Efficient All-Pairs Correlation Volume Sampling for Optical Flow Estimation

Karlis Martins Briedis[1,2] Markus Gross[1,2] Christopher Schroers[1]

[1]DisneyResearch|Studios [2]ETH Zürich

Abstract

Recent optical flow estimation methods often employ local cost sampling from a dense all-pairs correlation volume. This results in quadratic computational and memory complexity in the number of pixels. Although an alternative memory-efficient implementation with on-demand cost computation exists, this is significantly slower in practice and therefore many prior methods process images at downsampled resolutions, missing fine-grained details.

To address this, we propose an algorithm for both memory and compute-efficient implementation of the all-pairs correlation volume sampling, still matching the exact mathematical operator as defined by RAFT. Our approach outperforms on-demand sampling by up to 92% while maintaining equally low memory usage, and performs at least on par with the default implementation with up to 99% lower memory usage. As cost sampling makes up a significant portion of the overall runtime, this can translate to up to 63% savings for the total end-to-end model inference on high-resolution inputs. Our evaluation of existing methods includes an 8K ultra-high-resolution dataset and an inference-time extension of the SEA-RAFT method. With this, we achieve state-of-the-art results at high resolutions both in accuracy and runtime.

1. Introduction

Optical flow estimation is a classical low-level computer vision problem that involves estimating dense correspondences between video frames. It has found applications in many downstream video tasks, including action recognition [26], video compression [1], video inpainting [15, 35], and frame interpolation [21]. Many of these tasks are used to process ultra-high-resolution (UHR) content, such as in movie post-production, requiring high-quality flows at their original resolution.

The vast majority of optical flow estimation methods use cost matching between two input images [25, 34], with recent methods adopting dense all-pairs correlation volume sampling based on *RAFT* [28]. For correlation volume sampling, *RAFT* either pre-computes the full 4D volume or uses a memory-efficient *on-demand* cost sampling with a custom *CUDA* implementation. Both options have been adopted by many subsequent works [10–12, 20]. However, existing implementations encounter issues with high-resolution inputs. The full volume computation complexity grows quadratically with respect to the number of pixels, making it prohibitive to store at high resolutions. In contrast, the *on-demand* sampling achieves memory reduction at the expense of inefficient computations thus resulting in worse runtime performance. Due to this trade-off, several recent methods have been proposed to avoid sampling from the full volume [32, 39], or removing it [14, 30], but typically do so at the expense of estimation accuracy.

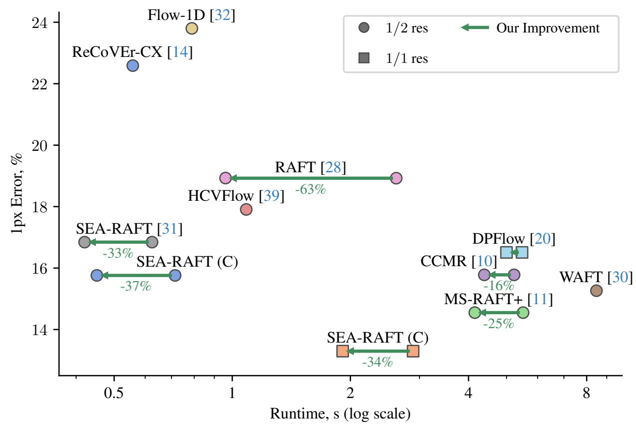

Figure 1. Comparison of the top-performing optical methods on an 8K ultra-high-resolution dataset. We plot the lowest 1px error and the respective runtime for each method across all inference resolutions. Additionally, we show the runtime improvement achieved by using our correlation sampling algorithm in *RAFT*-based methods with green arrows.

In this work, we propose a novel all-pairs correlation volume sampling algorithm, based on computation and sampling of a partial block sparse cost volume. By combining the strengths of both previous approaches, it eliminates the need for approximations or trade-off between speed and memory, and can be cheaply used even at very high input resolutions. The algorithm design is motivated by our analysis of the typical sampling patterns of the full correlation volume in practical applications, where we observe that only a small, regular part of the volume is sampled. This enables us to only compute the necessary subsections of the volume efficiently.

In isolation, the operation performs at least on par with the default implementation, while having up to 99% lower memory usage, and it outperforms *on-demand* sampling by up to 92% while at equally low memory usage. When used in *RAFT* as a drop-in replacement, it reduces the total end-to-end runtime by up to 67% compared to existing methods that are feasible to run with modern hardware. When applied to *SEA-RAFT* [31], which is already designed for efficiency, it provides approximately 33% runtime reduction.

Additionally, we generate a realistic *8K* optical flow dataset based on the BLENDER movie CHARGE and use it to evaluate existing optical flow methods at ultra-high-resolutions. Observing that estimating optical flow at high resolutions improves fine-grained details at the expense of not capturing large displacements, we propose a test-time extension of the *SEA-RAFT* method to perform cascaded inference. Without any additional training of the model, it allows reducing the endpoint error for large motion. Combined with our efficient correlation sampler, we achieve state-of-the-art results at the Pareto front of accuracy and runtime for *8K* flow estimation (Figure 1).

2. Related Work

Optical Flow Estimation. Traditional optical flow methods have used variants of local and global cost volumes. Here we only cover the methods most closely related to our work and refer to the survey by Zhai *et al.* [37] for a more complete overview.

FlowNetC [5] and *PWC-Net* [25] reintroduce the classical concept of cost volume computation for deep learning applications, but process a flattened local cost between source and warped target images. Similarly, several other methods employ cost volume processing but limit processing to the local neighborhood [9, 36].

