Dynamic Cost Volumes with Scalable Transformer Architecture for Optical Flow

Vemburaj Yadav, Alain Pagani, Didier Stricker

Augmented Vision, German Research Center for Artificial Intelligence (DFKI)
Kaiserslautern, Germany
{vemburaj.yadav, alain.pagani, didier.stricker}@dfki.de

Abstract

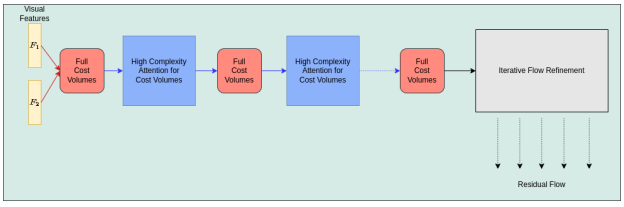
We introduce DCV-Net, a scalable transformer-based architecture for optical flow with dynamic cost volumes. Recently, *FlowFormer* [Huang et al., 2022], which applies transformers on the full 4D cost volumes instead of the visual feature maps, has shown significant improvements in the flow estimation accuracy. The major drawback of *FlowFormer* is its scalability for high-resolution input images, since the the complexity of the attention mechanism on the 4D cost volumes scales to $O(N^4)$, with N being the number of visual feature tokens. We propose a novel architecture where we obtain the *FlowFormer* type enrichment of matching cost representations, but using light-weight attention on the visual feature maps with quadratic ($O(N^2)$) complexity. Firstly, we generate sequential updates to the visual feature representations and, consequently, the cost volumes using lightweight attention layers. Secondly, we interleave this sequence of cost volumes with iterations of flow refinement, thereby modeling the update operator to handle dynamic cost volumes. Our architecture, with two orders of computational complexity lower than that of *FlowFormer*, demonstrates strong cross-domain generalization on the Sintel and KITTI datasets. We outperform *FlowFormer* on the KITTI dataset and achieve highly competitive flow estimation accuracies on the Sintel dataset.

Keywords: Transformer, Dynamic Cost Volumes, Scalability, etc.

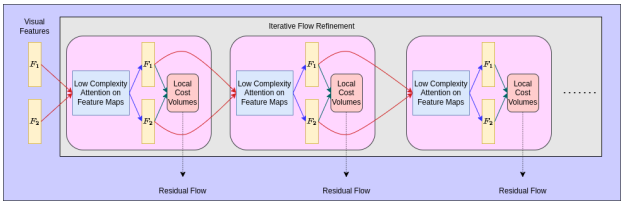
1 Introduction

Optical flow is a crucial concept in computer vision, aimed at estimating the per-pixel motion between frames and providing dense correspondences as valuable information for various downstream video tasks. Although optical flow has been studied since the seminal work of Horn and Schunck [Horn and Schunck, 1981], it remains a challenging problem due to difficulties such as fast-moving objects, occlusions, motion blur, and textureless surfaces.

Recently, optical flow estimation has witnessed a paradigm shift from the traditional optimization-based approach to deep learning-based methods. *RAFT* [Teed and Deng, 2020] makes it plausible to model near-global search space by constructing a $H \times W \times H \times W$ cost volume that measures similarities between all pairs of pixels of the $H \times W$ image pair. This is then complemented by an iterative framework



(a) FlowFormer type attention for Cost Volumes



(b) Our Dynamic Cost Volumes with scalable attention

Figure 1: Comparison of our DCV-Net with FlowFormer a) Static Cost Volume for refinement with Non-scalable attention blocks b) Dynamic Cost Volume for refinemet with Scalable attention blocks.

that retrieves local costs within local windows to perform flow residual regression. The current state-of-the-art methods are all based on such RAFT-style architectures.

The feature representations being obtained from a CNN backbone, it lacks the global context, and thus the matching costs remains unreliable for occluded pixels and pixels undergoing large motions. Several follow-up works like *GMFlow* [Xu et al., 2022] and *GMFlowNet* [Zhao et al., 2022] incorporated self and cross attention layers into the overall architecture to learn global context-aware feature representations. *FlowFormer* [Huang et al., 2022], on the other hand, directly processes the 4D cost volumes with transformer blocks. They adopt an encoder-decoder architecture to first encode the matching costs into a latent cost space (cost memory), and then iteratively query this cost memory (similar to cost volume lookup) to regress the flow.

