# Efficient Motion Feature Aggregation for Optical Flow via Locality-Sensitive Hashing

Weihai Chen[a], Ziyang Liu[a], Xingming Wu[a,1], Zhong Liu[a], Zhengguo Li[b]

[a] *School of Automation Science and Electrical Engineering, Beihang University, Beijing, 100191, Beijing, China*
[b] *VI Department, Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), 138632, Singapore, Republic of Singapore*

**Abstract**

Optical flow estimation aims to find the 2D motion field by identifying the corresponding pixels between two images. With the tremendous progress of deep learning, previous works rely on CNNs to regress the pixelwise correspondence. However, it is still challenging to estimate displacements at occlusions, where one point is imaged in the current frame but not in the next frame. To solve this problem, recent work proposes to aggregate motion features at a global scale. The aggregation allows motion information to be passed from non-occluded pixels to occluded pixels, which helps resolve ambiguities caused by occlusions. Though the global motion aggregation (GMA) works well, its computational complexity is quadratic to the input resolution. The complexity can be a limitation of GMA when applied to higher resolution. In this paper, we propose an efficient motion feature aggregation (EMFA) module, to approximate the original GMA at a much lower computational cost. The core insight of EMFA originates from a clustering scheme, where the motion information of one point is aggregated from the points in the same cluster. The clustering is implemented by a traditional hashing algorithm, called locality-sensitive hashing (LSH). Finally, we apply the EMFA module to the optical flow estimation at high resolution. Experimental results on publicly available datasets show that our method outperforms the state-of-the-art ones.

[1]Corresponding author: Xingming Wu, wxmbuaa@163.com.

[2]This work is supported by National Natural Science Foundation of China under Grant No. 61620106012.

*Keywords:*
Optical flow, efficient motion feature aggregation, clustering,
locality-sensitive hashing.

## 1. Introduction

Optical flow records pixel-wise displacements (or correspondences) between two images [1]. In many downstream video tasks, such as visual odometry, object tracking, semantic segmentation, and video inpainting, optical flow serves as a fundamental component providing dense correspondences as valuable clues for prediction [2, 3].

Recently, most optical flow methods are deep learning-based. One of the most significant advances in the development of this field was the introduction of the FlowNet [4]. The features and motion priors are learned by the CNNs rather than handcrafted in the traditional approaches. The pixel-wise correspondence is identified by building the cost volume, which computes the dot product between the learned deep feature representations of images. Subsequent works take the inspiration from the FlowNet and they estimate the optical flow iteratively, from coarse to fine, from large displacements to small displacements, such as the LiteFlowNet [5], PWC-Net [6], and FlowNet2 [7].

The RAFT [1] significantly improved the performance of deep learning-based optical flow methods. It builds the cost volume by computing similarities between full-resolution feature maps instead of warped feature maps. The warping operation is avoided since it brings ambiguities to the cost volume. The optical flow is decoded from the motion features via a convolutional gated recurrent unit (GRU) iteratively. The motion features are obtained by performing lookups on the cost volume. Recent outstanding optical flow methods are all based on the RAFT architecture, including the SCV [8], GMA [9], MS-RAFT [10], GMFlowNet [11], and CRAFT [12].

Despite the progress made by previous works, the optical flow estimation is still challenged by the occlusion problem. An occluded point refers to the one that is imaged in the current frame but not in the next. These points are either occluded by foreground moving objects or moving out of the frame. The most effective solution to the occlusion problem is the GMA [9]. It performs global motion aggregation (GMA) on the motion features. The motion information of an occluded point can be refined by aggregating information from non-occluded points, as shown in Figure 1(a). However,

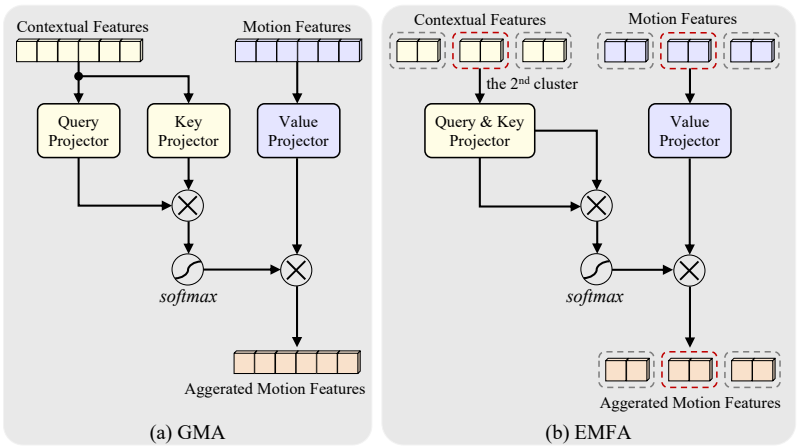



Figure 1: The diagram of the original GMA vs. the proposed EMFA. The clustered vectors in the EMFA are marked by dash lines.

the GMA is a variant of the non-local attention [13] and its computational complexity is quadratic to the input resolution. This can a limitation for optical flow estimation at higher resolution.

In this paper, we propose a more efficient way to aggregate the motion features. The idea comes from the observation that one pixel is mainly affected by the pixels that are highly correlated to it. By the *softmax*-based weighted sum in attention, pixels with similar appearances tend to take larger weights. For the pixels that have a large dissimilarity to the target pixel, they will be less informative. Therefore, the complexity of original GMA can be simplified by aggregating motion information from a few correlated pixels instead of all pixels.

How to determine the correlated pixels remains unsolved. We propose to adopt a clustering scheme, where the GMA is performed within each cluster. Considering the efficiency and accuracy, we use a traditional hashing algorithm to sort the pixels inspired by [14]. Each individual pixel will be assigned an hash code, and the pixels with the same hash code fall into the same cluster. The diagram of the efficient motion feature aggregation (EMFA) is shown in Figure 1(b). The aggregation can be computed in parallel.

Since the hashing algorithm is efficient, the overall complexity of EMFA is greatly reduced compared to the GMA. Benefiting from the well clustering, the EMFA only sacrifices a little on accuracy. The EMFA also differs from the SCV [8]. To cluster the correlated elements, a traditional hashing method is

adopted instead of the $k$-nearest neighbours algorithm. The hashing method is more simple and effective. Furthermore, we land the EMFA module to the MS-RAFT [10] architecture to estimate optical flow at higher resolution, such that the occlusions of small objects can be well handled. In summary, our main contributions are:

- A novel EMFA module is proposed to approximate the original GMA, targeting at the occlusion problem in optical flow estimation. The EMFA is much more efficient than the GMA.
- We land the EMFA module to the RAFT architecture and extend it to estimate optical flow at higher resolution. The benefit is that the occlusions of small objects can be well handled.
- Our method achieves the best optical flow results compared to the state-of-the-art deep learning methods on publicly available benchmarks.

