

Rethinking RAFT for Efficient Optical Flow

Navid Eslami, Farnoosh Arefi, Amir M. Mansourian, Shohreh Kasaei

Department of Computer Engineering

Sharif University of Technology

Tehran, Iran

Email: {navid.eslami, far.arefi, amir.mansourian, kasaei}@sharif.edu

Abstract—Despite significant progress in deep learning-based optical flow methods, accurately estimating large displacements and repetitive patterns remains a challenge. The limitations of local features and similarity search patterns used in these algorithms contribute to this issue. Additionally, some existing methods suffer from slow runtime and excessive graphic memory consumption. To address these problems, this paper proposes a novel approach based on the RAFT framework. The proposed Attention-based Feature Localization (AFL) approach incorporates the attention mechanism to handle global feature extraction and address repetitive patterns. It introduces an operator for matching pixels with corresponding counterparts in the second frame and assigning accurate flow values. Furthermore, an Amorphous Lookup Operator (ALO) is proposed to enhance convergence speed and improve RAFT’s ability to handle large displacements by reducing data redundancy in its search operator and expanding the search space for similarity extraction. The proposed method, Efficient RAFT (Ef-RAFT), achieves significant improvements of 10% on the Sintel dataset and 5% on the KITTI dataset over RAFT. Remarkably, these enhancements are attained with a modest 33% reduction in speed and a mere 13% increase in memory usage. The code is available at: <https://github.com/n3slami/Ef-RAFT>

Index Terms—Optical Flow, Large Displacement, Repetitive Patterns, Attention Mechanism, Deep Neural Networks

I. INTRODUCTION

Optical Flow is a fundamental challenge in computer vision, focusing on determining the displacement vector for each pixel between two consecutive frames. This technique holds immense significance across downstream tasks, like visual tracking [1], video segmentation [2], and robot navigation [3]. Traditionally, the problem has been tackled using different classical computer vision methods such as correlation-based [4], block matching [5], and energy minimization-based [6] techniques. However, these approaches have proven to be computationally expensive, making them impractical for real-time applications.

In recent years, deep learning has emerged as a promising alternative to conventional approaches. Deep learning techniques have the advantage of bypassing the need to formulate optimization problems and instead train networks to directly predict the optical flow. These methods [7]–[9] have demonstrated comparable performance to the top traditional methods while significantly reducing the inference time, making them faster and more efficient. In general, neural network models take a pair of consecutive images captured by a frame-based

camera as input and generate predictions for the optical flow that effectively warps pixels from one image to the other.

RAFT [10], which stands for Recurrent All-Pairs Field Transforms, is considered one of the most successful learning-based methods for optical flow estimation. It has gained significant popularity as a simple yet robust baseline approach in the field. While this method exhibits efficient performance when evaluated on benchmark datasets, it can still encounter challenges under specific conditions. For instance, when dealing with significant displacements or untextured/repetitive patterns, there is a possibility of encountering large errors in the estimated optical flow. To enhance the performance of optical flow estimation, advanced techniques have been developed specifically for enhancing the RAFT method. These techniques encompass attention-based operations [11], [12], graph models [13], and latent cost-volume augmentation [14]. However, it is important to note that these methods typically require additional computational resources and introduce significant inference time, which limits their practical application in real-world scenarios.

To tackle the challenges posed by large displacements and repetitive patterns in the RAFT method, this paper introduces two novel mechanisms: 1) the Amorphous Lookup Operator (ALO), and 2) the Attention-based Feature Localizer (AFL). The former replaces the original lookup operator of RAFT, allowing the network to vary the distribution of the correlation queries based on the input frames, resulting in the ability to recognize larger displacements. The latter transforms the resultant features of the feature encoders in such a way that the network can differentiate and match the pixels in a poorly textured region, reducing the ambiguities that stem from these regions and thus, mitigating the resultant errors in estimation.

In summary, the main contributions of this work are as follows:

- Proposing the Amorphous Lookup Operator (ALO) as a novel method for tackling large displacement problem of the optical flow estimation methods.
- Proposing the Attention-based Feature Localizer (AFL) as a novel method for tackling repetitive patterns problem of the optical flow estimation methods.
- Validating the effectiveness of the proposed methods by conducting extensive experiments on the Sintel and KITTI datasets.