RAFT [28] revisits the construction of an all-pairs correlation volume and combines it with recurrent iterative refinements that sample matching costs. It proposes two implementations of 4D correlation volume sampling that remain commonly used to date. This approach has subsequently been adopted by many methods, improving estimation of occluded regions [12], encoding matching costs [8, 22], diffusion-based flow updates[18], and other improvements [16, 17, 23, 24, 27, 38]. *SEA-RAFT* [31] revisits the original *RAFT* and introduces simple extensions to improve the efficiency and quality.

Other methods [10, 11] distribute the flow updates over multiple levels up to $1/2$ resolution. Due to sampling matching costs at high resolutions, the slower *on-demand* sampling has to be used.

Little work has been done on optical flow estimation of high input resolutions. Xu *et al*. [32] decompose optical flow estimation into two 1-dimensional operations, enabling processing up to 4K images and provide qualitative results of 4K flow estimation. Similarly, DIP [40] targets efficient inference at high resolution. The Spring benchmark [19] increases the realism and resolution of optical flow evaluation but is limited to HD resolution (1920 × 1080px). GMFlow [33] considers global motion but consequently becomes prohibitively expensive at high resolutions. Morimitsu *et al*. [20] target high resolution inference via an adaptive flow architecture and concurrently introduce quantitative evaluations on a synthetic dataset up to *8K* resolution.

Several works [6, 13, 32, 39] focus on reducing computational costs and memory usage by proposing architecture changes that avoid sampling the dense correlation volume but do so at the expense of estimation accuracy. More recently, ReCoVEr [14] and WAFT [30] remove cost volumes to avoid their time and memory complexity, and compensate it with use of larger feature extraction backbones.

Unlike prior work, we propose an algorithm to improve the efficiency of all-pairs correlation sampling, which is directly compatible with *RAFT* and its variants with no compromise in quality, and show that cost volumes can be used without their commonly assumed computational and space costs.

Operator Efficiency Improvements. Several prior works have focused on improving the computational efficiency of commonly used deep learning operators. FlashAttention [3, 4] proposes a memory-efficient implementation of attention as introduced in the Transformer [29]. Neighborhood Attention Transformer [7] provides an efficient implementation for local attention computation and proposes a method utilizing the efficient operator. In contrast, we not only propose a more efficient implementation of the all-pairs correlation volume sampling, matching the exact mathematical operator as defined by *RAFT*, but also introduce high-level algorithm improvements to the correlation sampling process.

3. Correlation Volume Sampling Analysis

In this section, we introduce the correlation sampling problem in more detail, present the default implementation, and provide an analysis of access patterns of the full 4D correlation volume, which form the foundation of our algorithm. The terms *cost* and *correlation* volume are used interchangeably.

3.1. Problem and Baseline Implementation

Given D-dimensional features $F^{1,2} \in \mathbb{R}^{H \times W \times D}$ extracted from two images, the visual similarity, or correlation, be-

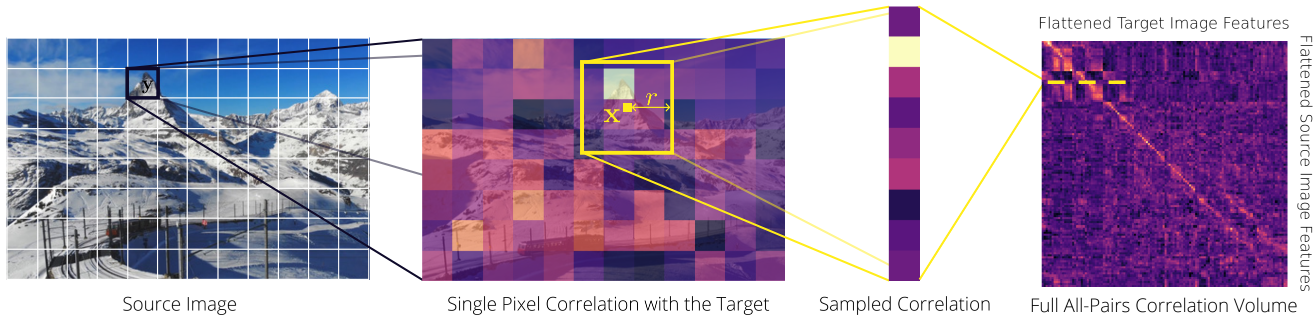

Figure 2. Overview of the correlation volume sampling. Given a map of correlation between the features of a single source pixel and the features of every pixel in another image, bilinear sampling is used to extract local matching costs around a point of interest. When repeated for every source pixel, the costs are stored in a dense all-pairs correlation volume, where each row and column correspond to a source and target pixel, respectively. This is repeated on multiple levels of scale.

tween two pixels is defined as the dot product of their feature vectors.

A lookup is performed by bilinearly sampling at a local grid around an interest pixel $\mathbf{x}$, with sampling positions defined as integer offsets within radius r. More formally, the sampled correlation at source pixel $\mathbf{y}$ is defined as

$$\mathcal{C}_r(\mathbf{y}, \mathbf{x}) = \{\langle F^1_{\mathbf{y}}, F^2_{\mathbf{x}+\mathbf{dx}}\rangle | \mathbf{dx} \in \mathbb{Z} \wedge ||\mathbf{dx}||_\infty \leq r\}. \quad (1)$$

See Fig. 2 for a visualization.

The default implementation of correlation sampling first precomputes a dense 4-dimensional correlation volume $\mathbf{C} \in \mathbb{R}^{H_1 \times W_1 \times H_2 \times W_2}$, where H and W are the height and width of both image features. This is achieved by flattening both images along spatial dimensions, as illustrated on the right side of Fig. 2, and computing the full correlation volume using a single matrix-matrix multiplication

$$\begin{gathered}\mathbf{C} = \bar{F}^1 \cdot \bar{F}^2, \\ \text{where } \bar{F}^1 \in \mathbb{R}^{[H_1 \times W_1] \times D}, \bar{F}^2 \in \mathbb{R}^{[H_2 \times W_2] \times D}.\end{gathered} \quad (2)$$

The output is then reshaped back to 4 dimensions, and bilinear sampling is directly performed on the precomputed $\mathbf{C}$.

