

Local Attention Transformers for High-Detail Optical Flow Upsampling

Alexander Gielisse, Nergis Tomen, Jan van Gemert
Computer Vision Lab, Delft University of Technology

a.s.gielisse@tudelft.nl, n.tomen@tudelft.nl, j.c.vangemert@tudelft.nl

Abstract

Most recent works on optical flow use convex upsampling as the last step to obtain high-resolution flow. In this work, we show and discuss several issues and limitations of this currently widely adopted convex upsampling approach. We propose a series of changes, in an attempt to resolve current issues. First, we propose to decouple the weights for the final convex upsampler, making it easier to find the correct convex combination. For the same reason, we also provide extra contextual features to the convex upsampler. Then, we increase the convex mask size by using an attention-based alternative convex upsampler; Transformers for Convex Upsampling. This upsampler is based on the observation that convex upsampling can be reformulated as attention, and we propose to use local attention masks as a drop-in replacement for convex masks to increase the mask size. We provide empirical evidence that a larger mask size increases the likelihood of the existence of the convex combination. Lastly, we propose an alternative training scheme to remove bilinear interpolation artifacts from the model output. Our proposed ideas could theoretically be applied to almost every current state-of-the-art optical flow architecture. On the FlyingChairs + FlyingThings3D training setting we reduce the Sintel Clean training end-point-error of RAFT from 1.42 to 1.26, GMA from 1.31 to 1.18, and that of FlowFormer from 0.94 to 0.90, by solely adapting the convex upsampler.

1. Introduction

Current state-of-the-art deep optical flow models [9, 14, 25, 26, 30, 36, 38] is heavily inspired by RAFT [30]. RAFT applies an iterative, recurrent, optimization and makes this possible by reducing memory and compute by predicting flow at 1/8th of the input resolution. This low-resolution flow map is then upsampled to full resolution. Upsampling methods for optical flow have different requirements than traditional image upsampling methods like bilinear interpolation. *i.e.*, if two objects at an edge move in different directions, interpolating them in the middle will give values

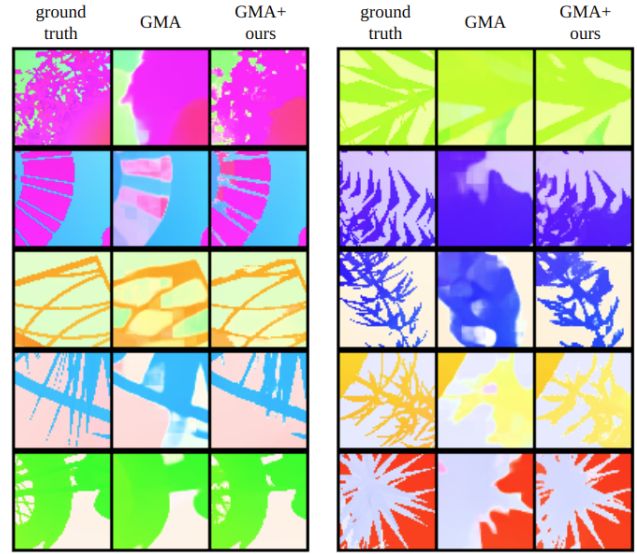


Figure 1. Current optical flow methods ignore fine details. We show 10 High-detail 64x64 patches from the FlyingThings3D [19] dataset and compare a recent baseline (GMA [14]) with adding our upsampler. Our upsampling better preserves fine details.

around the mean of the two directions. This is never correct in terms of optical flow; a pixel either follows one object, or the other. Alternatively, upsampling methods that do not use interpolation like nearest neighbor upsampling makes flow edges jagged and not aligned with object edges.

To solve these issues for optical flow, RAFT [30] proposes convex upsampling. There, the main idea is to upsample by a convex combination of the low-resolution neighbors: each low-resolution pixel is weighted, where all weights must be positive and sum to 1, which is easily obtained by applying the softmax function, see 2. Following the success of RAFT [30], this convex upsampling is currently adopted practically unchanged in all current SOTA flow models [9, 14, 25, 26, 30, 33, 36, 38].

In this paper, we make the observation that convex upsampling for optical flow has received relatively little attention in the literature: the design of convex upsampling has not changed much since RAFT [30].

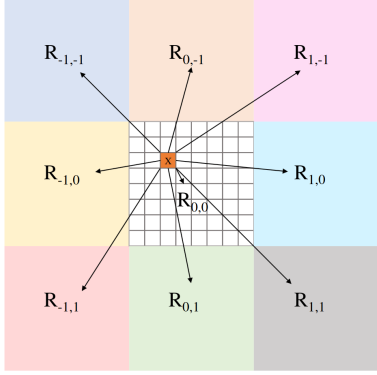


Figure 2. Original convex optical flow upsampling as proposed by RAFT [30]. A high-resolution sub-pixel value is written as a convex combination of the low-resolution input. The convex mask weights are guaranteed to be positive and sum to 1 by using softmax. This convex upsampling is adopted by the state of the art. We propose improvements based on the observation that the convex combination through a softmax can be rephrased as neighborhood attention [7], leading to increased accuracy, which can be used as a drop-in replacement for all existing models.

Here, we rephrase the traditional convex upsampling in a more flexible model using Neighborhood Attention (NA) [7]. Neighborhood Attention is a natural model for convex upsampling, as it is also local, and by the softmax operator inherently provides a convex combination. One of the benefits is that NA allows to decouple the number of learnable parameters from the mask size, allowing larger input sizes.

We have the following contributions. We rephrase convex upsampling as Neighborhood Attention (NA). NA allows us to evaluate larger mask sizes, making more solutions possible. It also allows us to replace the 1-step 8x upsampling with three hierarchical steps of 2x upsampling, which helps retain spatial information. The hierarchical upsampling allows us to also include the input image features at matching resolutions, to better align flow with object edges. We also investigate decoupling the final upsampling model from the intermediate upsamplings used in the recurrent optimization. As a final investigation we explore the role of optical flow sampling in data-augmentation.