II. RELATED WORK

Traditionally, optical flow was commonly approached using energy-based methods [6], [15], [16] that employed a variational approach by defining a data term and a regularization term. To address the challenge of large displacements, improved estimation methods such as pyramid approaches [17] were introduced. Additionally, feature-based methods emerged as a solution to mitigate the large displacement problem [18]–[22]. These methods defined a matching term and utilized dynamic programming and interpolation techniques to enhance accuracy and robustness.

Significant progress has been made in the field of optical flow estimation, thanks to recent advancements in deep learning. FlowNet [7] introduced the first end-to-end network capable of tackling this task. Subsequently, a series of methods based on neural networks [8], [9], [23]–[29] have been developed, aiming to enhance performance through coarse-to-fine or iterative approaches. These methods have contributed to a notable improvement in optical flow estimation. Typically, these methods employ a correlation matrix to capture the similarities between pixels in consecutive frames. This correlation matrix takes the form of a 4D volume, which can be quite large depending on the size of the frames being processed. Due to the limited receptive field of Convolutional Neural Networks (CNNs), transformer-based approaches have recently emerged [14], [30], [31] as a solution. These transformer architectures have shown superior performance compared to CNNs. However, it is worth noting that transformer-based approaches often require a substantial amount of data and annotations to achieve their optimal performance.

RAFT [10], drawing inspiration from variational methods, incorporates recurrent gate units and correlation matrices at multiple scales. It stimulates an iterative optimization problem that improves the flow estimations through iterative refinement. Following the utilization of large correlation matrices in RAFT, numerous efforts have been made to enhance the performance of RAFT in various aspects [11]–[13], [32], [33]. Many of the mentioned works focus on leveraging the attention mechanism to improve RAFT. In particular, [32] introduces an attention-based approach to estimate the similarity between pairs of pixels. By incorporating vertical and horizontal correlations, it enhances the estimation process. Notably, [14] replaces correlation matrices with correlation memory, which consists of tokens describing pixels. Correlation values are computed using attention within this memory structure. Additionally, [33] employs the attention mechanism to address the impact of noise on correlation matrices, resulting in the extraction of more globally coherent and semantically stable features. Occlusion poses a significant challenge in optical flow estimation, and [11] addresses this issue by employing attention to generalize information to occluded areas. By utilizing attention, it aims to provide a more comprehensive understanding of the scene, even in the presence of occlusions. Repetitive patterns, on the other hand, introduce difficulties in determining corresponding pixels due to their similar feature

vectors. [12] utilizes attention to identify pixels that are highly likely to be correct correspondences, which proves beneficial in handling scenarios with large displacements. By leveraging attention, it enhances the ability to cope with repetitive patterns and improve the accuracy of optical flow estimation.

In this paper, building upon the RAFT framework, two novel methods are introduced to specifically tackle the challenges posed by large displacements and repetitive patterns. These methods, referred to as Attention-based Feature Localizer and Amorphous Lookup Operator significantly enhance the performance of RAFT. Notably, these proposed techniques achieve notable improvements while imposing a minimal burden on memory consumption and runtime.

III. PROPOSED METHOD

In this section, first the optical flow baseline, RAFT [10], is reviewed as this work is based on the RAFT architecture. Subsequently, a detailed explanation of the two proposed methods for addressing large displacements and repetitive patterns are provided.

A. Optical Flow Estimation Baseline

Given a pair of consecutive frames, I_1 and I_2 , a dense displacement field (a_1, a_2) is estimated. This field maps each pixel (u, v) in I_1 to its corresponding coordinates $(u', v') = (u + a_1(u), v + a_2(v))$ in I_2 . As illustrated in Fig1, the RAFT model consists of some main components, each of which is briefly explained below. For more information about the RAFT baseline, refer to [10].

1) *Encoders*: Two encoders with residual blocks are used in RAFT to extract features and reduce noise. As depicted in Fig 1, the feature encoder, g_θ , is employed to extract features from both frames. On the other hand, the context encoder, h_θ , is applied to I_1 to extract features for optimization and estimate the flow.

2) *Correlation Volume*: This volume aims to represent the similarities between the points in the pair of frames. Let's suppose that $F_1 = g_\theta(I_1)$ and $F_2 = g_\theta(I_2)$ are the extracted features from the feature encoder for I_1 and I_2 , respectively. The correlation volume, C , can be calculated as follows [10]:

$$C_{i,j,x,y}^l = \begin{cases} \langle F_1(i, j), F_2(x, y) \rangle & l = 0 \\ \text{AveragePooling}_{x,y}(C_{i,j,x,y}^{l-1}) & l \geq 1. \end{cases} \quad (1)$$

Where C^l denotes the correlation at the l -th level. In fact, the first level ($l = 0$) is created by calculating the dot product of each pair of points in F_1 and F_2 . The next three levels are formed by applying average pooling on the last two dimensions of the correlation volume in the former layer.