In practice, *RAFT* constructs a 4-level pyramid by average pooling the last two dimensions of $\mathbf{C}$ and performs a lookup on the pooled volumes to increase the perceptual window.

Alternatively, a memory-efficient *on-demand* implementation computes the values of Eq. 1 directly for each source frame pixel. This approach reduces computational and memory complexity and does not require storing any intermediate values. However, in practice, it underperforms compared to the baseline due to operations that are not hardware-friendly and the lack of result reuse between iterations. We refer to *RAFT* [28] for more details on the sampling procedure.

3.2. Correlation Volume Access Patterns

The lookup grid for each update is defined over a local neighborhood around the current flow estimate. It ensures that the number of sampled cells per row is limited to the local grid size, *i.e.*, $(2r+1)^2$ elements. During iterative updates, the local neighborhood is shifted but remains close to the previous iteration with a significant overlap of the sampled region. Thus, across all flow update iterations, the total number of sampled columns per single source pixel remains low and does not increase with input size.

To empirically verify this, we run the default *RAFT* implementation on the *Sintel* [2] optical flow training dataset, while tracking which correlation volume cells are sampled. In Fig. 3[a], we visualize the sampled cells for a single image over all update iterations, represented as a $[H_1 \times W_1] \times [H_2 \times W_2]$ volume, where each row shows all matches of a single source pixel. On average, over the whole dataset, only 1.6% of the cells are being sampled. This suggests that we can build a more efficient algorithm that only computes the necessary elements.

However, keeping track of which values are required and computing them efficiently is challenging. A straightforward way is to track them in a binary mask, but that would consume as much memory as the full correlation volume, thus becoming infeasible, while the computations would be similar to that of *on-demand* sampling.

To improve that, we suggest representing $\mathbf{C}$ in a block sparse format and making decisions on computations per block rather than per pixel. As shown in Fig. 3[b], averaging the number of sampled values per block slightly increases the ratio of cells that need to be computed but remains significantly lower than the full matrix for sufficiently small block sizes. Full numerical results of the sampling patterns are provided in the supplementary material.

Reshaped Cost Volume. As the local sampling grid is defined over a 2D neighborhood, when flattened, the values for each row are scattered over multiple column groups and span over many blocks when averaged (Fig. 3[a,b]). This can be improved by pooling the cost volume in a different layout that groups cells based on 2D patches on both source

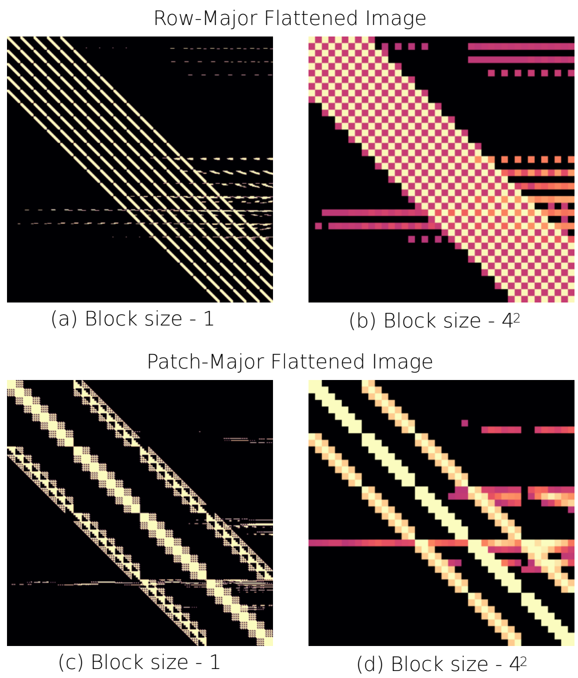

Figure 3. Sampling patterns of a single image over all *RAFT* iterations. Dark regions correspond to cells that have not been sampled while lighter values indicate more sampled values per block.

and target images.

To this end, we reshape the input images into a *patch-major* format, where each block is represented continuously in memory, before computing the correlation volume. As shown in Fig. 3[c,d], the block-aware layout significantly increases sparsity without any computational overhead.

4. Efficient Correlation Volume Sampler

Based on our observations on sampling patterns made in Section 3, we propose an efficient algorithm for all-pairs correlation sampling.

An overview of the algorithm is shown in Fig 4. Inputs are pre-processed once per given a pair of images and on every iteration of *RAFT*-based flow updates (typically $4-32$ iterations) we perform three main steps: *a*) determining regions of the volume that will get sampled and setting the computation mask; *b*) computing selected blocks with efficient tiled matrix-matrix multiplications; *c*) sampling computed blocks.

First, we describe a high-level algorithm, where computation mask and block sparse all-pairs correlation volume is stored explicitly. Then, we present a fused implementation that avoids building the computation mask and stores one block at a time, achieving linear $\mathcal{O}(n)$ time and space complexity in the number of input pixels.

We only cover the single-level case, which is extended to multi-level correlation volumes by computing every level independently, taking average-pooled target features F^2.

More implementation details, pseudocode, and complexity derivation are provided in the supplementary material.

4.1. Input Preprocessing

To minimize the number of blocks that need to be computed, we store images in a *patch-major* format as described in Sec. 3.2 with block height B. To simplify the algorithm implementation, we only consider rectangular blocks and use blocks of the same size across all steps.

At first, we pad inputs to a multiple of B, and split the image into B^2-sized tiles. Each tile is then independently flattened in *row-major* order followed by all tiles being flattened in *row-major* order. This is visualized in Fig. 4 with different-colored arrows. The flattened image is then stored in a contiguous memory block.