Although, *FlowFormer* has led to state-of-the-art flow estimation accuracy on several benchmarks, it has a major drawback in terms of its scalability to the resolution of the input image. Cost volume computation and storage in itself is one of the major bottlenecks for high resolution optical flow and real-time inference, since it scales quadratically ($O(N^2)$) with the number of visual tokens N in the visual feature map. Since context aggregation by attention in itself scales quadratically to the number of input tokens, applying transformer blocks directly on the 4D cost volumes induces a complexity of $O(N^4)$. On the other hand, applying attention layers to the feature maps retains the complexity of the attention layers to $O(N^2)$. So, a clear argument could be made to still use this style of feature enhancement instead of the attention on 4D cost volumes, especially for high resolution inputs. We demonstrate in this work that we achieve state-of-the-art performances and very good cross-domain generalization, yet still using the light weight attention on the feature maps instead of the cost volumes.

Another major drawback, not limited to *FlowFormer*, but also to any transformer based architecture for optical flow, is the application of sequential attention blocks for feature enhancement in addition to the sequential nature of the iterative flow refinement. In the RAFT-style architectures, even though a full 4D cost volume is first computed and stored, only local costs within a local window are being looked up for each pixel at any given iteration. The local cost volumes could theoretically be computed on the fly at each iteration, thereby enabling significant reduction in memory requirements. However, these architectures do not consider updating and enriching the matching costs over the refinement iterations, thus limiting the update operator to work only with a one-time computed static cost volume. We derive motivations from these observations and propose an architecture, where the feature maps are contextualized with attention layers sequentially concurrent to the sequential regression of residual flow, thus updating the matching costs over the iterations, and thereby modelimg the cost volume to be dynamic inside the architecture.

In this paper, we propose DCV-Net (Dynamic Cost Volume Network), a scalable transformer-based architecture for optical flow estimation with dynamic cost volumes. We emulate the *FlowFormer* type transformer-based updates to the matching costs, but at the same time enjoying the scalable attribute of the whole architecture for higher resolutions by leveraging the lower computational effort that comes with applying transformers to update the feature representations and computing on-the-fly dynamic local cost volumes, as shown in Fig. 1. We also propose a softmax based matching layer to transform the local matching costs into their corresponding matching distributions. This allows us to seamlessly incorporate a deep transformer architecture inside the refinement stage, while still respecting the RAFT-style optimization for flow regression with weight-tied update operators. We summarize our key contributions as follows:

1. A scalable transformer-based architecture for optical flow with dynamic cost volumes, which at the same time achieves competitive performance on all the benchmarks. Our approach could easily be scaled not only while running with high-resolution images, but also if the whole architecture is required to run with feature maps at a scale higher. In comparison, *FlowFormer* suffers from a poor scalability under both these scenarios.

2. A differentiable softmax-based matching layer to convert the dynamic local matching costs into matching distributions, thereby enabling a seamless integration of deep transformer based feature updates with the iterations of the flow refinement

3. A state-of-the-art cross validation performance on the KITTI dataset, and near-state-of-the-art cross validation performance on the Sintel dataset, while being only trained on synthetic datasets of Flying Chairs and Flying Things.

4. To our knowledge, this is the first attempt to demonstrate an architecture with transformers for optical flow, which is both scalable to higher resolution inputs, but at the same time delivers state-of-the art cross-domain generalization on real world datasets. Most of the previous works lacks in terms of one or more out of these 3 crucial aspects: scalability, generalization and performance.

2 Related Work

Deep Learning for Optical flow: Compared with traditional optimization-based flow estimation methods, data-driven methods directly learn to estimate optical flow from labeled data. Since *FlowNet* [Dosovitskiy et al., 2015], learning optical flow with neural networks has demonstrated superior flow estimation accuracies. Earlier works along these lines directly regress the optical flow by applying convolutional layers on the computed local cost volume. However, these architecture styles poses constraints on the channel dimension of the cost volumes, which restricts the search space to a local range, making it hard to tackle large displacements. Prominent works in these directions are *FlowNet* and *PWC-Net* [Sun et al., 2018], with the former predicting the flow from a local cost volume computed from the visual similarities of downsampled feature maps at a single resolution, while the latter builds hierarchical local cost volumes with warped features and progressively estimates flows from such local costs.