Our proposed ideas could theoretically be applied to almost every current state-of-the-art optical flow architecture. On the FlyingChairs + FlyingThings3D training setting we reduce the Sintel Clean training end-point-error of RAFT from 1.42 to 1.26, GMA from 1.31 to 1.18, and that of FlowFormer from 0.94 to 0.90, by solely adapting the convex upsampler. We will make all code available.

2. Related Work

Optical Flow is typically estimated between two consecutive frames by matching image-1 to image-2, and the seminal RAFT [30] approach inspired much follow up work. RAFT uses a gated recurrent unit in a step-wise, recurrent, optimization process. Because of its computational complexity, the recurrent optimization is done on 1/8th of the input image resolution, after which the low resolution output flow is upsampled to full resolution. The follow up work done by GMA [14] adds global attention on the input features to add global, contextual information. Similarly, SeparableFlow [36] improves the correlation volume construction by first ensuring global contextual features. This idea is extended by FlowFormer(++) [9, 25], CRAFT [26] and GMFlowNet [38] who add strong Transformer blocks. Essentially, RAFT is extended by improving and replacing different components other than the step-wise optimization, as this optimization approach remains the global design. So, the current optical flow state of the art [9, 14, 25, 26, 30, 36, 38] is based on RAFT and inherits its main properties: low resolution iterative optimization followed by upsampling. Here, we focus on these properties and on the low resolution upsampling in particular.

MS-RAFT(+) [12, 13] also explores different resolutions in a setting based on RAFT [30]. Important for us is that the convex upsampling is different, as they upsample 3 times by a factor of 2, rather than once by a factor of 8. Inspired by this, we investigate the impact of this 3-step approach when used on the other state-of-the-art networks that currently use a single upsampling operation by factor 8.

Upsampling is fundamental in computer vision tasks. Non learning-based approaches such as Nearest Neighbor, bilinear, or bicubic interpolation [6, 29], and learning-based approaches such as Transposed convolutions [18], PixelShuffle as used in super resolution [24], or convex upsampling [30]. Learned-based upsampling can be trained end-to-end [18, 24, 30] or progressive as done in GANs [15]. For optical flow, bilinear- or Nearest Neighbor interpolation can be used for decent performance [4, 11, 22] as long as the upsampling factor is small. All the current SOTA optical flow models [9, 14, 25, 26, 30, 33, 36, 38] require upsampling with factor 8. This makes traditional upsampling unsuitable, and hence convex upsampling is widely adopted by these models. Here, we extend this reasoning and formulate convex upsampling as local self-attention [7].

Attention and Local Attention Self-attention [31] used for image recognition [3] research is bringing back the hierarchical structure of convolutional neural networks. As such, SwinFormer [17] was proposed. In a similar way, Neighborhood Attention (NA) [7] was proposed. While the

general idea of NA is similar to SwinFormer [17], the main design difference is how the local attention maps relate to the queries. NA ensures that as long as a query is not near the image border, the query is always at the center of the local attention map. This property makes these local attention maps effectively similar to convex upsampling masks. An advantage of local attention maps is that their size is decoupled from the number of parameters, so we investigate the use of attention maps when taken directly as a drop-in replacement for convex upsampling masks.

3. Method

We show a visual comparison of the baseline convex upsampling of RAFT [30] versus our proposed method in 3 and will explain its modules in the following.

3.1. RAFT's convex upsampler

Context encoder Most SOTA models for optical flow use a context encoder that takes image-1 as input, generally based on a three-scale ResNet [8] architecture where each scale has a neural network r_l with $l \in (0, 1, 2)$ that down-scales the resolution by factor 2, as follows:

$$\text{image}_{1/2} = r_0(\text{image-1}) \in \mathbb{R}^{(64 \times (H/2) \times (W/2))} \quad (1)$$

$$\text{image}_{1/4} = r_1(\text{image}_{1/2}) \in \mathbb{R}^{(96 \times (H/4) \times (W/4))} \quad (2)$$

$$\text{image}_{1/8} = r_2(\text{image}_{1/4}) \in \mathbb{R}^{(128 \times (H/8) \times (W/8))} \quad (3)$$

Flow predictor The flow predictor takes $\text{image}_{1/8}$ and uses a linear projection to obtain the initial gated recurrent unit (GRU) internal state h_0 , as well as the shared input to each GRU step a . The GRU module is defined as g , and the optical flow map as flow . We define for image input size $H \times W$ that $h = (H/8)$ and $w = (W/8)$.

$$h_0 = \text{Conv}_{1 \times 1}(\text{image}_{1/8}) \in \mathbb{R}^{(128 \times h \times w)} \quad (4)$$

$$a = \text{Conv}_{1 \times 1}(\text{image}_{1/8}) \in \mathbb{R}^{(128 \times h \times w)} \quad (5)$$

$$\text{flow}_0 = 0_{(2 \times h \times w)} \in \mathbb{R}^{(2 \times h \times w)} \quad (6)$$

The correlation volume $c(p)$ in RAFT [30] is defined to provide the correlation information for a given pixel at spatial position p . For I refinement iterations the optimization approach is then defined as following, where $\text{flow}_{(I-1)}$ is the final low-resolution output flow.

$$\text{flow}_1, h_1 = g(h_0, a, \text{flow}_0, c(\text{flow}_0)) \quad (7)$$

$$\text{flow}_2, h_2 = g(h_1, a, \text{flow}_1, c(\text{flow}_1)) \quad (8)$$

...

