combines efficient NeXt1D convolution blocks with a fully recurrent structure to decrease computational costs. DCVNet [32] proposes constructing cost volumes with different dilation factors to capture small and large displacements simultaneously. NeuFlow v1 [21], our previous work, is the fastest optical flow method, being over ten times faster than mainstream optical flow methods while maintaining comparable accuracy on the Sintel and FlyingThings datasets.

More recently, RAFT [6] used recurrent all-pairs field transforms to achieve strong cross-dataset generalization. Following RAFT, GMA [15] introduced a global motion aggregation module to improve estimation in occluded regions. GMFlow [19] reformulated optical flow as a global matching problem. GMFlowNet [16] efficiently performed global matching by applying argmax on 4D cost volumes. CRAFT [7] used a Semantic Smoothing Transformer layer to make features more global and semantically stable. FlowFormer [8], [33] encodes the 4D cost tokens into a cost memory with alternate-group transformer layers in a latent space. SKFlow [18] benefits from super kernels to complement the absent matching information and recover the occluded motions. DIP [34] introduced the first end-to-end PatchMatch-based method, achieving high-precision results with lower memory. RPKNet [17] utilized Partial Kernel Convolution layers to produce variable multi-scale features and efficient Separable Large Kernels to capture large context information. Sea-Raft [20] proposed a new loss (mixture of Laplace) and directly regressed an initial flow for faster convergence. Many works have also been proposed to either reduce computational costs or improve flow accuracy [35].

III. Proposed Approach: NeuFlow v2

We introduce NeuFlow v2, an improved version that maintains the highest inference speed while approaching state-of-the-art accuracy across multiple datasets. We developed a new simple backbone module that efficiently extracts low-level features for optical flow tasks, and a simple RNN refinement module that uses only CNNs, without GRU or LSTM, improving both accuracy and efficiency simultaneously. With additional efficient modules, including cross-attention and global matching, we have created a stable optical flow model capable of running in real-time on edge devices.

A. Simple Backbone

In NeuFlow v1, we proposed a similar shallow backbone for extracting low-level features from multi-scale images. In NeuFlow v2, we have eliminated redundant parts and retained only the effective components. The intuition behind this design is that sufficient low-level features are more crucial than high-level features in optical flow tasks. Fig. 4 details the new simple backbone. We extract features from 1/2, 1/4, and 1/8 scale images using a CNN block composed of convolution, normalization, and ReLU layers.

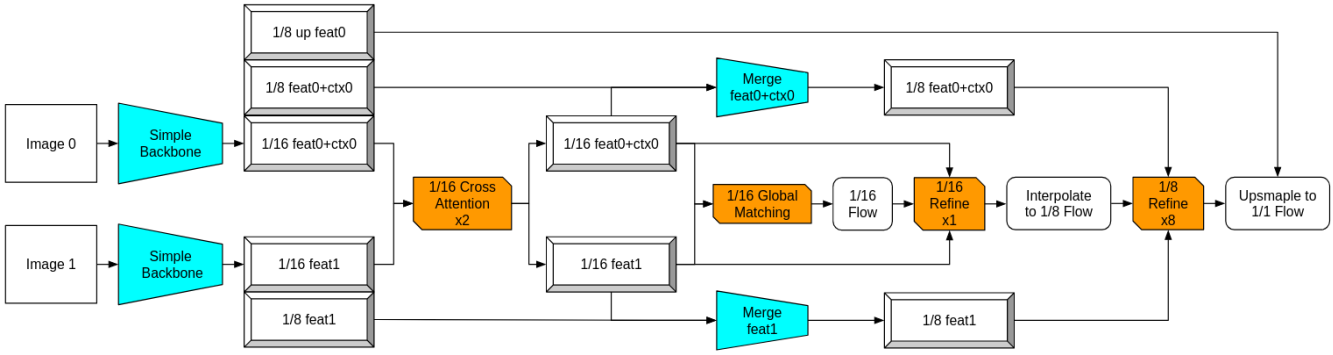

Fig. 3: NeuFlow v2 Architecture: We begin with a simple CNN backbone that outputs features and context at 1/8 and 1/16 scales for both images. The feature vectors at a 1/16 scale are then fed into cross-attention layers for feature enhancement. Next, we perform global matching to obtain an initial 1/16 flow, which is refined through one iteration. This flow is upsampled to a 1/8 scale and further refined over eight iterations. The refined 1/8 flow is then upsampled to full resolution using a convex upsampling module. The entire design follows the principle of global attention followed by local refinement. Details of the simple backbone and refinement module are presented in Figures 3 and 4.