Transformers for Optical Flow: Exploiting the long-range modeling capacity of attention layers [Vaswani et al., 2017], Recent works ([Xu et al., 2022, Zhao et al., 2022, Huang et al., 2022]) employs transformers to further weaken the network-determined bias and learn feature relationships from data. *GMFlowNet* [Zhao et al., 2022] proposed to first contextualize the individual feature maps of both the frames by a sequential application of several self-attention layers, followed by the RAFT-style cost volume computation and iterative flow refinement. *GMFlow* [Xu et al., 2022], on thew other hand enhances the feature maps with alternating applications of self and cross attention layers, thereby providing also the inter-frame context to the feature maps. However, the dense correspondence field is directly regressed from the cost volume using the softmax based differentiable matching layer. The lack of iterative refinement, along with the presence of heavy transformer blocks in *GMFlow* massively degrades the generalization ability of the architecture, which being demonstrated from their inferior cross-validation performance on the KIITI dataset. *FlowFormer* achieves state-of-the-art accuracy by replacing the CNN based backbone for feature extraction with a transformer, and directly applying attentions on the 4D cost volumes.

3 Method

Given a pair of source and target RGB images I_1 and I_2 respectively, optical flow aims at recovering a dense correspondence field (f^1, f^2) for every source pixel (u, v) in I_1, such that it maps to the locations (u', v') in I_2, with $(u', v') = (u + f^1(u, v), v + f^2(u, v))$. Our network design is based on the amalgamation of attention layers into the successful *RAFT* [Teed and Deng, 2020] architecture. Fig. 2 represents an overall visual representation of our network architecture. For completeness, refer to our supplementary material for a brief description of 1) the key components of RAFT architecture and 2) the attention mechanism in transformers.

3.1 Dynamic Cost Volume

The core contribution of our work lies at the heart of modeling the 4D cost volumes to be dynamic inside the architecture, but at the same time maintaining the scalability of the model to the input image resolution. The temporal nature of the cost volumes is inherently suited to our proposed integration of them within the iterative refinement stage of the architecture, which also operates in a temporal fashion. We therefore first present the

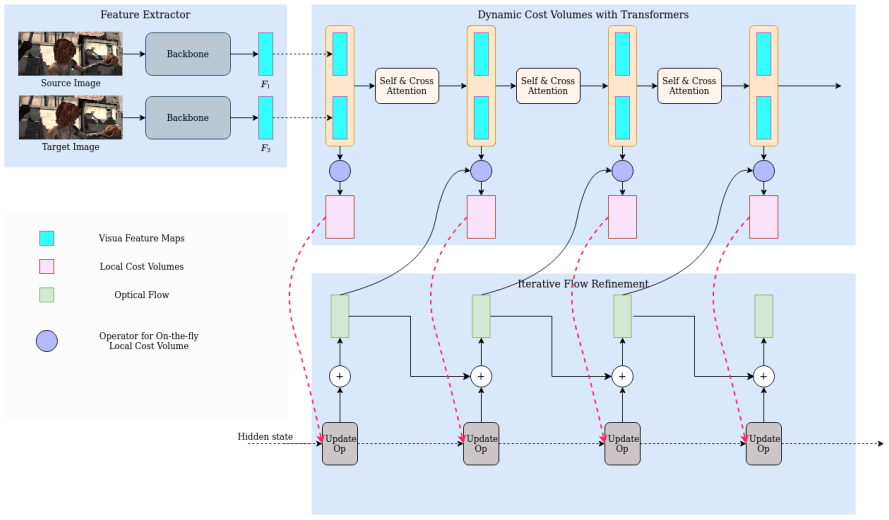


Figure 2: Architecture of DCV-Net. DCV-Net consists of 3 main components: 1) A sequence of visual feature maps generated by layers of a transformer architecture. 2) A sequence of Local Cost Volumes (on-the-fly computed variants of Dynamic Cost Volumes). 3) Iterative Flow refinement stage which recurrently regresses the residual flow using the Dynamic Cost Volumes.

mathematical formulation of our dynamic cost volume with scalable attention. To provide a comprehensive understanding of our proposed approach in comparison to previous works, we introduce two additional formulations. First, we consider an approach similar to *FlowFormer*, where transformer layers are directly applied to the 4D cost volume, resulting in a sequence of cost volumes. Second, we consider approaches similar to *GMFlow* and *GMFlowNet* where transformer layers are applied to the visual feature maps, generating a sequence of feature maps with richer representations. In both scenarios, the cost volume remains static during the refinement stage.