$$\text{flow}_{(I-1)}, h_{(I-1)} = g(h_{(I-2)}, a, \text{flow}_{(I-2)}, c(\text{flow}_{(I-2)})) \quad (9)$$

$$\text{with}$$

$$\text{flow}_{(I-1)} \in \mathbb{R}^{(2 \times (H/8) \times (W/8))} \quad (10)$$

$$h_{(I-1)} \in \mathbb{R}^{(128 \times (H/8) \times (W/8))} \quad (11)$$

Mask predictor Now take flow_j to be a low-resolution flow map with hidden state h_j where $j \in (0, 1, \dots, (I-1))$. For RAFT's [30] upsampler which is used to upsample the flow_j from size $(h \times w)$ to $(H \times W)$, a convex combination of the low-resolution nearby pixels from flow_j is used. That is, for every low-resolution pixel $P \in \text{flow}_j$, find sub-pixel values p_i by first predicting a 3×3 scalar mask mask_{p_i} for every sub-pixel p_i . This mask_{p_i} has 9 scalar values; $\text{mask}_{p_i} = (w_0, w_1, \dots, w_{(m^2-1)})$ for mask size $m = 3$. For a mask centered on low resolution pixel $P \in \text{flow}_j$, all the values within the mask including the pixel itself, form the neighbors of P ; called $N_P \in \text{flow}_j$. Then, the value of the sub-pixel p_i is calculated using the dot-product as following, for upsampling factor f and mask size m . Furthermore, σ refers to the use of the ReLU [5] activation function.

$$\text{masks} = \text{Conv}_{1 \times 1}(\sigma(\text{Conv}_{3 \times 3}(h_j))) \in \mathbb{R}^{(f^2 m^2)} \quad (12)$$

$$\text{masks} \in \mathbb{R}^{(f^2 \times m \times m)} = \text{masks} \in \mathbb{R}^{(f^2 m^2)} \quad (13)$$

$$i \in (0, 1, \dots, (f^2 - 1))$$

$$(\text{mask}_i, N_P) \in \mathbb{R}^{(m \times m)} \quad (14)$$

$$p_i = \langle \text{softmax}(\text{mask}_i), N_P \rangle \in \mathbb{R}^1 \quad (15)$$

3.2. Neighborhood Attention Transformers for Convex Upsampling

For our attention-based convex upsampler, we start with the same input h_j , but also concatenate $\text{image}_{1/8}$ and flow_j , to build the embedding vectors e for all pixels $P \in \text{flow}_j$ as following.

$$\text{cat} = \text{Cat}(h_j, \text{image}_{1/8}, \text{flow}_j) \quad (16)$$

$$\in \mathbb{R}^{(258 \times h \times w)}$$

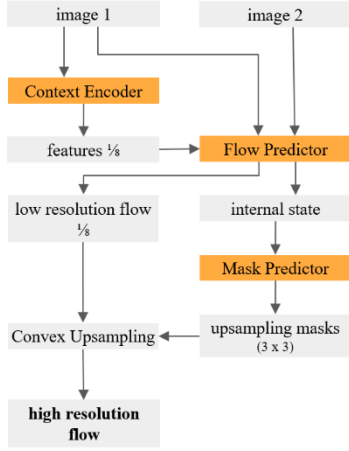
$$e = \text{Conv}_{1 \times 1}(\text{cat}) \in \mathbb{R}^{(D \times h \times w)} \quad (17)$$

Then we apply local Neighborhood Attention Transformers (NAT) [7] for local contextual feature enhancement, using 2 Transformer blocks which sequentially form T . For these Transformer blocks we use dimensionality $D = 128$ for scale $1/8$, $D = 64$ for scale $1/4$, and $D = 32$ for scale $1/2$. We set the head dimensionality to 32. Note that the tokens from the embedding vectors e correspond to the low-resolution pixels $P \in \text{flow}_j$.

$$e_{\text{ctx}} = T(e) \in \mathbb{R}^{(D \times h \times w)} \quad (18)$$

Then, for upsampling factor f , multi-head attention is used to form f^2 attention maps for each P ($h \times w$) from $e_{\text{ctx}} \in \mathbb{R}^{(D \times h \times w)}$. We use head dimensionality of $\frac{1}{2}D$ such that we halve the size of the embedding dimensionality for each upsampling operation. For f^2 heads, this requires a query, key, and value dimensionality of $\frac{1}{2}f^2D$, which can then be reshaped to obtain f^2 query, key, and value maps with dimensionality $\frac{1}{2}D$.

Original Convex Upsampler



Transformers for Convex Upsampling (ours)

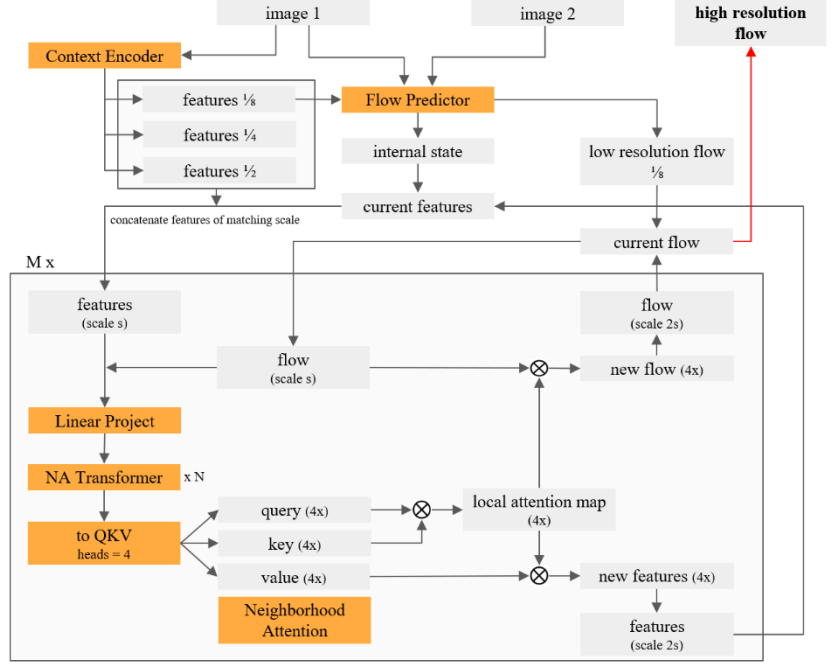


Figure 3. **Left:** the original convex upscaling method as proposed by RAFT [30]. **Right:** our proposed multi-step Transformer convex upsampling network. The feature extractor is adopted from RAFT [30] but we extract the features at 3 all intermediate scales. The Neighborhood Attention Transformer blocks [7] perform local neighborhood attention to enhance features. Attention is used to both upsample the low-resolution flow as well as the feature maps. The block at the bottom is executed M times and upsamples by factor 2, and hence for $M = 3$ the low-resolution flow is upsampled by factor 8.