3) *Lookup Operator*: Given a current estimate of optical flow (a_1, a_2) , each pixel $p = (u, v)$ in I_1 is mapped to its estimated correspondence in I_2 : $q = (u + a_1(u), v + a_2(v))$. Following that, a local grid is defined around q :

$$\mathcal{N}(q)_r = \{q + h | h \in \mathbb{Z}^2, \|h\|_1 \leq r\} \quad (2)$$

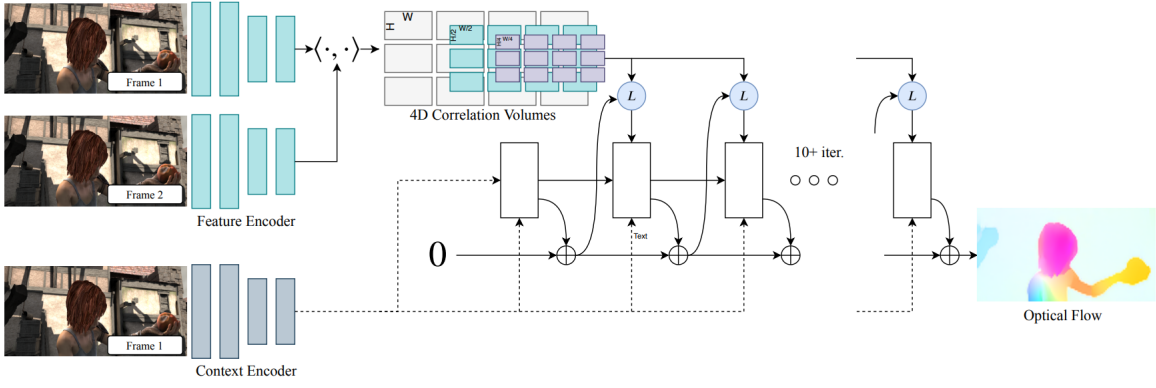


Fig. 1: Diagram of RAFT [10] encompasses three main components. 1) Feature encoders are employed to extract per-pixel features from the input frames. 2) A correlation layer constructs a correlation volume with dimensions $W \times H \times W \times H$ by computing the inner product of feature vectors for all pairs. 3) An update operator recurrently enhances the optical flow estimation by leveraging the current estimate to retrieve values from the set of correlation volumes.

where r is the radius of the grid. This operator extracts similarity values from the correlation volume as a vector for a pair of points, p and q .

4) *Recurrent Gate Units*: This component simulates the optimization algorithm, which estimates a sequence of flow estimates $\{a_1, a_2, \dots, a_N\}$ starting from an initial flow $a_0 = 0$. It takes the current flow, correlations, and a hidden state as input, and iteratively updates the flow and hidden state in output.

Finally, the loss function of the network is defined as follows:

$$\ell = \sum_{i=1}^N \gamma^{N-i} \|a_{gt} - a_i\|_1 \quad (3)$$

Where a_{gt} represents the ground truth flows, a_i denotes the calculated flows in the i -th step of the algorithm and γ is a hyperparameter.

B. Amorphous Lookup Operator

RAFT’s vanilla lookup operator has three major shortcomings: 1) Its structure imposes a hard limit of 256 pixels on the distance that the network can gather information from in each iteration. 2) In practice it often fails to find meaningful correspondences up to this range. 3) Since the lookup operator is used on all levels of the correlation volume, the data extracted is pooled from the same regions, causing unneeded redundancy. The idea of the Amorphous Lookup Operator (ALO) is to allow the network to change the lookup operator so that, based on the input frames, it can choose to query farther away similarities and reduce the redundancy of the extracted similarities.