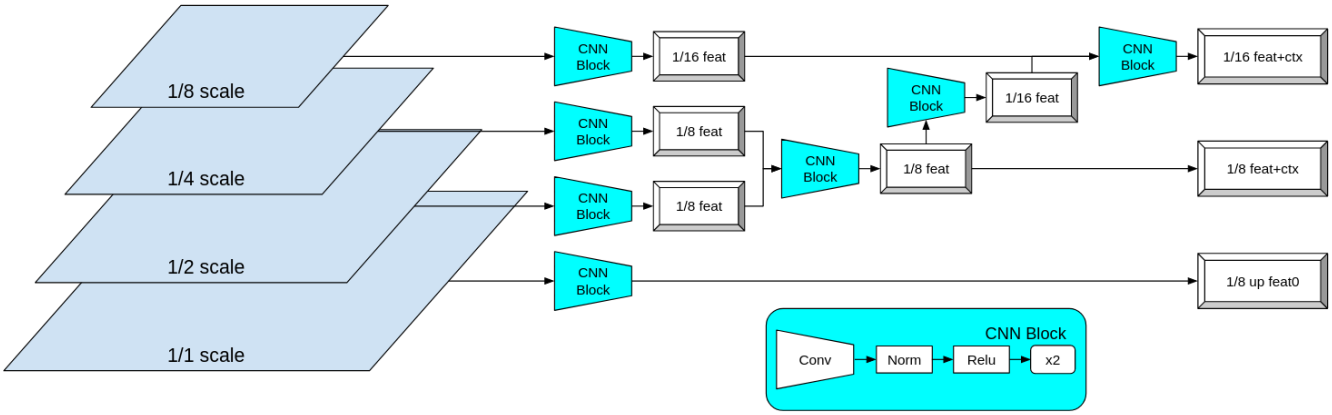

Fig. 4: NeuFlow Simple Backbone: We downsample the image into various scales, ranging from 1/1 to 1/8. A simple CNN block, consisting of two convolutional-norm-ReLU layers, is used to extract low-level features directly from the image. Then, using the same CNN block, we merge and resize these features into 1/8 and 1/16 scale outputs. The backbone outputs 1/8 and 1/16 scale features and context for further flow estimation, along with an additional 1/8 scale feature for convex upsampling. We believe that this simple backbone can extract better low-level features, which are more crucial than high-level features in optical flow tasks.

This same CNN block is used to concatenate and resize these features into the desired output scale, specifically to 1/16 scale features and context, as well as 1/8 scale features and context. Features are used for correlation computation, while context is used for flow refinement.

Note that the 1/1 scale image is used solely for convex upsampling and is not involved in estimating the 1/8 resolution flow. The ablation study in Section 4 demonstrates that features extracted from the full 1/1 scale image can cause overfitting on the training set (FlyingThings) and do not improve accuracy on unseen data (Sintel, KITTI).

B. Cross-Attention And Global Matching

Cross-attention is used to exchange information between images globally, enhancing the distinctiveness of matching features and reducing the similarity of unmatched features. Global matching is then applied to find corresponding features globally, enabling the model to handle large pixel displacements, such as in fast-moving camera situations. To reduce the computational burden of cross-attention and global matching, we operate on 1/16 scale features instead of 1/8 scale.

Similar to NeuFlow v1 and GMFlow [19], we utilize Transformers to implement cross-attention and global matching. While flow self-attention is typically used to refine pixels based on self-similarity after global matching, we instead use our simple RNN module to iteratively refine the estimated flow.

C. Simple RNN Refinement

We first compute the correlation within nearby 9x9 neighborhoods and warp this correlation using the estimated flow.