4.2. High-Level Implementation

Setting Computation Mask. First, we build a mask of which cell blocks in the cost volume need to be computed, such that all necessary values can be sampled. To this end, we take the convex integer grid positions for the original problem, as defined in Eq. 1, and then floor divide them with the chosen block size B to obtain the cell indices in the block mask. We then perform the reverse operation of grid sampling - *i.e.* scattering - to set these cells as 1 in the mask, while the rest is initialized with 0. By performing the reverse operation of the sampling step, we ensure that all sampled locations have been set to be computed.

Sparse Correlation Volume Computation. The mask is used to compute all blocks of the dense correlation volume that correspond to the non-zero entries of the mask, replacing the dense correlation volume computation defined in Eq. 2.

The computation is performed by a sampled block sparse matrix-matrix multiplication, where only non-zero blocks are stored in memory. As each block is produced by computing the product of two $B^4 \times D$ matrices, it is well optimized in hardware and thus can be computed efficiency.

Sparse Volume Sampling. In order to sample the sparse correlation volume, for each looked up cost value, the block index is computed as when setting the computation mask. Then, we gather the block index in the memory, and compute the coordinates relative to this block. Finally, the block is locally sampled similar to the baseline.

4.3. Fused Implementation

All steps of the algorithm, described in Section 4.2, can be implemented in a fused *CUDA* kernel, where each row of blocks of the dense correlation volume is serially computed and directly sampled in a single *CUDA* thread block, each

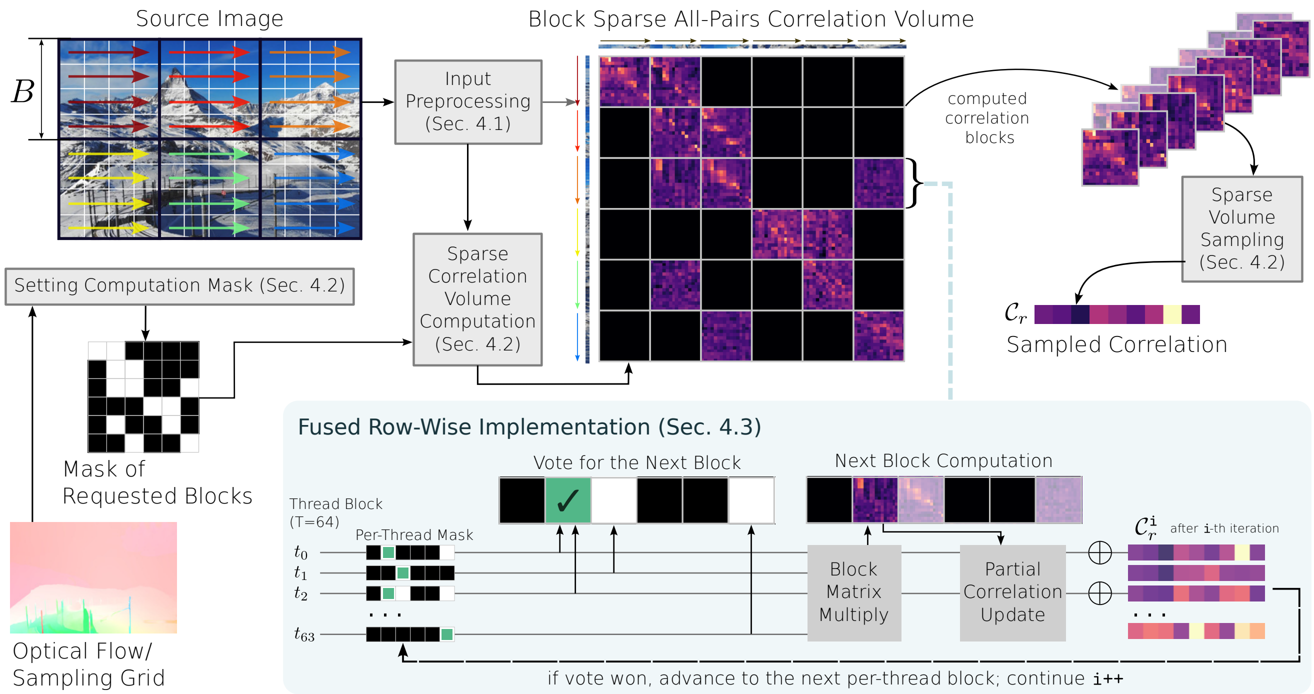

Figure 4. Algorithm overview. It consists of input preprocessing and 3 steps per iteration: *a*) determining blocks that need to be computed; *b*) computing selected blocks with block sparse matrix-matrix multiplication; *c*) sampling computed blocks.

thread sampling from one row, *i.e.* computing correlation for one source pixel. This is visualized in the bottom part of Figure 4. As all rows of blocks are independent, a thread block can perform all steps of the algorithm without writing any of the intermediate results to the global memory, but instead storing them in the shared memory and registers.

Implicit Computation Mask. To avoid storing, setting, and iterating over the full row of computation mask, we build it implicitly through inter-thread voting of the next non-zero cell.

First, similar to setting an explicit computation mask, each thread computes indices of all blocks it needs to sample from. For commonly used parameters, it is no more than 9 blocks, and we refer to the complexity analysis in the supplementary for derivation. As the local grid is well-structured, we can obtain their indices in a strictly increasing order and store them in a register array.

Next, all threads *vote* for the next smallest block to be computed within the thread block. The vote is implemented via an atomic minimum operation to a scalar in the shared memory, accessible to all threads in a block. After the vote, all threads fetch the index of the block that needs to be computed, and advance their local pointers to the next smallest block if their vote matched the outcome.