The ALO changes the original grids of the lookup operator by scaling and translating parts of it. That is, it first calculates four scalar parameters $s_x, s_y \in [1, 3]$, $d_x, d_y \in [0, 2]$ based on the input frames. It then transforms each point $p_{old} = (i, j)$ in the old lookup operator to

$$p_{new} = (s_x i + \mathbf{sign}(i)d_x, s_y j + \mathbf{sign}(j)d_y), \quad (4)$$

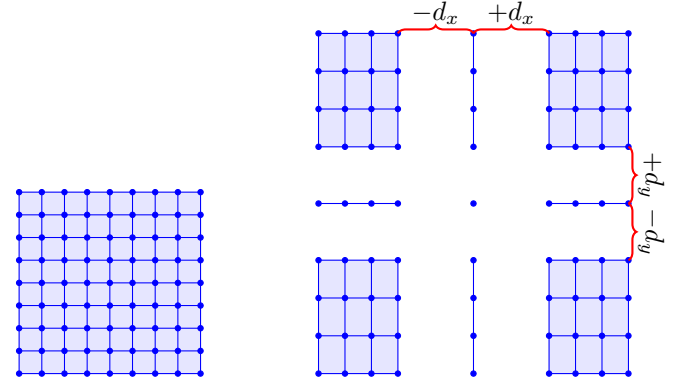


Fig. 2: Structure of the grid used in the original lookup operator (left), compared to the transformed lookup operator used in the ALO (right).

where $\mathbf{sign}(\cdot)$ is a function returning $+1$ for positive inputs, -1 for negative inputs, and 0 for an input of zero. This transformation breaks the grids into four sub-grids, which can be positioned and spaced arbitrarily, while maintaining symmetry, as shown in Fig. 2. The parameters s_x, s_y, d_x and d_y are calculated separately for each level of the correlation volume, which enable the network to adaptively pool similarity data from farther regions, while avoiding the extraction of too much redundant information.

The aforementioned scalars are calculated via the application of a light-weight network module similar in structure to the BAM [34] and CBAM [35] architectures. The structure of this module is depicted in Fig. 3. The network first concatenates the current hidden state with the context features, and enriches them with a 1×1 convolution. Then, it generates a global and compact descriptor of the algorithm’s state by doing a Max and a Min pooling operation, and concatenating the resultant vectors. It then feeds this descriptor into two single layer, fully connected networks, each with a Sigmoid

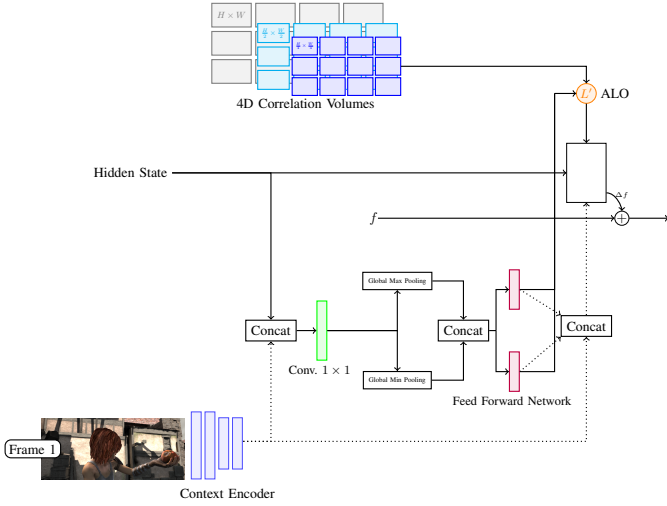


Fig. 3: Scalar parameter calculation network for the ALO.

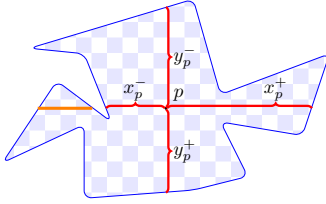


Fig. 4: Definition of the x_p^\pm and y_p^\pm values for a pixel p in a poorly textured region. The orange points depict the pixels that may cause us to err in our estimation of Δx_p and Δy_p .

activation function. These networks then apply an appropriate affine transform on the result of the Sigmoid function, outputting the scalar for all levels of the correlation volume. The final scalars are then fed into the ALO, and also concatenated with the context features of the network. Concatenation with the context feature map is crucial, as the GRU will not know what type of ALO is being used otherwise, and cannot reap the benefits of having this new lookup operator.

Therefore, the light weight of this structure allows the network to calculate the scalar parameters in a memory and compute efficient manner, adapting the lookup operator based on the input image and the current stage of the regression algorithm.