## 2. Related work

Optical flow has traditionally been treated as an energy minimization problem. A novel data-based search strategy was proposed in [15, 16], which finds most inliers while effectively avoids finding outliers for optimization. In the DCFlow, a full four-dimensional cost volume was presented, inspired from stereo matching pipelines [17]. The MRFlow proposes to jointly estimate the camera motion and the 3D structure of the scene over multiple frames [18].

Recently, deep neural networks have been adopted to predict the optical flow directly [4]. However, the computation of the match density for each pixel can be expensive. In the HD3, the full match density is decomposed into multiple scales hierarchically, and the optical flow is estimated from the local matching distributions at each scale conditioned on the matching and warping at coarser scales [19]. Such a coarse-to-fine estimation pipeline has been popularly used in CNN-based optical flow methods [7, 5, 6, 20]. The VCN introduced several simple modifications that improves the optical flow estimation, including the volumetric encoder-decoder architecture, the multi-channel cost volumes, and the separable volumetric filtering algorithm [21]. An asymmetric occlusion-aware feature matching module was proposed in [22], which can learn a rough occlusion mask that filters useless (occluded) areas immediately after feature warping without any explicit supervision.

The RAFT significantly improves the performance [1]. It builds the cost volume between two full-resolution feature maps instead of warped feature maps. The motion features are iteratively taken from the cost volume and then decoded into optical flows. Currently, most outstanding optical flow methods are based on the RAFT architecture. The MS-RAFT is an enhanced version of the RAFT, which builds multi-scale cost volumes from multi-scale image features [10]. The GMA module is proposed to aggregate motion features from similar pixels and it helps resolve the occlusion problem [9]. Following the GMA, the GMFlow performs the global motion aggregation directly on optical flows, named flow propagation [23]. In the GMFlowNet, the global matching is efficiently calculated by performing a dual-softmax operator on the 4D cost volume [11]. In the SCV, the $k$-nearest neighbours algorithm is adopted to select the most correlated pixels and the optical flow is estimated only by these few matches [8]. While the CRAFT [12], Flowformer [24], and Flowformer++ [25] introduce the Transformer architecture for optical flow estimation. The Anyflow represents the optical flow as a continuous coordinate-based representation and estimates accurate flow from images of various resolutions [26]. A new data-augmentation scheme, which introduces realistic distractions to the input frames, was proposed for training optical flow estimation models in [27]. The OCAI jointly performs the frame interpolation and the optical flow estimation, which employs occlusion awareness to resolve ambiguities in pixel values [28].

## 3. Method

In this section, we first briefly review the original GMA module [9]. Then, the novel EMFA module is proposed, which approximates the GMA at a much lower computational cost. Finally, we land the EMFA module to the RAFT architecture [1] and perform the motion feature aggregation at higher resolution benefiting from the EMFA's high efficiency.

### *3.1. Global Motion Aggregation*

Following [9], we let $\boldsymbol{x} \in \mathbb{R}^{N_s \times D_c}$ denote the contextual features and $\boldsymbol{y} \in \mathbb{R}^{N_s \times D_m}$ denote the motion features, where $N_s = H_s \times W_s$ and $H_s$ and $W_s$ are the height and width of the feature map at scale $s$. The $D_c$ and $D_m$ refer to the channel dimensions of $\boldsymbol{x}$ and $\boldsymbol{y}$. The $i^{\text{th}}$ feature vector of $\boldsymbol{x}$ is

denoted as $\boldsymbol{x}_i \in \mathbb{R}^{D_c}$. Formally, the GMA module is given by:

$$\widehat{\boldsymbol{y}}_i = \boldsymbol{y}_i + \alpha \sum_{j=1}^{N_s} f(\boldsymbol{W}_q \boldsymbol{x}_i, \boldsymbol{W}_k \boldsymbol{x}_j) \boldsymbol{W}_v \boldsymbol{y}_j \tag{1}$$

where $\alpha$ is a learnable scalar parameter. The $\boldsymbol{W}_q, \boldsymbol{W}_k \in \mathbb{R}^{D_c \times D_c}$, and $\boldsymbol{W}_v \in \mathbb{R}^{D_m \times D_m}$ are the projection weights for the query, key, and value, respectively. The $f$ denotes the dot-product attention that is given by:

$$f(\boldsymbol{a}_i, \boldsymbol{b}_j) = \frac{\exp({\boldsymbol{a}_i}^{\mathrm{T}} \boldsymbol{b}_j / D)}{\sum_{j=1}^{N} \exp({\boldsymbol{a}_i}^{\mathrm{T}} \boldsymbol{b}_j / D)}. \tag{2}$$

We don't use the 2D relative positional embedding since its improvement is negligible according to [9].

The GMA module globally aggregates motion features of similar pixels, which is motivated by the observation that points on an object often have homogeneous motions. It helps resolve the ambiguities caused by occlusions, where the motion information from the non-occluded pixels (high confidence) can be propagated to the occluded pixels (low confidence).

### *3.2. Efficient Motion Feature Aggregation*

For dense prediction tasks, model performance tends to improve with higher input resolution [29]. However, the GMA is a variant of the self-attention in [13] and its computational complexity is quadratic to the resolution of input features. The complexity can be a limitation of GMA when applied to higher resolution, where the computation becomes infeasible given contextual and motion features at larger scales.

To address the limitation, we propose the EMFA to aggregate motion features with a lower complexity. Our core insight is to let the given query vector attend to some instead of all key vectors in the attention. A general form of the EMFA can be simply formalized as:

$$\widehat{\boldsymbol{y}}_i = \boldsymbol{y}_i + \alpha \sum_{j \in \Omega_i} f(\boldsymbol{W}_q \boldsymbol{x}_i, \boldsymbol{W}_k \boldsymbol{x}_j) \boldsymbol{W}_v \boldsymbol{y}_j, \tag{3}$$

where $\Omega_i$ denotes the set that contains the indexes of key vectors attended by the query vector $\boldsymbol{W}_q \boldsymbol{x}_i$. If we fix the cardinality of $\Omega_i$, the complexity of EMFA can be reduced to linear. Notably, the Equation (3) is equivalent

to Equation (1) when the indexes of all key vectors are included by the $\Omega_i$. Thus, the proposed EMFA is a generalized variant of the original GMA.

As indicated by the Equations (1) and (2), the key vectors that are more correlated to the query vector dominant the GMA, while the less correlated ones play marginal roles. Therefore, the $\Omega_i$ is expected to include indexes of vectors that are highly correlated to the query, such that the EMFA is able to perform the global motion aggregation efficiently meanwhile sacrifices little on accuracy.

### *3.3. Locality-Sensitive Hashing*

To determine the $\Omega$ for each query vector, one can use the $k$-nearest neighbours ($k$NN) algorithm to select $k$ key vectors with the largest correlation scores as in [8]. While in this work, we propose a clustering method to select the correlated vectors, which is more efficient. Before clustering, we should keep $\boldsymbol{W}_q = \boldsymbol{W}_k$ in Equation (3) such that the query features equal to the key features. Then, we can sort the feature vectors according to their correlations, where the correlated ones are put into the same cluster. Finally, each $\Omega$ is determined by the cluster and the attention is performed within it.