Block Computation and Sampling. Having agreed on the next tile of the block sparse correlation volume, all threads fetch input values and jointly compute the correlation values (*Block Matrix Multiply* in Fig. 4), similar to how it is done in general matrix multiplications.

After computing the correlation tile, threads exchange their register values through shared memory, such that each thread stores all values corresponding to their source pixel. These registers are directly sampled to obtain partial sampled correlation values, adding residual to a global memory buffer (*Partial Correlation Update* in Fig. 4). After iterating over the whole row of blocks, the output buffer contains the correct and exact correlation values.

5. Isolated Sampling Evaluation

To evaluate the proposed algorithm, we first conduct experiments in isolation by only considering the all-pairs correlation sampling.

We run all methods on the *final* pass of the *Sintel* [2] benchmark's train set consisting of 1041 samples. The isolated tests are performed by extracting the intermediate query centroids with *RAFT* at ⅛ resolution and using randomly-generated feature vectors. For different resolutions, we upsample or downsample the input image. Unless otherwise noted, the experiments are run using *PyTorch* `2.2.2`, *CUDA* `12.2` and *NVIDIA GH200* chip, equipped with $576GB$ coherent memory and selected to perform measurements at very high resolutions even for memory-intensive methods. Throughout the experiments, we set the block height $B = 8$. The fused kernel was implemented

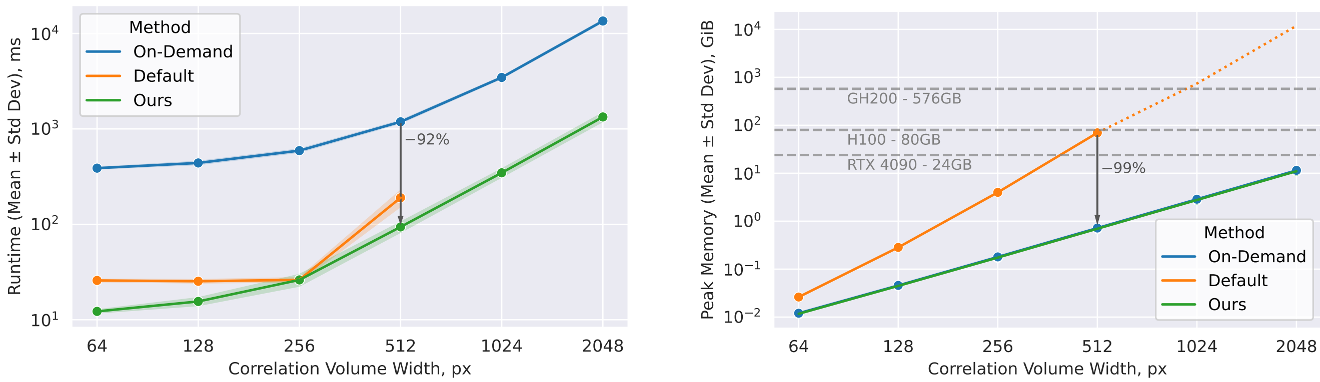

Figure 5. Runtime and peak memory consumption depending on the full correlation volume width. Standard deviation is displayed as shaded area, and we show memory capacity of different hardware as dotted lines.

in *NVIDIA CUTLASS CuTe* DSL, targeted for Ampere microarchitecture using warp-level MMA tensor cores with *bfloat16* inputs and *float32* accumulators.

The correctness was verified with unit tests and observing the endpoint error when used in *RAFT*. It achieves near-zero 0.03% endpoint error difference compared to the official implementation.

5.1. Isolated Correlation Sampling Results

We measure the runtime and the peak memory consumption, as reported by *PyTorch*, by running each operation for all dataset image pairs and report the mean and standard deviation over all sample points. As we observe only negligible variance between different runs with the same inputs, we measure a single run per image.

The default setting uses 2048×896 input image resolution, with the correlation volume size of $(256 \times 112)^2$, 256 feature channels and 32 flow update iterations, matching the official *RAFT* implementation.

5.1.1. Image Resolution

Primarily, we investigate the impact of the image resolution on the correlation computation and report the results in Fig. 5. As images are scaled uniformly, increasing the input width by $2\times$, increases the number of pixels by $4\times$ and the size of the dense correlation volume by $16\times$.

It can be observed that the default implementation has a quadratic memory increase and at 1024×448 resolution already requires $719GB$ to store the dense correlation volume, becoming prohibitively large to compute. On the other hand, *on-demand* sampling maintains low memory usage but significantly underperforms in runtime.

Our method achieves linear increase of both runtime and memory in the number of input pixels. Compared to prior methods, with fixed memory usage, it achieves more than 90% reduction in runtime, or, at fixed runtime, it achieves up to 99% reduction in memory usage.

The exact measurements are provided in the supplementary material.

5.1.2. Other variables

Additionally, we investigate the impact of the number of iterations, input feature dimensionality, as well as hardware, on the runtime and memory of correlation sampling and provide results in the supplementary document.

Similar to the results described in Section 5.1.1, they show favorable runtime and memory trade-off, and all considered GPU models show results consistent with those shown in Figure 5.

6. Ultra-High-Resolution Evaluation

To perform zero-shot evaluation and benchmarking of optical flow estimation methods on UHR inputs, we render several sequences from the BLENDER movie CHARGE with rendered ground truth displacements. We then propose an inference extension to the *SEA-RAFT* optical flow estimation method and perform extensive evaluation of the existing methods. We perform these tests with an *NVIDIA A100 80GB* GPU.

6.1. Dataset

We follow the commonly used benchmarking approach [2, 19] of generating frames and motion vectors from publicly accessible computer-generated movies. We choose BLENDER movie CHARGE as a recent photo-realistic movie that is not used in existing benchmarks. In total, we generate 335 frames at 8192×3432px resolution with super-resolved rendered ground at $16K$ resolution, following Mehl *et al.* [19]. Due to computational reasons, we only consider forward flow. The dataset consists of 332 evaluation pairs and tests for prediction of $9.3B$ pixels. An example of from the dataset can be seen in Figure 6.