C. Attention-based Feature Localizer

The vanilla RAFT architecture uses the outputs of the feature encoders to calculate the correlation volumes. Due to the convolutional structure of these encoders, the output features are inherently local, which may cause the network as whole not being able to differentiate between points in poorly textured regions of the frames. Consequently, the regression algorithm will have a hard time estimating a correct flow value for these regions of the frames, as it has to utilize the flow values outside of the regions to slowly correct the prediction of the region itself.

To remedy this, the aforementioned features are augmented with a set of “global features” that encapsulate some data from across the entire frames, enabling the network to differentiate the poorly textures pixels by looking at these global features. To achieve this, for each pixel p of each frame, the following four parameters are defined: x_p^+ , x_p^- , y_p^+ , and y_p^- , which represent how many pixels must we move from p to the right (resp. left, up, and down) to exit the poorly textured region. An example of these values is depicted in Fig. 4.

The idea then is to not estimate the values x_p^\pm and y_p^\pm themselves, but rather estimate $\Delta x_p = x_p^+ - x_p^-$ and $\Delta y_p = y_p^+ - y_p^-$, which essentially represent the whereabouts of the point p when compared to the “middle-points” of the poorly textured region. Therefore, Δx_p and Δy_p create something similar to a coordinate system for each poorly textured shape. One can expect the values of Δx_p and Δy_p to not change by much for corresponding points, when poorly textured regions are translated only.

Suppose the values Δx_p are calculated for the pixels in row r of the first frame. To this end, the multi-head attention mechanism is applied on the features resulting from the feature encoders of this frame, which we shall denote $f_{r,i}$ for the i -th pixel of this row. Then, the keys and values for a query $q_i = f_{r,i}$ are defined as

$$k_j = f_{r,j}, v_j = pe(i - j), \quad (5)$$

where $pe(\cdot)$ is the positional encoding function defined in [36] as $pe(\cdot)_{2k} = \sin(\frac{2\pi k \cdot \cdot}{10000^{k/d}})$, $pe(\cdot)_{2k+1} = \cos(\frac{2\pi k \cdot \cdot}{10000^{k/d}})$.

Since the keys k_i and queries q_i are defined in the same manner, in the context of a poorly textured region, this mechanism will take a weighted average of the relative positional encodings of the row, with the pixels in the region having a much larger weight that the pixels outside. This will result in an estimate of Δx_p in the form of a positional encoding, which gives the network some idea as to what the actual value of Δx_p is. The implementation uses four heads during this calculation, so that it can take into account more complex similarities between the feature vectors in the pooling procedure.

Notice that v_j is a “relative positional encoding” dependent on both i and j . To calculate this relative positional encoding, first the multi-head attention mechanism is applied using $v'_j = pe(j)$, which is the standard positional encoding, and then exploit the linearity of the attention mechanism along with well-known trigonometric identities to convert the input j into $i - j$. It is worth noting that the resulting estimation is not without error. For instance, in Fig. 4, the points colored in orange will bias the estimate of Δx_p towards the left.

This same procedure can be applied on all rows and columns of both frames to derive estimates for all values Δx_p and Δy_p , all using the same module. The resulting estimates are then concatenated with the features of their corresponding frames, effectively augmenting the old features with a set of global features that span the entire row and column of each pixel. This resulting estimate lends itself well to how the correlation volume is calculated, as closer values of the input to $pe(\cdot)$ correspond to a larger dot-product, and thus more similarity.

It is worth noting that this mechanism is applied only on the rows and columns, enabling it to enjoy excellent speed and memory efficiency.

IV. EXPERIMENTS

A. Datasets and Training Schedule

Based on prior works, the model is initially pretrained on the FlyingChairs dataset [7] for 100k iterations using a batch size of 10. Following that, it undergoes an additional pretraining phase on the FlyingThings dataset [37] for 200k iterations with a batch size of 6. In this work, no fine-tuning is performed. The evaluation is conducted on both the train split of the KITTI-15 [38] dataset and the train split of the MPI Sintel [39] dataset, considering both the Clean and Final passes of the data.

For the ablation results, fast configuration was employed for training and evaluation purposes. In this configuration, the model underwent training for 25k steps using the train set of the MPI Sintel dataset. Following the training phase, the model was evaluated on the train set of the KITTI-15 dataset.

The implementations of all the networks were done using the PyTorch framework. Training and inference processes were conducted on a single NVIDIA GeForce RTX 3090 GPU. The training process involves the utilization of the AdamW optimizer with a one-cycle learning rate scheduler. Additionally, the gradients are clipped to the range of $[-1, 1]$. The hyperparameter γ in the equation 3 is set to 0.8, similar to the approach used in [10].