Sintel (1024×436)	train, clean	train, final	RTX 2080 (s)	Jetson Orin Nano (s)	Batch Size (8G)
GMFlow (1 iter)	1.50	2.96	0.046	0.404	24
RAPIDFlow (12 iters)	1.58	2.94	0.038	0.252	103
CRAFT (32 iters)	1.27	2.79	0.347	N/A	2
RAFT (32 iters)	1.43	2.71	0.125	1.060	24
GMA (32 iters)	1.30	2.74	0.152	1.245	17
SKFlow (32 iters)	1.22	2.46	0.365	4.181	17
GMFlowNet (32 iters)	1.14	2.71	0.227	2.626	12
FlowFormer	**1.01**	**2.40**	1.007	11.196	4
SEA-RAFT(M) (4 iters)	1.21	4.04	0.061	0.524	23
DIP (20 iters)	1.30	2.82	0.499	3.615	12
RPKNet (12 iters)	1.12	2.45	0.158	0.947	68
SEA-RAFT(L) (4 iters)	1.19	4.11	0.096	0.910	22
NeuFlow v1	1.66	3.13	**0.009**	**0.064**	**107**
NeuFlow v2 (9 iters)	1.24	2.67	0.015	0.103	26
Kitti (1242×375)	Kitti epe	Kitti F1	RTX 2080 (s)	Jetson Orin Nano (s)	Batch Size (8G)
GMFlow (1 iter)	10.3	33.6	0.055	0.426	22
RAPIDFlow (12 iters)	5.87	17.7	0.045	0.254	99
CRAFT (32 iters)	4.88	17.5	0.385	N/A	2
RAFT (32 iters)	5.04	17.4	0.129	1.126	23
GMA (32 iters)	4.69	17.1	0.161	1.343	16
SKFlow (32 iters)	4.27	15.5	0.408	4.478	16
GMFlowNet (32 iters)	4.24	15.4	0.231	2.761	12
FlowFormer	4.09	14.7	1.002	11.172	5
SEA-RAFT(M) (4 iters)	4.29	14.2	0.075	0.549	22
DIP (20 iters)	4.29	13.7	0.523	3.767	12
RPKNet (12 iters)	3.79	13.0	0.157	0.976	64
SEA-RAFT(L) (4 iters)	**3.62**	**12.9**	0.101	0.953	20
NeuFlow v1	12.4	32.5	**0.010**	**0.070**	**115**
NeuFlow v2 (9 iters)	4.33	15.3	0.015	0.114	21

TABLE I: This table compares the latest optical flow methods based on their highest accuracy. All models were trained on the FlyingThings and FlyingChairs (C+T) datasets and evaluated on the Sintel and KITTI training sets. Inference time was measured on both an RTX 2080 and an edge computing device, the Jetson Orin Nano, using half-precision models. Batch size was determined during batch inference on an 8GB memory GPU with half-precision models. CRAFT is marked as N/A because the Jetson Orin Nano crashed during inference. The table shows that NeuFlow v2 achieves good accuracy while being significantly faster than any other models.

We then concatenate the warped correlation, context features, estimated flow, and previous hidden state, processing them through eight layers of simple 3x3 convolutional layers followed by ReLU activation to output the refined optical flow and updated hidden state. The simple RNN refinement is performed at both the 1/16 scale flow and the 1/8 scale flow (Fig. 5).

To address the vanishing or exploding gradient problem, most RNNs use GRU or LSTM modules to compute the current hidden state based on the previous hidden state and current inputs (warped correlation, context, and flow) and to decode the estimated flow using the current hidden state. However, GRU and LSTM modules often have too few layers to effectively merge the current input with the previous hidden state. In contrast, we use deep CNN layers to effectively merge the inputs (warped correlation, context, and flow) with the hidden state. This approach has been tested to avoid unstable gradient issues and significantly improve accuracy, as detailed in the ablation study in Section 4.

HardTanh [36] is applied before the hidden state to constrain the feature values within a specific range (in our case, -4 to 4) and to address numerical stability issues. Using traditional Tanh can lead to extremely large or small values when the hidden state values approach -1 or 1, potentially causing overflow. HardTanh helps mitigate this problem by maintaining the values within a manageable range.

D. Multi-Scale Feature/Context Merge

In NeuFlow, our simple backbone lacks the depth of convolution, resulting in features having a small receptive field. Cross-attention at the 1/16 scale provides global attention, offering a global feature/context. However, 1/8 scale features/context do not have a global receptive field. Therefore, we merge the 1/16 global features/context with the 1/8 local features/context to ensure that the 1/8 scale features/context contain both global and local information (Fig. 6).

The merge block consists of two layers of CNNs with ReLU activation and normalization. In practice, features and context are merged individually using the same merge block structure.