$$Q = \text{Conv}_{1 \times 1}(e_{\text{ctx}}) \in \mathbb{R}^{(\frac{1}{2}f^2 D \times h \times w)} \quad (19)$$

$$K = \text{Conv}_{1 \times 1}(e_{\text{ctx}}) \in \mathbb{R}^{(\frac{1}{2}f^2 D \times h \times w)} \quad (20)$$

$$V = \text{Conv}_{1 \times 1}(e_{\text{ctx}}) \in \mathbb{R}^{(\frac{1}{2}f^2 D \times h \times w)} \quad (21)$$

$$(Q, K, V) \in \mathbb{R}^{(f^2 \times \frac{1}{2} D \times h \times w)} \quad (22)$$

Then, we construct the local attention maps using neighborhood attention (NA) [7] with window size m . Note that to obtain these local attention maps (LAM) they are normalized using the softmax [1] function, so similarly to convex upsampling masks; the values are positive and sum to 1.

$$\text{LAM} = \text{NA}_{\text{maps}}(m, Q, K) \in \mathbb{R}^{(4 \times m \times m \times h \times w)} \quad (23)$$

We then aggregate these local attention maps with the low-resolution flow f_j , as well as the value features V . Again, we use neighborhood attention [7] for this.

$$\text{flow}_{\text{up}} = \text{Aggregate}(\text{LAM}, \text{flow}_j) \quad (24)$$

$$h_{\text{up}} = \text{Aggregate}(\text{LAM}, V) \quad (25)$$

Note that this aggregation operation on a per-pixel level is implemented as following, when $\text{mask}_i \in \text{LAM}_{\text{pre}}$ where

LAM_{pre} refers to the attention maps from LAM before softmax normalization.

$$p_i = \langle \text{softmax}(\text{mask}_i), N_P \rangle \in \mathbb{R}^1 \quad (26)$$

So, aggregation as used in local attention is effectively similar to convex upsampling, as Equation 15 and Equation 26 are equivalent. For both, the dot product is taken within a sliding window between the convex maps, or attention maps, and the low-resolution flow, or the values. For simplicity, we ignore the different padding implementation, as this is not necessarily relevant here.

From Equation 24, we obtain flow_{up} , which is the by factor f convex upsampled flow. We also obtain the by factor f upsampled features h_{up} based on Equation 25.

Hierarchical upsampling In the convex upsampling implementation of RAFT as described in Section 3.1, upsampling is done once by factor 8. The most important reason for this is that convex upsampling expects a feature map and a low-resolution flow map as inputs. However, it only provides a single output, namely the upsampling masks; see Equation 12. Therefore, the operation cannot easily be repeated. For our TCU, this is different. As we can see from

Equations 24 and 25; we have the same inputs as outputs, but at a different scale. We can effectively repeat this operation as often as needed, and are able to upsample 3 times by factor 2, rather than once by factor 8. This has several advantages, one is a spatial inductive bias; the network only has to learn the alignment of 4 sub-pixels at the time, rather than 64 at once.

Adding the input image as extra information Another advantage of the hierarchical approach as discussed in the previous paragraph is that it allows for concatenating feature maps from different scales. Note that from Equations 1 till 3 we have image features at 3 scales, yet only use the $1/8$ scale from Equation 3 as only these are used as inputs in equations 4 and 5. If we adopt our hierarchical method, we are able to concatenate each of the scales from Equations 1 ($1/2$), 2 ($1/4$) and 3 ($1/8$), to the upsampling step of our convex upsampling with corresponding scale, as could be done in Equation 16.

Larger mask size For the traditional convex upsampler, it is difficult to increase the mask size m due to the use of a single vector of size $m^2 f^2$ for the mask values, with mask size m and upsampling factor f . This is because Equation 12 uses a fully-connected layer for this purpose, and hence increasing m comes with a strong increase in the number of parameters, and is difficult to optimize. Fortunately, our Transformer based Convex Upsampler (TCU) does not have this problem. As the convex masks are simply local attention maps, and the size of local attention maps is not dependent on the number of parameters, we can freely increase the mask size m , as long as it fits in memory.

We use an increased mask size to explore finding new solutions. A strong limitation of convex upsampling is that a high-resolution pixel can only be predicted correctly if there exists a convex combination of the low-resolution neighbor pixels N_P such that the dot-product forms the desired output value. Therefore, if the low-resolution flow map N_P is not correct or not locally informative enough and no such convex combination exist, the desired output value can never be obtained. We propose to increase the size of N_P and include more low-resolution pixels, rather than just look at the direct neighbors using a 3×3 mask. For our sequential upsampling steps we use mask sizes (9, 7, 5), in that order from low- to high-resolution.