Details on the dataset generation are provided in the supplementary material.

6.2. Cascaded Inference

Our initial experiments indicated that evaluation at high resolutions degrade the performance of estimating large dis-

	Best-Accuracy Input Width	1px error % ↓	EPE px ↓	LM - 1px error % ↓	LM-EPE px ↓	Best Runtime, s	Without Ours, s	Our Improvement
GMFlow [33]	1024 - $^1/_8$	43.9	2.95	94.3	24.46	**0.09**	n/a	
PWC-Net [25]	4096 - $^1/_2$	24.4	3.26	61.4	57.14	<u>0.21</u>	n/a	
Flow-1D [32]	4096 - $^1/_2$	23.8	2.23	71.3	31.58	0.79	n/a	
DIP [40]	8192 - $^1/_1$	19.8	4.00	51.2	85.62	11.57	n/a	
FlowFormer [8]	8192 - $^1/_1$	16.9	3.22	56.4	58.16	16.57	n/a	
FlowFormer++ [22]	8192 - $^1/_1$	16.8	3.41	55.3	55.70	16.68	n/a	
SCV [13]	4096 - $^1/_2$	19.1	2.83	46.1	46.42	14.66	n/a	
HCVFlow [39]	4096 - $^1/_2$	17.9	2.08	54.4	32.92	1.09	n/a	
ReCoVEr-CX [14]	4096 - $^1/_2$	22.6	3.01	94.6	58.71	0.56	n/a	
WAFT [30]	4096 - $^1/_2$	15.3	2.74	62.6	65.14	8.47	n/a	
MS-RAFT+ [11]	4096 - $^1/_2$	<u>14.5</u>	<u>1.92</u>	34.6	32.03	4.15	5.51	-25%
RAFT [28]	4096 - $^1/_2$	18.9	5.90	49.8	36.41	0.96	2.62	-63%
CCMR [10]	4096 - $^1/_2$	15.8	2.16	39.7	42.86	4.39	5.23	-16%
DPFlow [20]	8192 - $^1/_1$	16.5	<u>1.92</u>	<u>34.4</u>	**18.03**	5.00	5.48	-9%
SEA-RAFT [31]	4096 - $^1/_2$	16.8	4.94	39.8	39.83	0.42	0.62	-33%
SEA-RAFT (Cascaded)	4096 - $^1/_2$	15.8	**1.90**	36.8	<u>18.58</u>	0.45	0.72	-37%
	8192 - $^1/_1$	**13.3**	2.70	**31.6**	21.53	1.91	2.89	-34%

Table 1. Quantitative evaluation of optical flow estimation methods on a $8K$ dataset. For each method, we list the resolution that can be run with $80GB$ of GPU memory and obtains the best 1px error. For a full comparison with other methods, we also list our $^1/_2$ results. We report the 1px outlier rate, endpoint-error (EPE), both metrics for pixels with large motion (LM, magnitude over 128px) and the best runtime across all variants with and without our improvements. We highlight the best (in **bold**) and second-best (<u>underlined</u>) method.

placements. To mitigate it, we propose a simple cascaded test-time extension of *SEA-RAFT* that requires no additional training.

Before applying any iterative flow updates, we recursively initialize flows as a lower resolution estimate. More formally, whenever the minimum input dimension is more than $800px$, we bilinearly downscale inputs to $^1/_4$ resolution, estimate the flow, and initialize the flows with $^1/_2$ downscaled outputs (note that flows are updated at $^1/_8$ resolution).

It is akin to a multi-resolution version of the *warm-start* initialization used in *RAFT*, and unlike *MS-RAFT+* [11], it does not require training of multiple resolution modules.

6.3. Evaluation Results

We evaluate several methods on different levels of processing scale by bilinearly downsampling inputs to $\{^1/_8; ^1/_4; ^1/_2; ^1/_1\}$ of their resolution and bilinearly upscaling the flow outputs.

Flow accuracy. Quantitative end-to-end evaluation results are reported in Table 1. For each method, we list the processing level of scale with the lowest 1 pixel error across all scales the method could be run with $80GB$ of GPU memory, and report the lowest runtime across feasible variants both with and without our improvements.

First, we observe a significant reduction in runtime, achieving more than 30% improvement for the majority of *RAFT*-based methods. Second, our proposed cascaded inference achieves the best 1 pixel error, both overall and for large-motion pixels. Third, the majority of prior methods either cannot be run on the full 8K resolution inputs, or their quality degrades. Overall, we show that with our improvements, cost-volume-based methods can outperform prior efficient methods both in runtime and quality.

Additionally, we investigate the impact of resolution on the prediction quality and provide a qualitative comparison in Figure 6, showing that cascaded high-resolution inference allows estimating large motion while retaining fine-grained details. Full quantitative measurements are provided in the supplementary material.

Runtime and Memory. In Table 2, we show the improvement in runtime and peak memory usage of our method in an end-to-end estimation, depending on the image resolution.

Across all methods and input resolutions, our method achieves similar or better runtime as the default sampling with improvement in memory, and similar memory as the *on-demand* sampling with a significant improvement in runtime.

Additional Datasets. As our performance improvements are agnostic to the input data, we achieve comparable similar results on other optical flow benchmarks, and provide results in the supplementary material.