To implement the clustering, we take inspiration from a traditional hashing scheme, called locality-sensitive hashing (LSH) [14]. Specifically, each query/key vector will be assigned a hash code $\mathcal{H}(\boldsymbol{W}_q\boldsymbol{x}_i)$ via a pre-defined hash function $\mathcal{H}$. The vectors that have the same hash code are sorted into the same cluster, i.e.:

$$\Omega_i = \{j|\mathcal{H}(\boldsymbol{W}_q\boldsymbol{x}_i) = \mathcal{H}(\boldsymbol{W}_k\boldsymbol{x}_j)\}. \tag{4}$$

The $\Omega_i$ is also referred to as the hash bucket. To ensure that the correlated vectors will be assigned the same hash code, the hash function is defined as:

$$\mathcal{H}(\boldsymbol{W}_q\boldsymbol{x}_i) = \arg\max_h \boldsymbol{R}\boldsymbol{W}_q\boldsymbol{x}_i[h], \tag{5}$$

where $h \in [1, N_h]$ and $\boldsymbol{R}\boldsymbol{W}_q\boldsymbol{x}_i[h]$ denotes the $h^{\text{th}}$ element of the vector $\boldsymbol{R}\boldsymbol{W}_q\boldsymbol{x}_i$. The $\boldsymbol{R} \in \mathbb{R}^{N_h \times D_c}$ denotes a fixed matrix that is randomly generated and $N_h$ is the number of hash buckets (i.e. clusters).

The matrix $\boldsymbol{R}$ can be viewed as a combination of $N_h$ vectors of size $D_c$, where the $h^{\text{th}}$ vector can be denoted as $\boldsymbol{R}_h \in \mathbb{R}^{D_c}$. According to [13], the dot-product of two vectors is big when they are correlated and small if not. Thus, the objective of Equation (5) is to find the most correlated vector to

$\boldsymbol{W}_q\boldsymbol{x}_i$ in $\boldsymbol{R}$. We denote this vector as $\boldsymbol{R}_{\hat{h}}$, where $\hat{h} = \mathcal{H}(\boldsymbol{W}_q\boldsymbol{x}_i)$ represents the hash code. If the query vector $\boldsymbol{W}_q\boldsymbol{x}_i$ and the key vector $\boldsymbol{W}_k\boldsymbol{x}_j$ have the same hash code, e.g. the $\hat{h}$, they are both very close to the vector $\boldsymbol{R}_{\hat{h}}$, which also implies that the $\boldsymbol{W}_q\boldsymbol{x}_i$ and the $\boldsymbol{W}_k\boldsymbol{x}_j$ have a close distance and they are correlated to each other. Therefore, the correlated vectors are more likely to be put into the same hash bucket (cluster) while the uncorrelated ones are not via Equations (4) and (5). The vector $\boldsymbol{R}_h$ can be viewed as the centroid of the $h^{\text{th}}$ cluster.

A proper matrix $\boldsymbol{R}$ is required by the clustering. The distribution of $\boldsymbol{R}$ is expected to be as close as possible to that of the input features [30]. Since the deep features are subjected to the Gaussian distribution [31, 32], the $\boldsymbol{R}$ is randomly sampled by:

$$\boldsymbol{R} \sim \mathcal{N}(\text{mean}(\boldsymbol{x}\boldsymbol{W}_q),\ \text{std}(\boldsymbol{x}\boldsymbol{W}_q)^2), \tag{6}$$

where $\mathcal{N}$ denotes the Gaussian distribution. The $\text{mean}(\boldsymbol{x}\boldsymbol{W}_q)$ and $\text{std}(\boldsymbol{x}\boldsymbol{W}_q)$ denote the mean and standard deviation of the query features $\boldsymbol{x}\boldsymbol{W}_q$, respectively.

### *3.4. Robust batch computation*

By taking the LSH scheme, all the query vectors are sorted into different hash buckets and the motion feature aggregation can be performed within each bucket. However, the size of each bucket tends to be different. The batch computation (i.e. the parallel computation) of buckets is not feasible and the computation is still inefficient. To overcome this problem, we force the uneven buckets into even chunks. Similar to the hash bucket $\Omega_i$, the chunk attended by the $i^{\text{th}}$ query vector $\boldsymbol{W}_q\boldsymbol{x}_i$ is denoted by the $\Phi_i$. There are totally $N_h$ chunks and the size of each chunk is fixed to $N_\Phi = \lceil \frac{N_s}{N_h} \rceil$.

To make the indexes from the same hash bucket fall into the same chunk, we sort the query vectors according to the ascending order of their hash codes (from 1 to $N_h$). For the vectors with the same hash code, we sort them according to their spatial locations, i.e. their indexes. We use a function $\mathcal{P}$ to define the sorting operation, which maps the original index $i$ into a new index $p_i$. The $p_i$ is given by $p_i = \mathcal{P}(\mathcal{H}(\boldsymbol{W}_q\boldsymbol{x}_i);\ i)$. Then, the sorted vectors are evenly split into chunks, such that the attention within chunks can be computed in batch. The whole process is shown in Figure 2. The $\Phi_i$ is given by:

$$\Phi_i = \{j | \lfloor \frac{p_i}{N_\Phi} \rfloor N_\Phi \leq j \leq \lceil \frac{p_i}{N_\Phi} \rceil N_\Phi\}. \tag{7}$$

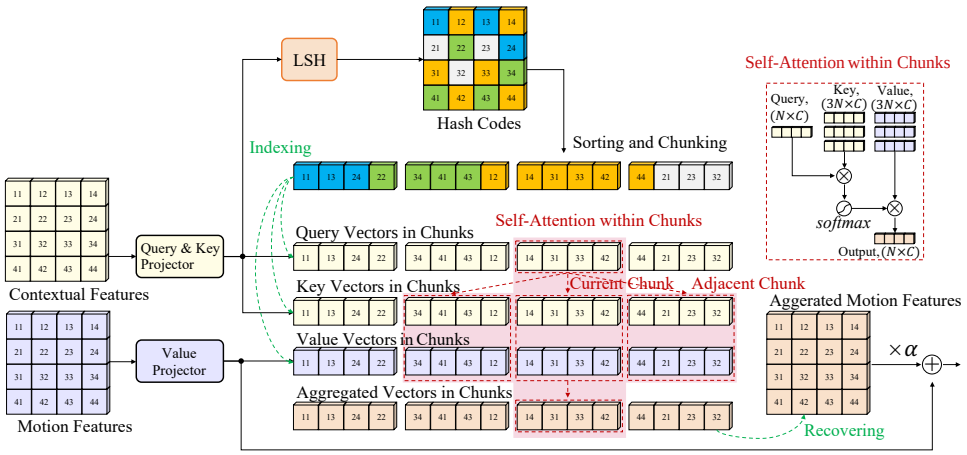



Figure 2: The diagram of the proposed EMFA module. The contextual features are first projected into query and key features, which are then clustered via the LSH algorithm. The number inside each square denotes the spatial location of the vector. The color of square (i.e. black, green, orange, and gray) denotes the hash code of the vector. The query, key, and value vectors are sorted and split into chunks according to their hash codes and spatial locations, as described in Section 3.4. Finally, the self-attention is performed within chunks, where the query vector aggregates motion information from only a few correlated vectors. The output vectors are recovered back to the original locations to obtain the final aggregated motion features.