IV. Experiments

A. Training and Evaluation Datasets

We first train the model solely on the FlyingThings [9] dataset for a fair comparison with other models, as most models have undergone the same procedure. Additionally, we trained the model using a mixed dataset comprising Sintel [9], KITTI [37], and HD1K for real-world applications. For evaluation, we followed the common practice of using the Sintel and KITTI datasets to demonstrate the model's

	things train	things val	sintel clean train	sintel final train	RTX 2080	kitti epe	kitti F1	RTX 2080
Full	2.97	3.44	1.24	2.67	0.015	4.33	15.3	0.015
Backbone Module								
YOLO v8 Backbone	2.56	3.15	1.34	2.96	0.016	4.87	17.0	0.016
1/1 backbone	2.73	3.28	1.19	2.81	0.016	4.92	15.7	0.016
Refine Module								
-2 layers	2.85	3.41	1.22	2.76	0.014	5.21	16.5	0.014
+2 layers	2.67	3.17	1.18	2.80	0.016	4.62	15.6	0.016
half feature dimension	3.26	3.79	1.37	2.91	0.014	6.66	21.7	0.014
use ConvGRU	3.00	3.62	1.28	3.87	0.018	7.36	20.4	0.019
Architecture								
remove cross attention	3.99	4.37	1.60	4.17	0.014	5.18	16.3	0.013
remove global match, 1/16 refine=4	2.85	3.24	1.21	2.88	0.015	4.89	16.1	0.014
remove 1/16 refine	3.00	3.79	1.40	4.06	0.013	5.52	19.6	0.013

TABLE II: Ablation Study: We conducted ablation studies on each module, including replacing the simple backbone with a YOLO backbone, adding full-resolution features to the simple backbone, adding/reducing layers in the RNN refinement module, reducing feature dimensions, and replacing one CNN layer with a ConvGRU layer. For the overall architecture, we experimented with removing cross-attention layers, removing global matching while conducting additional refinement of the 1/16 scale flow, and removing the 1/16 refinement entirely.

	things train	things val	sintel clean train	sintel final train	RTX 2080	kitti epe	kitti F1	RTX 2080
1/16 iters=1	**2.97**	**3.44**	**1.24**	**2.67**	**0.015**	**4.33**	**15.3**	**0.015**
1/16 iters=3	2.87	3.35	1.22	2.80	0.016	4.24	15.3	0.016
1/16 iters=5	2.86	3.37	1.22	2.82	0.017	4.17	15.2	0.018
1/8 refine iters=4	3.14	3.63	1.31	2.76	0.011	5.58	17.9	0.012
1/8 refine iters=6	3.02	3.50	1.26	2.69	0.013	4.66	15.9	0.013
1/8 refine iters=8	**2.97**	**3.44**	**1.24**	**2.67**	**0.015**	**4.33**	**15.3**	**0.015**
1/8 refine iters=10	2.95	3.41	1.23	2.68	0.017	4.17	15.0	0.017

TABLE III: This table shows how different iterations affect both accuracy and inference time. The default configuration is 1 iteration of 1/16 refinement and 8 iterations of 1/8 refinement.

generalization capabilities. We followed the same procedure and data augmentation settings as RAFT, utilizing RAFT's [6] training and evaluation code.

For real-world use cases, we also train the model with a mixed dataset that includes FlyingThings, Sintel, KITTI, HD1K, and VIPER [38]. The pretrained model is available in our repository.

B. Comparision with Latest Optical Flow Methods

We compare our method to several state-of-the-art optical flow methods renowned for their superior accuracy. We also measure the computation time on both the RTX 2080 and the edge computing platform Jetson Orin Nano, considering image sizes of 1024×436 for Sintel and 1242×375 for KITTI. The inference batch size is measured on an 8GB GPU to assess the memory usage of different models.

Among these methods, Sea-Raft-Large [20] achieves the highest accuracy on KITTI. However, both Sea-Raft-Large and Sea-Raft-Medium have similar accuracy issues on the Sintel-Final dataset, whereas NeuFlow v2 runs about seven times faster. RPKNet [17] is a state-of-the-art model in terms of accuracy across all datasets, and NeuFlow v2 offers comparable accuracy while being 10 times faster. For other heavy models like DIP and FlowFormer [8], NeuFlow v2 maintains similar accuracy while being 30-100 times faster. Some models, such as GMFlow [19] and RapidFlow [31], prioritize faster inference speed but at the cost of reduced accuracy.

C. Ablation Study

Backbone Module: We tried replacing our simple backbone with the YOLO v8 backbone, which is very efficient and deeper, thus extracting more high-level features. However, the results show that this change does not improve model generalization and actually reduces accuracy on the Sintel and KITTI datasets.