B. Evaluation Metrics

The main evaluation metric for the MPI Sintel dataset is the end-point error (EPE), which represents the average error of the flow estimation at each pixel, measured in terms of the number of pixels. For the KITTI-15 dataset, the evaluation metrics used are F1-EPE and F1-All. F1-EPE is the same as the EPE metric described for the Sintel dataset. On the other hand, F1-All measures the percentage of outliers, specifically pixels whose end-point error exceeds either 3 pixels or 5% of the ground truth flow magnitude. This metric is averaged over all pixels in the dataset. To validate the efficiency of the proposed method, additional metrics such as the number of parameters in the network, runtime, and memory usage are reported.

C. Experimental Results

Extensive experiments were conducted to validate the effectiveness of the proposed method by comparing it with several existing methods, including RAFT used as the baseline. The results of the proposed method in comparison to these existing methods are presented in Table I. The table demonstrates that the proposed method achieves great results in terms of EPE on the challenging Sintel dataset, particularly on the Final pass. Moreover, on the KITTI dataset and the F1-All evaluation metric, the proposed method achieves the third-best performance. It is worth noting that GMFlowNet, being a transformer-based method, requires a larger amount of training data and has a higher number of parameters compared to our

method. In light of this, the proposed method still achieves remarkable results overall. Specifically, compared to RAFT, it demonstrates more than a 10% improvement on the Sintel dataset and nearly a 6% improvement on the KITTI dataset, which means the proposed method plays a significant role in improving large errors.

Additionally, it is important to mention that CRAFT and GMA are two methods that exhibit results comparable to the proposed method. These methods have pursued independent approaches in their research and can even be combined with the proposed method to further enhance performance.

Table II presents a runtime comparison between the proposed method and RAFT, considering two different numbers of steps. The table illustrates that the proposed method improves upon RAFT while introducing only a 33% runtime overhead. This indicates a balanced trade-off between accuracy and efficiency, as the proposed method achieves better results while maintaining reasonable runtime performance.

Table III provides a comparison of RAFT, the proposed method, and GMFlowNet in terms of the number of parameters and memory usage. As mentioned earlier, the proposed method utilizes significantly less GPU memory compared to GMFlowNet, making it feasible to run on the 24GB RAM of the Nvidia GeForce 3090 GPU. Additionally, the table reveals that the proposed method introduces only a slight increase in the number of parameters and memory usage (13%) compared to RAFT, despite its significant improvement in flow estimation accuracy. This highlights the efficiency of the proposed method in achieving impressive results with relatively fewer parameters and memory requirements.

TABLE I: Comparison of the proposed method with existing techniques on the Sintel and KITTI datasets. **Green**, **blue**, and **red** colors denote the first, second, and third-best results.

Method	MPI Sintel (Train)		KITTI-15 (Train)	
	EPE (Clean)	EPE (Final)	F1-EPE	F1-All
PWC-Net [9]	2.55	3.93	10.35	33.7
LiteFlowNet [24]	2.48	4.04	10.39	28.5
LiteFlowNet2 [40]	2.24	3.78	8.97	25.9
VCN [27]	2.21	3.68	8.36	25.1
MaskFlowNet [25]	2.25	3.61	-	23.1
FlowNet2 [23]	2.02	3.54	10.08	30.0
DICL [41]	1.94	3.77	8.70	23.60
Flow1D [32]	1.98	3.27	5.59	22.95
RAFT [10]	1.43	2.71	5.02	17.46
FM-RAFT [42]	1.29	2.95	6.98	19.3
CRAFT [33]	1.27	2.79	4.88	17.50
GMA [11]	1.30	2.74	4.69	17.10
AGFlow [13]	1.31	2.69	4.82	17.0
KPA-Flow [43]	1.28	2.68	4.46	15.90
S-Flow [29]	1.30	2.59	4.60	15.90
GMFlowNet [12]	1.14	2.71	4.24	15.40
EMD-S [28]	1.31	2.67	5.0	17.0
Ef-RAFT (ours)	1.27	2.60	4.83	16.45

D. Ablation

The effectiveness of the proposed components was further validated through an ablation study. Quick training was conducted, where the lower steps method was trained on the Sintel dataset and evaluated on the KITTI dataset. The results in