Notably, the hash buckets and the chunks are not the one-to-one correspondence. For the indexes from the $h^{\text{th}}$ bucket, most of them will fall into the $h^{\text{th}}$ chunk, while the rest may scatter into adjacent chunks, i.e. the $(h-1)^{\text{th}}$, $(h+1)^{\text{th}}$, $(h-2)^{\text{th}}$, $(h+2)^{\text{th}}$ chunks and so on. Thus, the $\Phi_i$ has to be extended to adjacent chunks to cover the scattered indexes that belong to the $\Omega_i$. For a trade-off between accuracy and efficiency, one chunk back and one chunk before are included by the new $\Phi_i$:

$$\begin{aligned}\Phi_i = \{j | \max(\lfloor \frac{p_i}{N_\Phi} \rfloor N_\Phi - N_\Phi, 1) \leq j \\ \leq \min(\lceil \frac{p_i}{N_\Phi} \rceil N_\Phi + N_\Phi, N_s)\}.\end{aligned} \tag{8}$$

The robust EMFA is also challenged by improper clustering, where the correlated vectors sometimes fall into different hash buckets (4). To alleviate this, we propose to cluster several times, with each time using a different matrix $\boldsymbol{R}$. The $u^{\text{th}}$ randomly generated matrix is denoted as $\boldsymbol{R}_u$. The $\Omega_i$ and $\Phi_i$ determined by the $\boldsymbol{R}_u$ are denoted as $\Omega_i^u$ and $\Phi_i^u$, respectively. Each

$\boldsymbol{R}_u$ corresponds to an attention output. Similar to the multi-head attention in [13], the multiple outputs can be aggregated via learnable linear weights. While in this work, we aggregate the outputs through a weighted-sum formulation, which is more explainable [12]. The final EMFA of a motion vector $\boldsymbol{y}_i$ is given by:

$$\boldsymbol{y}_i^u = \sum_{j \in \Phi_i^u} f(\boldsymbol{W}_q \boldsymbol{x}_i, \boldsymbol{W}_k \boldsymbol{x}_j) \boldsymbol{W}_v \boldsymbol{y}_j, \tag{9}$$

$$\boldsymbol{f}_i^u = \sum_{j \in \Phi_i^u} f(\boldsymbol{W}_q \boldsymbol{x}_i, \boldsymbol{W}_k \boldsymbol{x}_j), \tag{10}$$

$$\widehat{\boldsymbol{y}}_i = \boldsymbol{y}_i + \alpha \sum_{u=1}^{N_r} \frac{\boldsymbol{f}_i^u}{\sum_{v=1}^{N_r} \boldsymbol{f}_i^v} \boldsymbol{y}_i^u, \tag{11}$$

where $N_r$ denotes the number of generated matrices $\boldsymbol{R}$. Compared to the original GMA in Equation (1), the EMFA reduces the complexity by allowing the query vector to aggregate motion information from only a few correlated vectors. The Equation (11) can be simply extended to all the $N_s$ motion vectors for batch computation.

### *3.5. Network and Loss function*

We land the EMFA module to the RAFT architecture [1]. Following [9], the aggregated motion features are concatenated with the original motion features. Similar to [10], the optical flows are predicted at multiple scales, including the $\frac{1}{16}$, $\frac{1}{8}$, and $\frac{1}{4}$ of the original resolution. The predictions start from the coarsest scale, and end at the finest. The EMFA module is inserted at each scale and it is feasible for high-resolution predictions benefiting from its high efficiency. The overall network architecture, denoted as the EMFANet, is illustrated in Figure 3.

The loss function is formalized by the L1 distance between the predicted and ground truth optical flows:

$$\mathcal{L} = \sum_{i=1}^{N_{\text{scales}}} \sum_{j=1}^{N_{\text{iters}}(i)} \gamma_{i,j} ||\boldsymbol{o}_{\text{gt}} - \boldsymbol{o}_{i,j}||_1, \tag{12}$$

$$\gamma_{i,j} = \gamma^{(\sum_{u=1}^{N_{\text{scales}}} N_{\text{iters}}(u)) - (\sum_{v=1}^{i-1} N_{\text{iters}}(v)) + j)}, \tag{13}$$

where $\boldsymbol{o}_{i,j}$ denotes the optical flow predicted at the $j^{\text{th}}$ iteration of the $i^{\text{th}}$ scale and $\gamma_{i,j}$ corresponds to the penalty weight. The $\boldsymbol{o}_{\text{gt}}$ denotes the ground

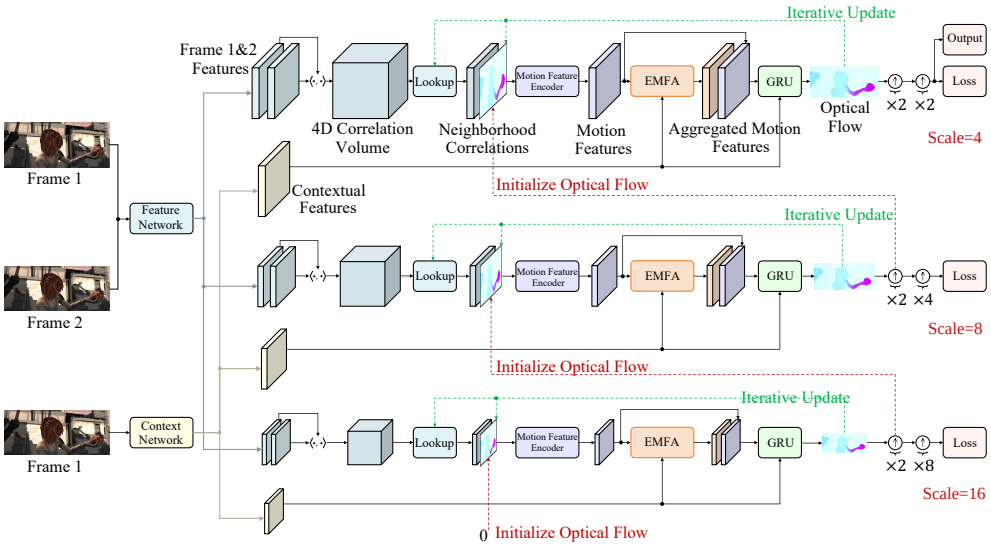



Figure 3: The overall network architecture of the EMFANet. The EMFANet estimates optical flows at three scales, each of them having a similar architecture to the RAFT [1]. The EMFA module is inserted at each scale to enhance the motion features. The texts in brackets denote the corresponding spatial resolutions. Notably, there are 3 scales in the network, where $s \in \{16, 8, 4\}$.

truth optical flow. The $N_{\text{scales}}$ and $N_{\text{iters}}(i)$ denote the number of scales and the number of iterations at the $i^{\text{th}}$ scale, respectively.