As stated in the simple backbone section, we do not extract features from the full-resolution image. According to our ablation study, full-resolution features do not improve accuracy on Sintel and KITTI and tend to cause overfitting, as indicated by increased accuracy on the FlyingThings dataset.

Refine Module: We use 8 layers of CNN to output both refined optical flow and the hidden state in the refinement module. We experimented by reducing and adding 2 layers to observe the impact. The results show that reducing the number of layers slightly decreases accuracy, while adding layers does not improve accuracy. This indicates that eight layers provide a balanced configuration.

Our default feature dimensions are 128 for the 1/16 refinement and 96 for the 1/8 refinement. Reducing the feature dimensions by half (64 for the 1/16 refinement and 48 for the 1/8 refinement) results in a significant drop in accuracy.

In the refinement module, we use multiple layers of CNN to output the hidden state. We also experimented with replacing the first CNN layer with a ConvGRU to output the

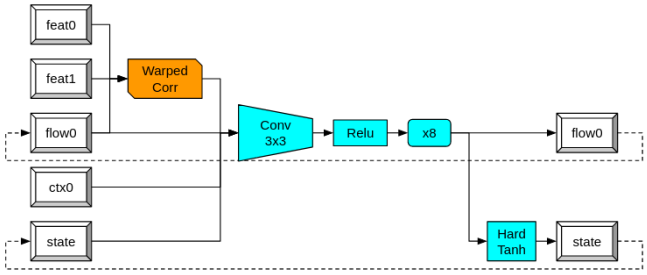

Fig. 5: NeuFlow Simple RNN Refinement: We first compute the correlation within nearby pixels and warp these values using the currently estimated flow. The warped correlation, current estimated flow, context features, and hidden state are then fed into a series of 3x3 convolution layers followed by ReLU activation, repeated eight times. At the end of these layers, the network outputs both the refined flow and an updated hidden state for the next iteration. Instead of using GRU or LSTM modules, we simply use CNNs to generate the hidden state. A hard tanh function is applied to constrain the hidden state within a certain range, ensuring numerical stability.

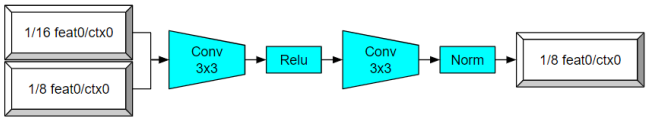

Fig. 6: NeuFlow Merge Module: We concatenate the 1/8 scale features/context with the interpolated 1/16 scale features/context and use simple CNN blocks to output 1/8 scale features/context, thereby incorporating both global and local information.

hidden state and using the remaining seven CNN layers to decode a refined flow. The results show that this approach significantly reduces accuracy.

Architecture: Cross-attention is used to exchange information between two input images globally. Removing it does not significantly affect accuracy on the KITTI dataset, but it causes a substantial drop in accuracy on the Sintel dataset.

Global matching provides the initial optical flow, which can handle large motions. Removing it and adding three more iterations of 1/16 refinement helps address large motion issues. The drop in accuracy indicates that global matching is working efficiently without requiring multiple refinements.

We only perform one iteration of 1/16 refinement. If we remove this and rely entirely on 1/8 refinement, overfitting occurs, as the training set accuracy remains unchanged while the accuracy drops significantly on all validation sets.

Different Iterations:

The default iteration count for 1/16 refinement is set to one, as additional iterations do not significantly improve accuracy. In contrast, 1/8 refinement benefits from more iterations. The default eight iterations already provide decent accuracy, but adding more iterations can further improve accuracy at the cost of increased inference time.

V. Conclusions and Future Work

In this paper, we proposed an efficient optical flow method, which accruacy is close to state-of-the-art while getting 10 times faster, enable real-time inference on edge computing device. We have released the code and model weights of NeuFlow_v2 (https://github.com/neufieldrobotics/NeuFlow_v2).

However, we also recognized that memory comsupmtion is heavy, which is caused by correlation computation. Various modules has addressed the problem [31], [17] which can be used in our architecture.

Our method also contains too many parameters (9 million), primarily due to the simple backbone and simple RNN refinement module, which rely heavily on CNNs. This can potentially lead to overfitting with training data. Many efficient modules can be replaced to reduce the number of parameters. For example, MobileNets [39], [40], [41] use depthwise separable convolutions, while ShuffleNet [42] utilizes pointwise group convolution.