Decoupling the upsampling Many flow prediction architectures [9, 14, 25, 26, 30, 36, 38] take a recurrent step-wise refinement approach to solve optical flow, as we describe in Equation 7 till 10. During training, a loss value is calculated for each intermediate flow map $\text{flow}_0, \text{flow}_1, \dots, \text{flow}_{(I-1)}$, where each flow map requires to be upsampled to high resolution in order to compare it to the

ground truth. Many SOTA works [9, 14, 25, 26, 30, 36, 38] choose to share the same convex upsampler and its weights over all steps. However, we consider the first flow predictions to be extremely noisy variants of the final flow. When the convex upsampler and its weights are shared over all steps, this is effectively equivalent to just adding strong noise to a part of the input data. In general, the loss is down-weighted for the first steps, but this could be insufficient. We want the convex upsampler of the final output refinement iteration to fully focus on its own objective, and not have its parameters shared with noisy estimates from earlier refinement iterations. To achieve this, we propose to decouple the convex upsampler of the last refinement iteration and give it its own weights.

This also brings the advantage that a different upsampling method can be used for the last refinement iteration. Our Transformed-based convex upsampling approach, TCU, uses more memory than the original convex upsampling approach. So, we exploit the idea of a decoupled upsampler for the last refinement iteration to make our TCU model feasible in practise. Namely, the original shared convex upsampler is used for the first $(I - 1)$ refinement iterations, and TCU is only used for the last refinement iteration which provides the final model output. Note that at test-time, only the upsampler of the last refinement iteration is used.

4. Effect of sampling in data-augmentation

In addition to the architecture change, we also investigate the training scheme. Noticeably, almost all recent works on optical flow are based on the same original PyTorch implementation of RAFT [30], and all use bilinear sampling for augmentation of their training data. The original reason of RAFT for convex upsampling was to avoid bilinear upsampling on flow maps. However, in the current setting convex upsampling is used, but with the learning objective to predict bilinearly interpolated flow. Interestingly, to the best of our knowledge, all public top submissions to the Sintel [2] leaderboard show bilinear interpolation artifacts in the form of white and non-crisp edges, even though these artifacts are not in the non-augmented training data. We explore an additional training scheme to remove bilinear interpolation artifacts. Disabling this interpolation in the augmentation is likely not a good idea, as it is used to avoid overfitting. Instead, we propose an additional training scheme; (-AUG). There, a trained model is trained for an additional 40K iterations with all interpolation-based augmentations disabled.

5. Experiments

Finetuning The only difference from the original training setting is the convex upsampler from the last refinement iteration. Training all model components from ran-

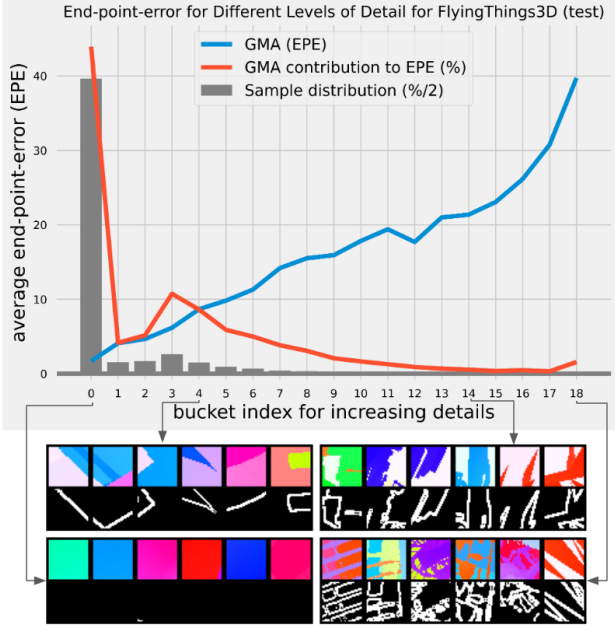


Figure 4. The average end-point-error for increasing amount of detail on FlyingThings3D (test) [19], after being trained on C+T. The performance degrades fast for higher amount of detail. To give an impression of the level of detail per bucket, 6 random patches from each bucket are shown for FlyingThings3D [19]. The contribution of each level of detail to the overall EPE is given as well. Although it is not provided here, the graph for the Sintel [2] dataset looks almost similar.

dom initialization in this highly similar setting would be unnecessarily costly. Instead, all the training sessions are started with pre-trained weights for the flow predictor and the convex upsampler of the first $(I - 1)$ iterations. Only the weights of the convex upsampler for the last refinement iteration are randomly initialized. For all experiments, we fine-tune for 100K iterations with a batch size of 3, on the dataset that the model was last trained on. A learning rate of $1e - 4$ is used for the pre-trained weights, and a learning rate of $2e - 4$ is used for the untrained upsampler of the last refinement iteration. We will make all code available.

5.1. Performance on High-Detail Areas

An important reason for our upsampler is the performance on high-detail areas. To investigate this, we look at the performance on non-overlapping 32×32 patches of the test data. We take the ground truth optical flow and assume the number of edge pixels in the ground truth flow map to strongly correlate with the amount of details, and hence with the difficulty for the convex upsampler. We extract spatial gradients using the Kornia library [23], and consider the L_2 norm on the 4-dimensional vector that comes from extracting the ∂x and ∂y gradients for each XY chan-

nel as an edge detector. To obtain a binary edge map, a binary threshold of 8 is used. To get the level of detail for a patch, the average value is taken of the binary edge map of that patch. Next, we plot the average end-point-error (EPE) for each level of detail, for which a bin width of 0.02 is used. Samples with level of detail that is larger than the specified domain are placed in the last bucket.

The results in Figure 4 show a strong degrade in accuracy for increasing amounts of detail, which confirms that our hypothetical problem exists. We provide the statistics on the amount of samples per bucket from Figure 4 in Table 1. From this table, note for example that the buckets ($b \geq 8$) only contain 2% of the patches, yet contribute for 13% to the end-point-error. This strongly highlights the importance of our method, as even though the amount of high-detail patches is low, its impact on the end-point-error can be significant.