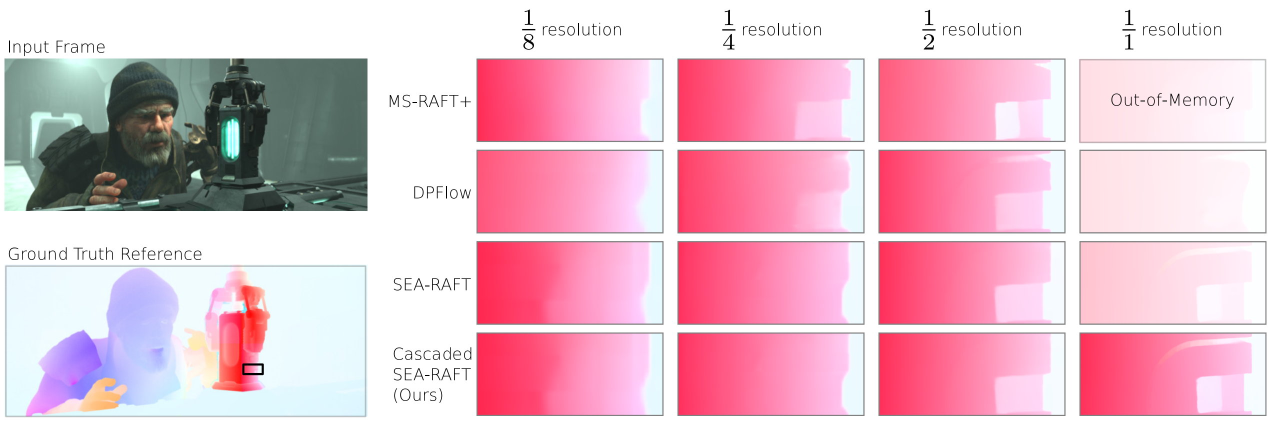

Figure 6. Qualitative comparison across different evaluation scales for MS-RAFT+ [11], DPFlow [20], and SEA-RAFT [31]. Image from Charge by Blender Studio.

		Default		**On-Demand Sampling**		**Ours**	
Method	Input Width	Runtime	Memory	Runtime	Memory	Runtime	Memory
RAFT	1024 - $^1/_8$	0.07 (−16%)	3.42 (+6%)	0.56 (+559%)	3.23 (=)	0.09	3.23
	2048 - $^1/_4$	0.25 (+2%)	8.78 (+115%)	0.89 (+261%)	4.08 (=)	0.25	4.08
	4096 - $^1/_2$	OOM		2.62 (+172%)	7.46 (=)	0.96	7.46
	8192 - $^1/_1$	OOM		10.47 (+206%)	20.96 (=)	3.43	20.96
MS-RAFT+	1024 - $^1/_8$	OOM		0.44 (+43%)	4.42 (−1%)	0.31	4.45
	2048 - $^1/_4$	OOM		1.41 (+34%)	8.69 (−1%)	1.05	8.80
	4096 - $^1/_2$	OOM		5.51 (+33%)	25.78 (−2%)	4.15	26.21
DPFlow	1024 - $^1/_8$	0.14 (−2%)	3.43 (+6%)	0.15 (+3%)	3.24 (=)	0.14	3.24
	2048 - $^1/_4$	0.40 (+12%)	9.53 (+146%)	0.41 (+15%)	3.87 (=)	0.35	3.87
	4096 - $^1/_2$	OOM		1.41 (+10%)	6.57 (=)	1.27	6.57
	8192 - $^1/_1$	OOM		5.48 (+10%)	17.38 (=)	5.00	17.38
SEA-RAFT	1024 - $^1/_8$	0.03 (=)	3.47 (+7%)	0.09 (+200%)	3.24 (=)	0.03	3.25
	2048 - $^1/_4$	0.13 (+28%)	8.86 (+146%)	0.18 (+76%)	3.56 (−1%)	0.11	3.60
	4096 - $^1/_2$	OOM		0.62 (+49%)	5.22 (=)	0.42	5.22
	8192 - $^1/_1$	OOM		2.63 (+48%)	11.82 (=)	1.78	11.82

Table 2. Runtime (s) and peak memory usage (GiB) of the full optical flow method end-to-end evaluation depending on the correlation computation variant at different scales of the inputs. We report the relative difference compared to our method in parentheses. OOM indicates that the method requires more than $80GB$ of memory and fails with an out-of-memory error.

7. Conclusion

In this paper, we propose an efficient all-pairs correlation sampling algorithm, which allows to significantly improve performance of *RAFT*-based methods. First, we analyze the existing approaches for all-pairs correlation volume sampling, their volume sampling patterns in a practical optical flow application and propose an algorithm that utilizes these observations to perform correlation sampling more efficiently. In extensive experiments, we show that our method achieves low memory consumption and runtime when compared to previous solutions. Additionally, we evaluate existing methods on an *8K* resolution dataset and achieve state-of-the-art results on both in accuracy and performance.

In this work, we did not perform extensive low-level fine-tuning, or architecture-specific optimizations for a fully optimized implementation. Additionally, we only consider forward pass optimizations, as training is typically done at lower resolutions due to memory requirements of other parts of the method. However, the same optimizations can be applied to the backward pass, and we leave it to future work to investigate that.