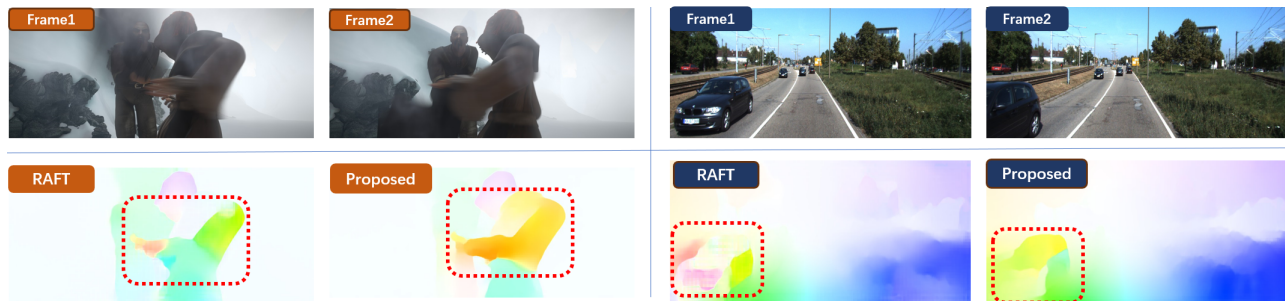


Fig. 5: Qualitative comparison between the proposed method and RAFT. Frames with orange and blue labels are from Sintel and KITTI datasets, respectively.

TABLE II: Runtime comparison between the proposed method and RAFT.

Method	#Steps	Runtime (s)	EPE (Clean)	EPE (Final)
RAFT	32	87.81	1.46	2.69
Proposed	32	117.64	1.29	2.62
RAFT	22	60.39	1.48	2.69
Proposed	22	85.91	1.28	2.60

TABLE III: Number of parameters and memory usage comparison.

Method	#Steps	#Parameters	Memory Used
RAFT	12	5.3 M	11988 (MiB)
GMFlowNet	12	9.3 M	≥ 24000 (MiB)
RAFT	12	5.7 M	13520 (MiB)

Table IV show that both ALO and AFL components contribute to the improvement over RAFT. While the AFL alone achieves better results on the KITTI dataset, the combined inclusion of both components is believed to offer greater learning ability. Overall, the study demonstrates that the synergistic effect of incorporating both components leads to enhanced performance compared to individual components.

TABLE IV: Ablation for Ef-RAFT to validate the effectiveness of proposed ALO and AFL components.

Method	MPI Sintel (Train)		KITTI-15 (Train)	
	EPE (Clean)	EPE (Final)	F1-EPE	F1-All
RAFT [10]	1.73	2.40	8.14	22.60
w/o AFL	1.57	2.19	6.40	18.81
w/o ALO	1.81	2.41	3.15	11.36
Proposed	1.57	2.13	6.02	18.75

E. Qualitative Assessment

Qualitative assessment was conducted to further validate the proposed method. Figure 5 demonstrates that the proposed method is capable of estimating flows that are significantly better than RAFT. The visual comparison clearly showcases the improved accuracy and quality of the flow estimations achieved by the proposed method.

V. CONCLUSION AND FUTURE WORK

In this paper, Ef-RAFT was presented, a re-imagining of the renowned RAFT network. The novel ideas of Amorphous Lookup Operator and Attention-based Feature Localizer were explored, which enabled Ef-RAFT to improve upon its predecessor while also keeping the computational complexity and memory footprint low. Then experiments were conducted to showcase the improved accuracy and efficiency of Ef-RAFT, as well as demonstrating the necessity of the structures used with an ablation study. It is believed that Ef-RAFT can be extended in several directions: 1) The Attention-Based Feature Localizer is not robust to rotations in the poorly textured regions. Applying the same mechanism on rotated lines in the images, instead of the rows and columns alone, may be a promising first step forward. 2) Ef-RAFT's ideas are orthogonal to what is used in many other papers, e.g. CRAFT and GMA. Combining these methods in an efficient and practical manner may also open doors to better accuracies with a light-weight network.