5.2. Transformers for Convex Upsampler

Next, we investigate the impact of the proposed individual components. **+DC** refers to decoupling the last refinement’s convex upsampler, and giving it its own weights. **+FT** refers to concatenating the features from the context branch to the input of the convex upsampler. When TCU is used features are added at all scales, otherwise only the low resolution features are appended to the input. **+TCU(a/b/c)** refers to the use of our Transformers for Convex Upsampling (TCU) with mask size a for the first upsampling step, b for the second step, and c for the last step. Lastly, **-AUG** refers to the additional fine-tuning steps with disabled interpolation-based augmentations.

FlyingThings3D We first investigate the performance for several of our proposed models on the FlyingThings3D [19] test data, after being trained on C+T. The results hereof are shown in Figure 5. From this figure we observe that all our proposed changes result in improvements over the previous model that did not have the change. This is as expected, as we do not expect strong relations between each of our proposed changes, as each of our proposed changes aims at providing improvements in a different way. While these results are good, it is also important to consider situations where this might not be the case. This result is calculated for the test data of the dataset that the model was trained on, so there is no measure of cross-dataset generalization here. For cross-dataset generalization performance, we consider the performance on Sintel Clean [2], which is done next.

Sintel Clean We create the same graph as before, but now for the Sintel Clean [2] dataset. The results hereof are shown in Figure 5. In general, similar patterns as for FlyingThings3D [19] are observed. However, the gap between with and without (-AUG) is no longer as clear as before.

Statistic	Bucket	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
number of samples		290,817	11,241	12,280	19,340	11,034	6,662	4,896	2,985	2,190	1,441	1,018	721	558	347	281	168	196	111	434
samples (percentage)		79.3%	3.1%	3.4%	5.3%	3.0%	1.8%	1.3%	0.8%	0.6%	0.4%	0.3%	0.2%	0.2%	0.1%	0.08%	0.05%	0.05%	0.03%	0.12%
samples (reverse cumulative percentage)		100%	20.7%	17.6%	14.3%	9.0%	6.0%	4.2%	2.8%	2.0%	1.4%	1.0%	0.8%	0.6%	0.4%	0.3%	0.2%	0.2%	0.15%	0.12%
contribution to error (percentage)		44.0%	4.1%	5.1%	10.7%	8.6%	5.9%	5.0%	3.8%	3.1%	2.1%	1.6%	1.3%	0.9%	0.7%	0.5%	0.3%	0.5%	0.3%	1.6%
contribution to error (reverse cumulative percentage)		100%	56.1%	52%	46.9%	36.2%	27.6%	21.7%	16.7%	12.9%	9.8%	7.7%	6.1%	4.8%	3.9%	3.2%	2.7%	2.4%	1.9%	1.6%

Table 1. Distribution of samples over each of the buckets in Figure 4 for FlyingThings3D [19].

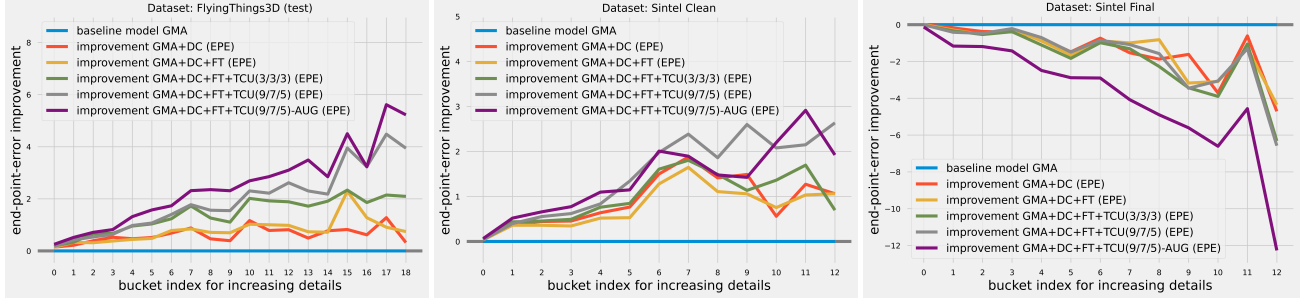


Figure 5. Improvement in end-point-error for increasing amount of detail on FlyingThings3D (test) [19], Sintel Clean (train) and Sintel Final (train) [2], for a series of our proposed models, after being trained on FlyingChairs (train) [4] and FlyingThings3D (test) [19].

Method	FlyingThings3D		Sintel (train)		KITTI-15 (train)		Number of parameters
	Train	Test	Clean	Final	F1-epe	F1-all	
GMA (recomputed)	10.34	3.07	1.31	2.75	4.48	16.86	443K
+DC	9.38	2.84	1.23	2.78	4.43	16.89	443K
+DC+FT	9.53	2.86	1.24	2.79	4.55	16.92	743K
+DC+TCU(3/3/3)+FT	9.51	2.73	1.22	2.83	4.52	16.80	695K
+DC+TCU(9/7/5)+FT	9.24	2.75	1.21	2.80	4.36	16.26	700K
+DC+TCU(9/7/5)+FT-aug	8.97	2.61	1.18	3.01	4.50	16.64	700K

Table 2. Training and generalization performance for our different convex upsampling approaches. Model names are explained in Section 5.2.

Possibly, interpolation artifacts on edges is a good thing for cross-dataset generalization. Reason for this could be that interpolation on edges results in the ‘safe choice’ for a model as it provides values around the mean, which then on average provides similar performance as sharp edges that are sometimes wrong, when evaluated on the end-point-error.

An interesting result from this is that increasing the mask size for TCU from (3/3/3) to (9/7/5) again provides clear improvements. This, together with the same result for FlyingThings3D [19], provides empirical evidence for our hypothesis that a larger mask size can makes more convex solutions possible, which in turn improves performance.