## 4. Experiments

### *4.1. Experimental setup*

We use Python and Pytorch to implement the method. Experiments are conducted on the 4029GP-TRT server carried with NVIDIA TITAN RTX GPU of 24GB memories.

#### *4.1.1. Network details*

The feature and context encoders of the EMFANet are the U-Net-style networks, where the number of feature channels from the coarsest scale to the finest scale are 256, 128, and 96 [10]. The number of GRU iterations from the coarsest to the finest are 4, 6, and 8 during training, and 10, 15, and 20 during test. For the 4D correlation volume, the number of pyramids is set as 2 and the lookup radius at each pyramid is set as 4 [1]. The chunk

size $N_\Phi$ at each scale is set as 128. The corresponding number of chunks is $N_h = \lceil \frac{N_s}{N_\Phi} \rceil$ and the feature map is padded from the size of $N_s$ to $N_\Phi N_h$. The number of clustering times $N_r$ is set as 4.

*4.1.2. Training details*

The training scheme is similar to [1]. The EMFANet is first pre-trained on the FlyingChairs and the FlyingThings3D [33] successively, each by 100k iterations. Then, it is fine-tuned on a mixed dataset of Sintel [34], KITTI-15 [35], HD1K [36], and FlyingThings3D with the same proportions suggested by [1] for another 100k iterations. Finally, the EMFANet is fine-tuned on the KITTI-15 and the HD1K for additionally 50k iterations. The model weights obtained from the current training stage are passed to the next one for initialization. The batch size is set as 10, 6, 6, and 6 on the four training stages. The initial learning rate is set as 4e-4, 1.25e-4, 1.25e-4, and 1e-4, respectively. The random cropping, flipping, resizing, and color jitter are used for data augmentation. The optimizer is AdamW and the gradients are clipped to the range $[-1, 1]$. The learning rate scheduler is One Cycle. The $\gamma$ in Equation (13) is set as 0.8.

*4.2. Comparison with the state-of-the-art methods*

We compare the EMFANet with the state-of-the-art opical flow methods on the well-known Sintel and KITTI-15 benchmarks, as shown in Table 1. The results of other methods are directly quoted from their papers. For the Sintel benchmark, the average end-point-error (EPE) between predicted and ground truth optical flows on the clean pass and the final pass are reported. For the KITTI-15 benchmark, the Fl-epe and the Fl-all are reported.

*4.2.1. Generalization performance*

We train the EMFANet on the FlyingChairs and the FlyingThings3D (C+T), and evaluate it on the training sets of Sintel and KITTI-15. The EMFANet achieves the generalization performance on par with the GM-FlowNet [11], and surpasses all the other methods. Compared to the GMA [9], the EMFANet reduces 8.5% and 8.0% errors on the clean and final passes of Sintel (train), respectively. This demonstrates that aggregating motion information at higher resolution helps the model resolve the occlusion problem at a finer scale.

Table 1: The comparison with the state-of-the-art optical flow methods on the Sintel and KITTI-15 benchmarks. The C, T, S, K, and H denote the training sets of FlyingChairs, FlyingThings3D, Sintel, KITTI-15, and HD1K, respectively. The generalization performances on Sintel (train) and KITTI-15 (train) after training on FlyingChairs and FlyingThings3D are reported in the middle section. The bottom section shows the results on Sintel (test) and KITTI-15 (test) after dataset specific fine-tuning. For Sintel evaluation, the pre-trained model is fine-tuned on T+S+K+H. For KITTI-15 evaluation, the Sintel model is additionally fine-tuned on K+H. Notably, when evaluating on Sintel (test), the warm-start strategy [1] is adopted in RAFT, SCV, GMA, MS-RAFT, CRAFT, and EMFANet. Among each section, the best and the second best results are **bolded** and underlined, respectively.

| Training data | Method | Sintel (train) | | KITTI-15 (train) | | Sintel (test) | | KITTI-15 (test) |
|---|---|---|---|---|---|---|---|---|
| | | Clean | Final | Fl-epe | Fl-all | Clean | Final | Fl-all |
| - | FlowFields (2015) [15] | - | - | - | - | 3.75 | 5.81 | 15.31 |
| - | FlowFields++ (2018) [16] | - | - | - | - | 2.94 | 5.49 | 14.82 |
| S | DCFlow (2017) [17] | - | - | - | - | 3.54 | 5.12 | 14.86 |
| S | MRFlow (2017) [18] | - | - | - | - | 2.53 | 5.38 | 12.19 |
| C+T | HD3 (2019) [19] | 3.84 | 8.77 | 13.17 | 24.0 | - | - | - |
| | LiteFlowNet (2018) [5] | 2.48 | 4.04 | 10.39 | 28.5 | - | - | - |
| | PWC-Net (2018) [6] | 2.55 | 3.93 | 10.35 | 33.7 | - | - | - |
| | LiteFlowNet2 (2020) [20] | 2.24 | 3.78 | 8.97 | 25.9 | - | - | - |
| | VCN (2019) [21] | 2.21 | 3.68 | 8.36 | 25.1 | - | - | - |
| | MaskFlowNet (2020) [22] | 2.25 | 3.61 | - | 23.1 | - | - | - |
| | FlowNet2 (2017) [7] | 2.02 | 3.54 | 10.08 | 30.0 | 3.96 | 6.02 | - |
| | RAFT (2020) [1] | 1.43 | 2.71 | 5.04 | 17.4 | - | - | - |
| | SCV (2021) [8] | 1.29 | 2.95 | 6.80 | 19.3 | - | - | - |
| | GMA (2021) [9] | 1.30 | 2.74 | 4.69 | 17.1 | - | - | - |
| | GMFlowNet (2022) [11] | **1.14** | 2.71 | **4.24** | **15.4** | - | - | - |
| | CRAFT (2022) [12] | 1.27 | 2.79 | 4.88 | 17.5 | - | - | - |
| | EMFANet | 1.19 | **2.52** | 4.66 | 16.5 | - | - | - |
| C+T+S+K+H | LiteFlowNet2 (2020) [20] | (1.30) | (1.62) | (1.47) | (4.8) | 3.48 | 4.69 | 7.74 |
| | PWC-Net++ (2019) [37] | (1.71) | (2.34) | (1.50) | (5.3) | 3.45 | 4.60 | 7.72 |
| | VCN (2019) [21] | (1.66) | (2.24) | (1.16) | (4.1) | 2.81 | 4.40 | 6.30 |
| | MaskFlowNet (2020) [22] | - | - | - | - | 2.52 | 4.17 | 6.10 |
| | RAFT (2020) [1] | (0.76) | (1.22) | (0.63) | (1.5) | 1.61 | 2.86 | 5.10 |
| | SCV (2021) [8] | (0.79) | (1.70) | (0.75) | (2.1) | 1.77 | 3.88 | 6.17 |
| | GMA (2021) [9] | (0.62) | (1.06) | (0.57) | (1.2) | 1.39 | 2.47 | 5.15 |
| | GMFlow (2022) [23] | - | - | - | - | 1.74 | 2.90 | 9.32 |
| | MS-RAFT (2022) [10] | - | - | - | - | 1.37 | 2.67 | 4.88 |
| | GMFlowNet (2022) [11] | (0.59) | (0.91) | (0.64) | (1.51) | 1.39 | 2.65 | 4.79 |
| | CRAFT (2022) [12] | (0.60) | (1.06) | (0.58) | (1.34) | 1.45 | **2.42** | 4.79 |
| | AnyFlow (2023) [26] | - | - | - | - | 1.26 | 2.63 | 4.64 |
| | DistractFlow (2023) [27] | - | - | - | - | - | 2.77 | 4.82 |
| | OCAI (2024) [28] | - | - | - | - | - | 2.63 | **4.13** |
| | EMFANet | **(0.53)** | **(0.91)** | **(0.56)** | **(1.16)** | **1.26** | 2.58 | 4.53 |