REFERENCES

- [1] M. Vihlman and A. Visala, "Optical flow in deep visual tracking," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 12 112–12 119.
- [2] C. Yang, H. Lamdouar, E. Lu, A. Zisserman, and W. Xie, "Self-supervised video object segmentation by motion grouping," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 7177–7188.
- [3] G. C. de Croon, C. De Wagter, and T. Seidl, "Enhancing optical-flow-based control by learning visual appearance cues for flying robots," *Nature Machine Intelligence*, vol. 3, no. 1, pp. 33–41, 2021.
- [4] A. Singh, *Optic flow computation: a unified perspective*. IEEE computer society press Los Alamitos, 1991, vol. 3.
- [5] S. S. Beauchemin and J. L. Barron, "The computation of optical flow," *ACM computing surveys (CSUR)*, vol. 27, no. 3, pp. 433–466, 1995.
- [6] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.
- [7] 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.
- [8] 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.
- [9] 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.

- [10] 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.
- [11] 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.
- [12] 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.
- [13] A. Luo, F. Yang, K. Luo, X. Li, H. Fan, and S. Liu, "Learning optical flow with adaptive graph reasoning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 36, no. 2, 2022, pp. 1890–1898.
- [14] 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.
- [15] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime tv-l 1 optical flow," in *Pattern Recognition: 29th DAGM Symposium, Heidelberg, Germany, September 12-14, 2007. Proceedings 29*. Springer, 2007, pp. 214–223.
- [16] M. J. Black and P. Anandan, "A framework for the robust estimation of optical flow," in *1993 (4th) International Conference on Computer Vision*. IEEE, 1993, pp. 231–236.
- [17] T. Brox, A. Bruhn, N. Papenberger, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *Computer Vision—ECCV 2004: 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part IV 8*. Springer, 2004, pp. 25–36.
- [18] C. Bailer, B. Taetz, and D. Stricker, "Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4015–4023.
- [19] M. Menze, C. Heipke, and A. Geiger, "Discrete optimization for optical flow," in *Pattern Recognition: 37th German Conference, GCPR 2015, Aachen, Germany, October 7-10, 2015, Proceedings 37*. Springer, 2015, pp. 16–28.
- [20] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "Deepflow: Large displacement optical flow with deep matching," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 1385–1392.
- [21] T. Brox and J. Malik, "Large displacement optical flow: descriptor matching in variational motion estimation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 3, pp. 500–513, 2010.
- [22] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "Epicflow: Edge-preserving interpolation of correspondences for optical flow," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1164–1172.
- [23] 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.
- [24] 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.
- [25] S. Zhao, Y. Sheng, Y. Dong, E. I. Chang, Y. Xu *et al.*, "Maskflownet: Asymmetric feature matching with learnable occlusion mask," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 6278–6287.
- [26] J. Hur and S. Roth, "Iterative residual refinement for joint optical flow and occlusion estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5754–5763.
- [27] G. Yang and D. Ramanan, "Volumetric correspondence networks for optical flow," *Advances in neural information processing systems*, vol. 32, 2019.
- [28] C. Deng, A. Luo, H. Huang, S. Ma, J. Liu, and S. Liu, "Explicit motion disentangling for efficient optical flow estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 9521–9530.
- [29] F. Zhang, O. J. Woodford, V. A. Prisacariu, and P. H. Torr, "Separable flow: Learning motion cost volumes for optical flow estimation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 807–10 817.
- [30] Y. Lu, Q. Wang, S. Ma, T. Geng, Y. V. Chen, H. Chen, and D. Liu, "Transflow: Transformer as flow learner," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 18 063–18 073.
- [31] 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.
- [32] H. Xu, J. Yang, J. Cai, J. Zhang, and X. Tong, "High-resolution optical flow from 1d attention and correlation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 498–10 507.
- [33] 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.
- [34] J. Park, S. Woo, J.-Y. Lee, and I. S. Kweon, "Bam: Bottleneck attention module," *arXiv preprint arXiv:1807.06514*, 2018.
- [35] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.
- [36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
- [37] 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.
- [38] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.
- [39] 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.
- [40] T.-W. Hui, X. Tang, and C. C. Loy, "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.
- [41] J. Wang, Y. Zhong, Y. Dai, K. Zhang, P. Ji, and H. Li, "Displacement-invariant matching cost learning for accurate optical flow estimation," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15 220–15 231, 2020.
- [42] S. Jiang, Y. Lu, H. Li, and R. Hartley, "Learning optical flow from a few matches," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 16 592–16 600.
- [43] A. Luo, F. Yang, X. Li, and S. Liu, "Learning optical flow with kernel patch attention," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8906–8915.