Sintel Final For Sintel Final [2] there exist some important differences. Mainly, Sintel Final contains many strong blurring-based effects in an attempt to mimic real-world camera effects such as motion blur. This introduces a very important aspect; aligning flow with image edges is not a good thing. For Sintel Final, we would instead like to have a model that learns where the actual object edges are based on an image that contains motion blur. To inspect the performance, the same graph as before is generated, but now for Sintel Final [2]. The results hereof can be found in Figure 5. As expected, every step we take towards better aligning

flow with objects edges, degrades the models performance for this dataset. Interestingly, removing the bilinear interpolation artifacts from the model output (-AUG) causes a steep decrease in model performance. Clearly, bilinear interpolation artifacts provide an advantage here. This again confirms our earlier idea that possibly bilinear interpolation artifacts on edges are a ‘safe choice’ when evaluated with the end-point-error, as the values are around the mean. If this is indeed the case, it makes sense that these artifacts help for strong cross-dataset generalization where edges do not align with flow.

Overall, our GMA+DC+FT+TCU(9/7/5) model sets a strong new baseline on all datasets, except Sintel Final. This is shown in Table 2. While cross-dataset generalization is an important aspect of optical flow models, we do not believe it to be realistic to build a model that generalizes to such impactful motion-blur artifacts that are not at all in the training data. It is important to note that generalization to KITTI-15 [21] is good, even though it consists of actual real-world images, taken by a camera.

General Comparison results are reported in Table 3 for various settings (*e.g.* C+T, C+T+S+K+H). While the C+T setting can be seen as a good measure of cross-dataset generalization, training with the training data of specific datasets is also considered interesting. Therefore, we integrate our approach on GMA [14] for the C+T+S+K+H setting. As we observe here, our method can provide an improvement on Sintel Final as well when the training data also contains these blurring artifacts, as is the case for this setting.

Training Data	Method	Sintel (train)		KITTI-15 (train)		Sintel (test)		KITTI-15 (test)
		Clean	Final	F1-epe	F1-all	Clean	Final	F1-all
C+T	HD3 [35]	3.84	8.77	13.17	24.0	-	-	-
	PWC-Net [27]	2.55	3.93	10.35	33.7	-	-	-
	LiteFlowNet2 [10]	2.24	3.78	8.97	25.9	-	-	-
	VCN [34]	2.21	3.68	8.36	25.1	-	-	-
	MaskFlowNet [37]	2.25	3.61	-	23.1	-	-	-
	FlowNet2 [11]	2.02	3.54	10.08	30.0	-	-	-
	DICL-Flow [32]	1.94	3.77	8.70	23.6	-	-	-
	RAFT [30]	1.43	2.71	5.04	17.4	-	-	-
	RAFT (recomputed)	1.42	<u>2.69</u>	5.01	17.5	-	-	-
	RAFT+ALL (ours)	<u>1.26</u>	2.74	4.92	<u>17.4</u>	-	-	-
	RAFT+ALL-aug (ours)	1.28	2.93	<u>4.90</u>	17.5	-	-	-
	GMA [14]	1.30	2.74	4.69	17.1	-	-	-
	GMA (recomputed)	1.31	<u>2.75</u>	4.48	16.9	-	-	-
	GMA+ALL (ours)	1.21	2.77	<u>4.47</u>	17.0	-	-	-
	GMA+ALL-aug (ours)	<u>1.18</u>	3.01	4.50	<u>16.6</u>	-	-	-
	SeparableFlow [36]	1.30	2.59	4.60	15.9	-	-	-
	GMFlowNet [38]	1.14	2.71	4.24	15.4	-	-	-
	FlowFormer [9]	1.01	2.40	4.09 [†]	14.7 [†]	-	-	-
	FlowFormer (recomputed)	0.94 [†]	<u>2.33[†]</u>	<u>4.24[†]</u>	<u>14.9[†]</u>	-	-	-
	FlowFormer+ALL (ours)	0.90[†]	2.34 [†]	4.57 [†]	15.4 [†]	-	-	-
	FlowFormer+ALL-aug (ours)	0.91 [†]	2.37 [†]	4.52 [†]	15.1 [†]	-	-	-
C+T+S+K+H	LiteFlowNet2 [10]	(1.30)	(1.62)	(1.47)	(4.8)	3.48	4.69	7.74
	PWC-Net+ [28]	(1.71)	(2.34)	(1.50)	(5.3)	3.45	4.60	7.72
	VCN [34]	(1.66)	(2.24)	(1.16)	(4.1)	2.81	4.40	6.30
	RAFT [30]	(0.76)	(1.22)	(0.63)	(1.5)	1.61*	2.86*	5.10
	GMA [14]	-	-	-	-	1.39*	2.47*	5.15
	GMA (recomputed)	(0.63)	(1.05)	(0.58)	(1.3)	-	-	-
	GMA+ALL (ours)	(0.58)	(0.97)	(0.62)	(1.4)	1.45*	2.44*	-
	GMA+ALL-aug (ours)	(0.55)	(0.90)	(0.58)	(1.3)	1.44*	2.47*	5.03

Table 3. General comparison of our proposed models against other works. Here, ALL refers to our ‘+DC+TCU(9/7/5)+FT’ setting. Furthermore, C+T refers to training on FlyingChairs [4] and then on FlyingThings3D [19]. Next, C+T+S+K+H refers to training on a mix of data from FlyingChairs [4], FlyingThings [19], Sintel [2], KITTI-15 [21], and HD1K [16]. The values in parentheses ‘()’ are calculated on training data that the model was already trained on. *The warm start strategy is used as described by RAFT [30]. [†]The tile technique is used for evaluation as described by FlowFormer [9]. Note that to make training feasible for FlowFormer we do not fine-tune the transformer models, and only use features from scales 1/4 and 1/8 as scale 1/2 cannot be obtained.

6. Discussion

We observe that our -AUG training scheme decreases the presence of these artifacts in the model output, as can be seen in our submission to the Sintel public scoreboard. Possibly, completely removing these artifacts would require longer training without augmentations. However, this will likely cause an overfit to the training data, so we leave a better solution to this for future work.