Table 2: The ablation study.

| No. | Multi-scale | EMFA | Sintel (train) | | KITTI-15 (train) | | FlyingThings3D (val) | | FlyingChairs (val) |
|---|---|---|---|---|---|---|---|---|---|
| | | | Clean | Final | Fl-epe | Fl-all | Clean | Final | EPE |
| 1 | ✗ | ✗ | 1.40 | 2.70 | 5.03 | 17.5 | 4.24 | 3.80 | 0.82 |
| 2 | ✓ | ✗ | 1.21 | 2.85 | 4.58 | 14.4 | 3.87 | 3.64 | 0.77 |
| 3 | ✗ | ✓ | 1.33 | 2.67 | 4.84 | 17.3 | 3.97 | 3.53 | 0.80 |
| 4 | ✓ | ✓ | 1.20 | 2.53 | 4.54 | 16.4 | 3.56 | 3.36 | 0.75 |

#### *4.2.2. Sintel benchmark*

The pre-trained EMFANet is fine-tuned on the training data of FlyingThins3D, Sintel, KITTI-15, and HD1K, and then evaluated on the Sintel test set. The EMFANet achieves the best performance on the clean pass. It reduces 8.0% error compared to the second best MS-RAFT [10], which shows that the proposed EMFA module has a very positive impact on optical flow estimation. However, the EMFANet is not competitive on the final pass compared to the GMA [9] and the CRAFT [12]. The performance drop may originate from that the final pass includes more motion blur areas and textureless surfaces, which have less-informative feature representations and lead to improper clustering in the EMFA module. This problem can be alleviated by clustering more times.

#### *4.2.3. KITTI-15 benchmark*

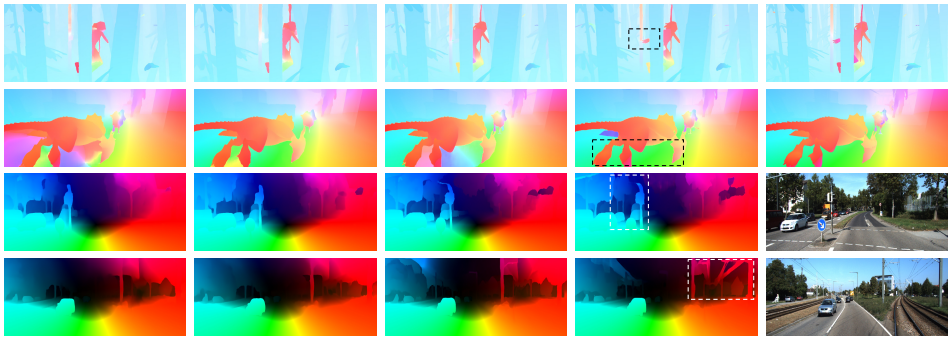
The EMFANet is further fine-tuned on the training set of KITTI-15 and HD1K, and evaluated on the KITTI-15 test set. The EMFANet achieves the lowest Fl-all and improves 5.4% over the second best CRAFT [12]. The multi-scale concept of EMFANet makes it able to estimate optical flows of fast-moving small objects in urban traffic scenes.

#### *4.2.4. Qualitative comparison*

We visualize the estimated optical flows in Figure 4 for qualitative comparison. It can be seen that the EMFANet is able to produce optical flows with more precise and clear boundaries. By extending the EMFA to higher resolution, our method is more robust to occlusions of small objects, as indicated by the motion details of hand shown on the first row of Figure 4.

### *4.3. Ablation study*

The ablation study is conducted to investigate the effect of the proposed EMFA module. All ablated versions are trained on the FlyingChairs and the FlyingThings3D successively, each by 100k iterations. The ablation results on



(a) RAFT (b) GMA (c) MS-RAFT (d) EMFANet (e) Ground truth / Image

Figure 4: The qualitative comparison of RAFT [1], GMA [9], MS-RAFT [10], and the proposed EMFANet. The top two rows show the results on Sintel (test) and the bottom two show the results on KITTI-15 (test). The areas where the EMFANet significantly outperforms other methods are marked by dotted boxes.

Table 3: The GPU memory consumption and running time during training and test on the Sintel dataset. The input resolution is $368 \times 768$. The inference time per frame is taken by the average value of running 100 times.

| Method | batch=1 | batch=2 | Inference time |
|---|---|---|---|
| **Multi-scale + GMA** | 12.4 GB | 23.7 GB | 0.5128s |
| **Multi-scale + EMFA** | 8.5 GB | 15.9 GB | 0.3952s |

the Sintel (train), KITTI-15 (train), FlyingThings3D (val), and FlyingChairs (val) are reported in Table 2. Notably, the FlyingThings3D (val) contains 1024 image pairs in both the clean and final passes, which are evenly sampled from the whole validation set according to [23].

*4.3.1. Multi-scale*

✓ indicates that the optical flows are predicted at three scales, including the $\frac{1}{16}$, $\frac{1}{8}$, and $\frac{1}{4}$ of the original resolution [10]. ✗ indicates that the optical flows are predicated at single scale, which is the $\frac{1}{8}$ of the original resolution [1]. Comparing the No. 1 and No. 2 ablation methods, we can clearly see that the multi-scale scheme reduces the EPE by a large margin. The large and minor motions can be captured via estimating optical flows at lower and higher resolutions, respectively.

*4.3.2. EMFA*

✓ indicates that the EMFA module is inserted after the motion feature encoder at each prediction scale. ✗ indicates that the EMFA module is not

used. As shown by the results of the No. 1 and No. 3 ablation methods, adding the EMFA module improves the model performances on all the four datasets. This demonstrates that the occlusion problem can be effectively alleviated by the proposed EMFA module. The EMFA is similar to the GMA [9], but it has much fewer computations. It can also be observed that the No. 4 ablation method improves the No. 2, indicating that the occlusions are handled at higher resolution via the EMFA module.