References

[1] M. Zhai, X. Xiang, N. Lv, and X. Kong, "Optical flow and scene flow estimation: A survey," *Pattern Recognition*, vol. 114, p. 107861, 2021.

[2] 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 *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2758–2766.

[3] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *IJCAI'81: 7th international joint conference on Artificial intelligence*, vol. 2, 1981, pp. 674–679.

[4] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, pp. 91–110, 2004.

[5] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman, "Sift flow: Dense correspondence across different scenes," in *Computer Vision–ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part III 10*. Springer, 2008, pp. 28–42.

[6] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 2020, pp. 402–419.

[7] X. Sui, S. Li, X. Geng, Y. Wu, X. Xu, Y. Liu, R. Goh, and H. Zhu, "Craft: Cross-attentional flow transformer for robust optical flow," in *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2022, pp. 17 602–17 611.

[8] 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 *European conference on computer vision*. Springer, 2022, pp. 668–685.

[9] N. Mayer, E. Ilg, P. Hausser, 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4040–4048.

[10] W. Wang, D. Zhu, X. Wang, Y. Hu, Y. Qiu, C. Wang, Y. Hu, A. Kapoor, and S. Scherer, "Tartanair: A dataset to push the limits of visual slam," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 4909–4916.

[11] L. Mehl, J. Schmalfuss, A. Jahedi, Y. Nalivayko, and A. Bruhn, "Spring: A high-resolution high-detail dataset and benchmark for scene flow, optical flow and stereo," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 4981–4991.

[12] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8934–8943.

[13] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2462–2470.

[14] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4161–4170.

[15] S. Jiang, D. Campbell, Y. Lu, H. Li, and R. Hartley, "Learning to estimate hidden motions with global motion aggregation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 9772–9781.

[16] S. Zhao, L. Zhao, Z. Zhang, E. Zhou, and D. Metaxas, "Global matching with overlapping attention for optical flow estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 17 592–17 601.

[17] H. Morimitsu, X. Zhu, X. Ji, and X.-C. Yin, "Recurrent partial kernel network for efficient optical flow estimation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 5, 2024, pp. 4278–4286.

[18] S. Sun, Y. Chen, Y. Zhu, G. Guo, and G. Li, "Skflow: Learning optical flow with super kernels," *Advances in Neural Information Processing Systems*, vol. 35, pp. 11 313–11 326, 2022.

[19] H. Xu, J. Zhang, J. Cai, H. Rezatofighi, and D. Tao, "Gmflow: Learning optical flow via global matching," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 8121–8130.

[20] Y. Wang, L. Lipson, and J. Deng, "Sea-raft: Simple, efficient, accurate raft for optical flow," *arXiv preprint arXiv:2405.14793*, 2024.

[21] Z. Zhang, H. Jiang, and H. Singh, "Neuflow: Real-time, high-accuracy optical flow estimation on robots using edge devices," *arXiv preprint arXiv:2403.10425*, 2024.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[23] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3354–3361.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[25] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[27] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[28] T.-W. Hui, X. Tang, and C. C. Loy, "Liteflownet: A lightweight convolutional neural network for optical flow estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8981–8989.

[29] ——, "A lightweight optical flow cnn—revisiting data fidelity and regularization," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 8, pp. 2555–2569, 2020.

[30] T.-W. Hui and C. C. Loy, "Liteflownet3: Resolving correspondence ambiguity for more accurate optical flow estimation," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16*. Springer, 2020, pp. 169–184.

[31] 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," 2024.

[32] H. Jiang and E. Learned-Miller, "Dcvnet: Dilated cost volume networks for fast optical flow," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 5150–5157.

[33] X. Shi, Z. Huang, D. Li, M. Zhang, K. C. Cheung, S. See, H. Qin, J. Dai, and H. Li, "Flowformer++: Masked cost volume autoencoding for pretraining optical flow estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 1599–1610.

[34] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8925–8934.

[35] L. Kong, C. Shen, and J. Yang, "Fastflownet: A lightweight network for fast optical flow estimation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 10 310–10 316.

[36] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of machine learning research*, vol. 12, pp. 2493–2537, 2011.

[37] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VI 12*. Springer, 2012, pp. 611–625.

[38] S. R. Richter, Z. Hayder, and V. Koltun, "Playing for benchmarks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2213–2222.

[39] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[40] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[41] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1314–1324.

[42] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6848–6856.