References

[1] Eirikur Agustsson, David Minnen, Nick Johnston, Johannes Balle, Sung Jin Hwang, and George Toderici. Scale-Space Flow for End-to-End Optimized Video Compression. In *Pro-*

ceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8503–8512, 2020. 1

[2] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conf. on Computer Vision (ECCV)*, pages 611–625. Springer-Verlag, 2012. 3, 5, 6

[3] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024. 2

[4] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-Awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 2

[5] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2758–2766, 2015. 2

[6] Risheek Garrepalli, Jisoo Jeong, Rajeswaran C. Ravindran, Jamie Menjay Lin, and Fatih Porikli. DIFT: Dynamic Iterative Field Transforms for Memory Efficient Optical Flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2220–2229, 2023. 2

[7] Ali Hassani, Steven Walton, Jiachen Li, Shen Li, and Humphrey Shi. Neighborhood Attention Transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6185–6194, 2023. 2

[8] 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. In *Computer Vision – ECCV 2022*, pages 668–685. Springer Nature Switzerland, Cham, 2022. 2, 7

[9] Junhwa Hur and Stefan Roth. Iterative Residual Refinement for Joint Optical Flow and Occlusion Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5754–5763, 2019. 2

[10] Azin Jahedi, Maximilian Luz, Marc Rivinius, and Andrés Bruhn. CCMR: High Resolution Optical Flow Estimation via Coarse-To-Fine Context-Guided Motion Reasoning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6899–6908, 2024. 1, 2, 7

[11] Azin Jahedi, Maximilian Luz, Marc Rivinius, Lukas Mehl, and Andrés Bruhn. MS-RAFT+: High Resolution Multi-Scale RAFT. *International Journal of Computer Vision*, 132 (5):1835–1856, 2024. 1, 2, 7, 8

[12] 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. 1, 2

[13] Shihao Jiang, Yao Lu, Hongdong Li, and Richard Hartley. Learning Optical Flow From a Few Matches. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16592–16600, 2021. 2, 7

[14] Simon Kiefhaber, Stefan Roth, and Simone Schaub-Meyer. Removing cost volumes from optical flow estimators. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 79–89, 2025. 1, 2, 7

[15] Dahun Kim, Sanghyun Woo, Joon-Young Lee, and In So Kweon. Deep Video Inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5792–5801, 2019. 1

[16] Ao Luo, Fan Yang, Xin Li, and Shuaicheng Liu. Learning Optical Flow With Kernel Patch Attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8906–8915, 2022. 2

[17] Ao Luo, Fan Yang, Xin Li, Lang Nie, Chunyu Lin, Haoqiang Fan, and Shuaicheng Liu. GAFlow: Incorporating Gaussian Attention into Optical Flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9642–9651, 2023. 2

[18] Ao Luo, Xin Li, Fan Yang, Jiangyu Liu, Haoqiang Fan, and Shuaicheng Liu. FlowDiffuser: Advancing Optical Flow Estimation with Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19167–19176, 2024. 2

[19] Lukas Mehl, Jenny Schmalfuss, Azin Jahedi, Yaroslava Nalivayko, and Andrés Bruhn. Spring: A high-resolution high-detail dataset and benchmark for scene flow, optical flow and stereo. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2, 6

[20] Henrique Morimitsu, Xiaobin Zhu, Roberto M. Cesar, Xiangyang Ji, and Xu-Cheng Yin. DPFlow: Adaptive Optical Flow Estimation with a Dual-Pyramid Framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17810–17820, 2025. 1, 2, 7, 8

[21] Simon Niklaus and Feng Liu. Context-Aware Synthesis for Video Frame Interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1710, 2018. 1

[22] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1599–1610, 2023. 2, 7

[23] Austin Stone, Daniel Maurer, Alper Ayvaci, Anelia Angelova, and Rico Jonschkowski. SMURF: Self-Teaching Multi-Frame Unsupervised RAFT With Full-Image Warping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3887–3896, 2021. 2

[24] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17602–17611, 2022. 2

[25] 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. 1, 2, 7

[26] Shuyang Sun, Zhanghui Kuang, Lu Sheng, Wanli Ouyang, and Wei Zhang. Optical Flow Guided Feature: A Fast and Robust Motion Representation for Video Action Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1390–1399, 2018. 1

[27] Shangkun Sun, Yuanqi Chen, Yu Zhu, Guodong Guo, and Ge Li. SKFlow: Learning Optical Flow with Super Kernels. In *Advances in Neural Information Processing Systems*, 2022. 2

[28] Zachary Teed and Jia Deng. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. In *Computer Vision – ECCV 2020*, pages 402–419, Cham, 2020. Springer International Publishing. 1, 2, 3, 7

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. 2

[30] Yihan Wang and Jia Deng. WAFT: Warping-Alone Field Transforms for Optical Flow, 2025. 1, 2, 7

[31] Yihan Wang, Lahav Lipson, and Jia Deng. SEA-RAFT: Simple, Efficient, Accurate RAFT for Optical Flow. In *Computer Vision – ECCV 2024*, pages 36–54. Springer Nature Switzerland, Cham, 2025. 1, 2, 7, 8

[32] Haofei Xu, Jiaolong Yang, Jianfei Cai, Juyong Zhang, and Xin Tong. High-Resolution Optical Flow From 1D Attention and Correlation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10498–10507, 2021. 1, 2, 7

[33] Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, 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. 2, 7

[34] Jia Xu, Rene Ranftl, and Vladlen Koltun. Accurate Optical Flow via Direct Cost Volume Processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1289–1297, 2017. 1

[35] Rui Xu, Xiaoxiao Li, Bolei Zhou, and Chen Change Loy. Deep Flow-Guided Video Inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3723–3732, 2019. 1

[36] Gengshan Yang and Deva Ramanan. Volumetric Correspondence Networks for Optical Flow. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019. 2

[37] Mingliang Zhai, Xuezhi Xiang, Ning Lv, and Xiangdong Kong. Optical flow and scene flow estimation: A survey. *Pattern Recognition*, 114:107861, 2021. 2

[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. 2

[39] Yang Zhao, Gangwei Xu, and Gang Wu. Hybrid Cost Volume for Memory-Efficient Optical Flow. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 8740–8749, New York, NY, USA, 2024. Association for Computing Machinery. 1, 2, 7

[40] Zihua Zheng, Ni Nie, Zhi Ling, Pengfei Xiong, Jiangyu Liu, Hao Wang, and Jiankun Li. DIP: Deep Inverse Patchmatch for High-Resolution Optical Flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8925–8934, 2022. 2, 7