#### *4.3.3. Analysis of computational complexity*

We analyze the computational complexity of the proposed EMFA against the GMA. For simplicity, given the contextual and motion features of the same size $N_s \times D_c$, the complexity of the GMA is $\mathcal{O}({N_s}^2 D_c)$, which is the same with the self-attention [13]. For the EMFA, the LSH clustering is implemented by matrix multiplication, costing $\mathcal{O}(N_s D_c N_h)$, where $N_h$ represents the number of split chunks. The self-attention is performed within each chunk and the complexity gives at $\mathcal{O}(N_h {N_\Phi}^2 D_c) = \mathcal{O}(N_s D_c N_\Phi)$. The $N_\Phi$ denotes the chunk size and $N_s = N_h \times N_\Phi$. Therefore, given the clustering times $N_r$, the overall complexity of the EMFA is $\mathcal{O}(N_s D_c N_r (N_\Phi + N_h))$. Clearly, it is much lower than the GMA.

We extend both the GMA and EMFA to higher resolution, and compare their memory consumption and inference time during training and test, as shown in Table 3. We can see that the EMFA is computational affordable benefiting from the clustering scheme. In addition, the EMFA significantly reduces the computational complexity comparing their running speed.

### *4.4. Choice of hyper parameters*

In this section, we conduct controlled experiments to analyze the hyper parameters. The experimental details are kept the same with Section 4.3. The results are reported in Table 4. The settings which are used in our final model are underlined.

#### *4.4.1. Clustering methods*

'LSH' and 'E2LSH' indicates that the query/key vectors are clustered by the LSH scheme and the E2LSH (Exact Euclidean Locality-Sensitive Hashing [38]) scheme, respectively. While 'Random' indicates that the vectors are randomly clustered. We can see that the random clustering in EMFA leads

Table 4: The hyper parameters in EMFA. The settings which are used in our final model are underlined.

| Experiment | Method | Sintel (train) | | KITTI-15 (train) | | FlyingThings3D (val) | | FlyingChairs (val) |
|---|---|---|---|---|---|---|---|---|
| | | Clean | Final | Fl-epe | Fl-all | Clean | Final | EPE |
| Clustering methods | <u>LSH</u> | 1.20 | 2.53 | 4.54 | 16.4 | 3.56 | 3.36 | 0.75 |
| | E2LSH | 1.22 | 2.60 | 4.73 | 17.5 | 3.70 | 3.42 | 0.74 |
| | Random | 1.27 | 2.81 | 4.87 | 18.1 | 3.96 | 3.60 | 0.78 |
| Clustering times | $N_r = 1$ | 1.22 | 2.58 | 4.81 | 16.9 | 3.65 | 3.42 | 0.76 |
| | $N_r = 2$ | 1.20 | 2.57 | 4.59 | 16.6 | 3.65 | 3.36 | 0.76 |
| | <u>$N_r = 4$</u> | 1.20 | 2.53 | 4.54 | 16.4 | 3.56 | 3.36 | 0.75 |
| | $N_r = 8$ | 1.20 | 2.55 | 4.53 | 16.4 | 3.54 | 3.33 | 0.75 |
| Chunk size | $N_\Phi = 32$ | 1.21 | 2.59 | 4.77 | 17.1 | 3.62 | 3.36 | 0.74 |
| | $N_\Phi = 64$ | 1.17 | 2.54 | 4.62 | 16.9 | 3.56 | 3.30 | 0.74 |
| | <u>$N_\Phi = 128$</u> | 1.20 | 2.53 | 4.54 | 16.4 | 3.56 | 3.36 | 0.75 |
| | $N_\Phi = 512$ | 1.29 | 2.70 | 5.15 | 17.3 | 3.85 | 3.52 | 0.76 |
| Attended chunks | 1 | 1.20 | 2.58 | 4.64 | 16.8 | 3.57 | 3.35 | 0.75 |
| | <u>3</u> | 1.20 | 2.53 | 4.54 | 16.4 | 3.56 | 3.36 | 0.75 |

to a great performance drop. This is because that the correlated vectors are not sorted into the same cluster. The ambiguities at occlusions can not be resolved by aggregating motion information from uncorrelated vectors. Such a random feature aggregation may even bring a negative effect on the final optical flow estimation. Different from the LSH, the E2LSH is based on $p$-stable distributions and it measures the similarity by computing the Euclidean distance. The LSH performs better than the E2LSH as the cosine similarity is more suitable for deep network features.

#### *4.4.2. Clustering times*

For the LSH-based clustering, there is always possible that correlated vectors are clustered into different hash buckets due to the uncertainty of Gaussian sampling in Equation (6). To achieve a robust EMFA, we propose to sample several matrices $\boldsymbol{R}$ and cluster several times, and take the combination of all the results. As indicated by the results in Table 4, increasing the clustering times constantly improve the optical flow accuracy. As expected, the best result is achieved with the largest $N_r$, but it also yields the worst computational cost. The $N_r$ is set as 4 considering a trade-off between the accuracy and efficiency.

#### *4.4.3. Chunk size*

The efficiency of the EMFA over the GMA comes from that the attention is parallelly performed within each chunk, which contains only a few correlated elements. A small chunk size is sufficient if the elements are correctly

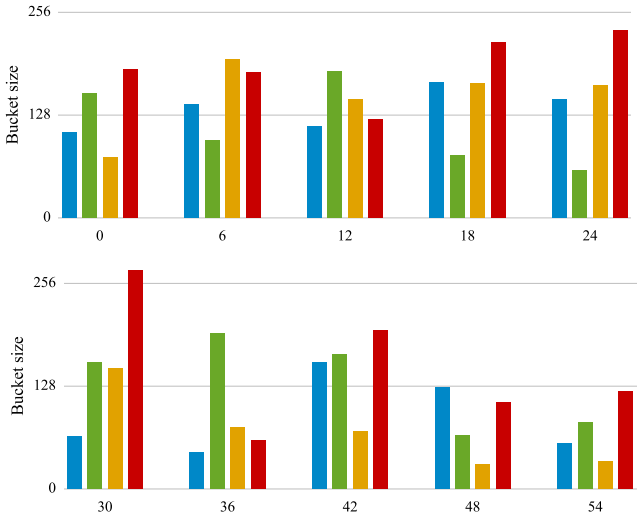



Figure 5: The size of a few hash buckets. The abscissa denotes the hash codes. The color denotes the clustering at different times. The clustered vectors are from the contextual features at scale $s = 8$, of the first test sample in KITTI-15 (test).

clustered. A large chunk size is able to cover more elements, but it has a price of more computational cost. Increasing the size improves the accuracy greatly. This demonstrates that the proposed EMFA works well by focusing on the most informative elements at a global scale. The $N_\Phi$ is set as 128 since a larger value improves little.

*4.4.4. Attended chunks*

'1' denotes that the attention is performed within the current chunk. '3' denotes that the attention is performed within the current chunk and its two adjacent chunks (back and before). It can be seen that the results are further improved by attending to more chunks, while the computations increase slightly. We further visualize the size of a few hash buckets (clusters) in Figure 5. Despite the bucket size is unbalanced, most of them are around 128 (i.e. the chunk size). Thus, the vectors belong to the same bucket can be robustly covered by attending to adjacent chunks.