Overall, we find good results by reconsidering the design of the convex upsampler and the bilinear interpolation on the flow during training. In general, all our methods; +TCU, +DC, +FT, and -AUG achieve a better fit on the training data. This was our initial goal, and therefore we can confirm that our changes have the desired effect. However, generalization is an important aspect of optical flow models. As such, we ask for careful consideration for adopting our methods when a large generalization gap exists, as in such case our method may not result in improvements. When there is no large generalization gap present as is the case for

Sintel Clean, we show in the (C+T) setting that all our proposed changes can provide improvements on a wide variety of models such as RAFT, GMA, and FlowFormer. We expect similar improvements for other models that currently use the original convex upsampling by factor 8.

Lastly, it is important to consider the accuracy of the edges in the training data. For example, KITTI-15 [21] has its flow maps constructed from sensory data, so its exact accuracy on minor details can possibly be off. Sintel Clean is considered a strong benchmark as the flow map is constructed with 100% certainty, and the images form a good representation of the high-detail edges. Possibly, Spring [20] would be a good evaluation metric, but unfortunately at the time of this work, this dataset has not yet been released.

References

- [1] John Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. *Advances in neural information processing systems*, 2, 1989.

- [2] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael 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*, pages 611–625. Springer, 2012.
- [3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [4] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.
- [5] Kuniyuki Fukushima. Cognitron: A self-organizing multilayered neural network. *Biological cybernetics*, 20(3-4):121–136, 1975.
- [6] Rafael C Gonzalez. *Digital image processing*. Pearson education india, 2009.
- [7] Ali Hassani, Steven Walton, Jiachen Li, Shen Li, and Humphrey Shi. Neighborhood attention transformer. 2022.
- [8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [9] Zhaoyang Huang, Xiaoyu Shi, Chao Zhang, Qiang Wang, Ka Chun Cheung, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Flowformer: A transformer architecture for optical flow. *arXiv preprint arXiv:2203.16194*, 2022.
- [10] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. A lightweight optical flow cnn—revisiting data fidelity and regularization. *IEEE transactions on pattern analysis and machine intelligence*, 43(8):2555–2569, 2020.
- [11] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017.
- [12] Azin Jahedi, Maximilian Luz, Lukas Mehl, Marc Rivinius, and Andrés Bruhn. High resolution multi-scale raft (robust vision challenge 2022). *arXiv preprint arXiv:2210.16900*, 2022.
- [13] Azin Jahedi, Lukas Mehl, Marc Rivinius, and Andrés Bruhn. Multi-scale raft: Combining hierarchical concepts for learning-based optical flow estimation. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 1236–1240. IEEE, 2022.
- [14] Shihao Jiang, Dylan Campbell, Yao Lu, Hongdong Li, and Richard Hartley. Learning to estimate hidden motions with global motion aggregation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9772–9781, 2021.
- [15] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.
- [16] Daniel Kondermann, Rahul Nair, Katrin Honauer, Karsten Krispin, Jonas Andrulis, Alexander Brock, Burkhard Gusefeld, Mohsen Rahimimoghaddam, Sabine Hofmann, Claus 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*, pages 19–28, 2016.
- [17] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.
- [18] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.
- [19] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas 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*, pages 4040–4048, 2016.
- [20] Lukas Mehl, Jenny Schmalfluss, Azin Jahedi, Yaroslava Naliwayko, and Andrés Bruhn. Spring: A high-resolution high-detail dataset and benchmark for scene flow, optical flow and stereo, 2023.
- [21] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3061–3070, 2015.
- [22] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4161–4170, 2017.
- [23] Edgar Riba, Dmytro Mishkin, Daniel Ponsa, Ethan Rublee, and Gary Bradski. Kornia: an open source differentiable computer vision library for pytorch. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3674–3683, 2020.
- [24] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016.
- [25] Xiaoyu Shi, Zhaoyang Huang, Dasong Li, Manyuan Zhang, Ka Chun Cheung, Simon See, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Flowformer++: Masked cost volume autoencoding for pretraining optical flow estimation. *arXiv preprint arXiv:2303.01237*, 2023.
- [26] Xiuchao Sui, Shaohua Li, Xue Geng, Yan Wu, Xinxing Xu, Yong Liu, Rick Goh, and Hongyuan Zhu. Craft: Cross-attentional flow transformer for robust optical flow. In *Pro-*

ceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 17602–17611, 2022.

- [27] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan 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*, pages 8934–8943, 2018.
- [28] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan 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):1408–1423, 2019.
- [29] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022.
- [30] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020.
- [31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [32] Jianyuan Wang, Yiran Zhong, Yuchao Dai, Kaihao Zhang, Pan Ji, and Hongdong Li. Displacement-invariant matching cost learning for accurate optical flow estimation. *Advances in Neural Information Processing Systems*, 33, 2020.
- [33] Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezaatoughi, and Dacheng Tao. Gmflow: Learning optical flow via global matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8121–8130, 2022.
- [34] Gengshan Yang and Deva Ramanan. Volumetric correspondence networks for optical flow. *Advances in neural information processing systems*, 32, 2019.
- [35] Zhichao Yin, Trevor Darrell, and Fisher Yu. Hierarchical discrete distribution decomposition for match density estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6044–6053, 2019.
- [36] Feihu Zhang, Oliver J Woodford, Victor Adrian Prisacariu, and Philip HS Torr. Separable flow: Learning motion cost volumes for optical flow estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10807–10817, 2021.
- [37] Shengyu Zhao, Yilun Sheng, Yue Dong, Eric I Chang, Yan Xu, et al. Maskflownet: Asymmetric feature matching with learnable occlusion mask. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6278–6287, 2020.
- [38] Shiyu Zhao, Long Zhao, Zhixing Zhang, Enyu Zhou, and Dimitris Metaxas. Global matching with overlapping attention for optical flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17592–17601, 2022.