## 5. Conclusion

In this paper, an efficient motion feature aggregation (EMFA) module is proposed targeting at the occlusion problem in optical flow estimation. Compared to the original global motion aggregation (GMA), the EMFA has

a much lower computational complexity since it performs the aggregation from only a few correlated pixels. The EMFA is further extended to estimate optical flows at higher resolution and the final model achieves better performance compared to the state-of-the-art methods on publicly available benchmarks.

## References


[1] Z. Teed, J. Deng, Raft: Recurrent all-pairs field transforms for optical flow, in: Proceedings of the European Conference on Computer Vision, Springer, 2020, pp. 402–419.

[2] H. Zhu, X. Sun, Q. Zhang, Q. Wang, A. Robles-Kelly, H. Li, S. You, Full view optical flow estimation leveraged from light field superpixel, IEEE Transactions on Computational Imaging 6 (2019) 12–23.

[3] M. Almatrafi, K. Hirakawa, Davis camera optical flow, IEEE Transactions on Computational Imaging 6 (2019) 396–407.

[4] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, T. Brox, Flownet: Learning optical flow with convolutional networks, in: Proceedings of the IEEE International Conference on Computer Vision, IEEE, 2015, pp. 2758–2766.

[5] T.-W. Hui, X. Tang, 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, IEEE, 2018, pp. 8981–8989.

[6] D. Sun, X. Yang, M.-Y. Liu, 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, IEEE, 2018, pp. 8934–8943.

[7] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, 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, IEEE, 2017, pp. 2462–2470.

[8] S. Jiang, Y. Lu, H. Li, R. Hartley, Learning optical flow from a few matches, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2021, pp. 16592–16600.

[9] S. Jiang, D. Campbell, Y. Lu, H. Li, R. Hartley, Learning to estimate hidden motions with global motion aggregation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, IEEE, 2021, pp. 9772–9781.

[10] A. Jahedi, L. Mehl, M. Rivinius, A. Bruhn, Multi-scale raft: Combining hierarchical concepts for learning-based optical flow estimation, in: 2022 IEEE International Conference on Image Processing, IEEE, 2022, pp. 1236–1240.

[11] S. Zhao, L. Zhao, Z. Zhang, E. Zhou, D. Metaxas, Global matching with overlapping attention for optical flow estimation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2022, pp. 17592–17601.

[12] X. Sui, S. Li, X. Geng, Y. Wu, X. Xu, Y. Liu, R. Goh, H. Zhu, Craft: Cross-attentional flow transformer for robust optical flow, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2022, pp. 17602–17611.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017).

[14] A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, L. Schmidt, Practical and optimal lsh for angular distance, Advances in neural information processing systems 28 (2015).

[15] C. Bailer, B. Taetz, 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, IEEE, 2015, pp. 4015–4023.

[16] R. Schuster, C. Bailer, O. Wasenmüller, D. Stricker, Flowfields++: Accurate optical flow correspondences meet robust interpolation, in: Proceedings of the IEEE International Conference on Image Processing, IEEE, 2018, pp. 1463–1467.

[17] J. Xu, R. Ranftl, V. Koltun, Accurate optical flow via direct cost volume processing, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2017, pp. 1289–1297.

[18] J. Wulff, L. Sevilla-Lara, M. J. Black, Optical flow in mostly rigid scenes, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2017, pp. 4671–4680.

[19] Z. Yin, T. Darrell, F. Yu, Hierarchical discrete distribution decomposition for match density estimation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2019, pp. 6044–6053.

[20] T.-W. Hui, X. Tang, C. C. Loy, A lightweight optical flow cnn—revisiting data fidelity and regularization, IEEE Transactions on pattern analysis and machine intelligence 43 (8) (2020) 2555–2569.

[21] G. Yang, D. Ramanan, Volumetric correspondence networks for optical flow, Advances in neural information processing systems 32 (2019).

[22] 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, IEEE, 2020, pp. 6278–6287.

[23] H. Xu, J. Zhang, J. Cai, H. Rezatofighi, D. Tao, Gmflow: Learning optical flow via global matching, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2022, pp. 8121–8130.

[24] Z. Huang, X. Shi, C. Zhang, Q. Wang, K. C. Cheung, H. Qin, J. Dai, H. Li, Flowformer: A transformer architecture for optical flow, in: Proceedings of the European conference on computer vision, Springer, 2022, pp. 668–685.

[25] X. Shi, Z. Huang, D. Li, M. Zhang, K. C. Cheung, S. See, H. Qin, J. Dai, 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, IEEE, 2023, pp. 1599–1610.

[26] H. Jung, Z. Hui, L. Luo, H. Yang, F. Liu, S. Yoo, R. Ranjan, D. Demandolx, Anyflow: Arbitrary scale optical flow with implicit neural representation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2023, pp. 5455–5465.

[27] J. Jeong, H. Cai, R. Garrepalli, F. Porikli, Distractflow: Improving optical flow estimation via realistic distractions and pseudo-labeling, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2023, pp. 13691–13700.

[28] J. Jeong, H. Cai, R. Garrepalli, J. M. Lin, M. Hayat, F. Porikli, Ocai: Improving optical flow estimation by occlusion and consistency aware interpolation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2024, pp. 19352–19362.

[29] H. Zhao, X. Qi, X. Shen, J. Shi, J. Jia, Icnet for real-time semantic segmentation on high-resolution images, in: Proceedings of the European Conference on Computer Vision, Springer, 2018, pp. 405–420.

[30] R. Xu, D. Wunsch, Survey of clustering algorithms, IEEE Transactions on neural networks 16 (3) (2005) 645–678.

[31] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Proceedings of the International Conference on Machine Learning, PMLR, 2015, pp. 448–456.

[32] Z. Liu, L. Zhang, W. Chen, X. Wu, Z. Li, S&cnet: A lightweight network for fast and accurate depth completion, Journal of Visual Communication and Image Representation 79 (2021) 103220.

[33] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, 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, IEEE, 2016, pp. 4040–4048.

[34] D. J. Butler, J. Wulff, G. B. Stanley, M. J. Black, A naturalistic open source movie for optical flow evaluation, in: Proceedings of the European Conference on Computer Vision, Springer, 2012, pp. 611–625.

[35] M. Menze, A. Geiger, Object scene flow for autonomous vehicles, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2015, pp. 3061–3070.

[36] 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: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, IEEE, 2016, pp. 19–28.

[37] D. Sun, X. Yang, M.-Y. Liu, J. Kautz, Models matter, so does training: An empirical study of cnns for optical flow estimation, IEEE transactions on pattern analysis and machine intelligence 42 (6) (2019) 1408–1423.

[38] A. Dasgupta, R. Kumar, T. Sarlós, Fast locality-sensitive hashing, in: Proceedings of the International Conference on Knowledge Discovery and Data Mining, 2011, pp. 1073–1081.