Attention on Cost Volumes: Given the feature maps $\mathbf{F}_1$ and $\mathbf{F}_2$ extracted from a CNN backbone, a 4D cost volume $\mathbf{C}$ is constructed to store similarity scores for potential matches between the frames. Enhancing the matching distribution for a specific source pixel involves considering the distributions of nearby and contextually similar source pixels. This fosters consensus among neighboring matches, resulting in more accurate flow estimation, particularly for occluded pixels. To generate a sequence of cost volumes $\mathbf{C}^{[i]}$ with improved matching costs, the cost volume is passed through a transformer architecture that employs self-attention using all the $H \times W \times H \times W$ tokens. Let M transformer blocks, each parameterized by $\theta^{[i]}$, be used to generate L stages of cost volumes $\mathbf{C}^{[j]}$, where $\mathbf{C}^{[0]}$ is the initial cost volume $\mathbf{C}$. The final cost volume $\mathbf{C}^{[L]}$ would then be looked-up in the Iterative Residual Refinement (IRR) stage to regress the optical flows $\mathbf{f}^{[k]}$ for K iterations. A mapping function p determines the index i of the transformer block used in the j^{th} stage of the architecture, allowing for sharing of transformer blocks across multiple stages, which is advantageous for our formulation with dynamic cost volumes. In a straightforward case of L stages and L transformer blocks, each transformer block is used in a single stage, resulting in the mapping function $p(j) = j$.

$$\begin{aligned} \mathbf{C}^{[j]} &= \text{Transformer}(\mathbf{C}^{[j-1]}; \theta^{p(j)}), \quad j = 1, 2, .., L \\ \mathbf{f}^{[k]} &= \text{IRR}(\mathbf{C}^{[L]}, \mathbf{f}^{[k-1]}), \quad k = 1, 2, .., K \end{aligned} \tag{1}$$

With N as the number of visual feature tokens, attention on the cost volumes with N^2 tokens induces a complexity of $O(N^4)$ both in terms of computational effort and memory requirements. *FlowFormer* tries to mitigate this by patchifying the 4D cost tokens and using a complex latent cost encoder and decoder architecture. On the other hand, our architecture enjoys the simplicity in terms of its integration into a RAFT-style architecture.

Attention on Feature Maps and Static Cost Volume: Here, instead of processing the 4D cost volume with a transformer architecture, the feature maps are sequentially enhanced using transformer blocks with self and cross attention layers. Let $\mathbf{F}_1^{[i]}$, $\mathbf{F}_2^{[i]}$ be the sequence of feature maps generated over L stages using M distinct transformer blocks as before. The feature maps $\mathbf{F}_1^{[L]}$, $\mathbf{F}_2^{[L]}$ from the final stage would then be used to construct the cost volume $\mathbf{C}$, which would then be used inside IRR. Unlike the previous formulation with attention on cost volumes, this formulation presents much more light-weighted architecture and superior scalability due to the quadratic complöexity of the attention blocks being applied on visual feature maps.

$$\begin{aligned}(\mathbf{F}_1^{[j]}, \mathbf{F}_2^{[j]}) &= \text{Transformer}(\mathbf{F}_1^{[j-1]}, \mathbf{F}_2^{[j-1]}; \theta^{p(j)}), \quad j = 1, 2, .., L \\ \mathbf{C} &= f(\mathbf{F_1}^{[L]}, \mathbf{F_2}^{[L]}) \\ \mathbf{f}^{[k]} &= \text{IRR}(\mathbf{C}, \mathbf{f}^{[k-1]}), \quad k = 1, 2, .., K\end{aligned} \tag{2}$$

Attention on Feature Maps and Dynamic Cost Volume: Previous works like *GMFlow* and *GMflowNet* employing the formulation with attention on feature maps lags behind significantly in terms of flow estimation accuracy, in comparison to *FlowFormer* type approaches where directly the cost volumes are contextualized using attention Also, in both both formulations above, cost volume remains static over the iterations of the update operator. We, therefore propose a formulation where we combine the advantages of the both the mentioned formulations, i.e, 1) we maintain the scalability of the architecture by sequentially enhancing the feature maps using attention 2) we simultaneously generate a sequence of cost volumes from the sequence of feature maps, and make this whole sequence of cost volumes dynamic by utilizing it over the iterations of the flow refinement. Unlike the previous 2 formulations where the refinement stage utilizes a fixed pre-computed cost volume, we concurrently update the cost volumes over the iterations of the flow refinement. A sequence of L feature maps $\mathbf{F}_1^{[i]}$, $\mathbf{F}_2^{[i]}$ and their corresponding cost volumes $\mathbf{C}^{[i]}$ are obtained using a module with M transformer blocks. Since, we have L stages of the cost volume and K iterations of refinement, the first L iterations of refinement benefits from the usage of dynamic cost volumes, and for the remaining $K - L$ iterations, the cost volume from the final stage would be utilized. Since only a portion of the full cost volume is being utilized at each refinement iteration, we compute these local cost volumes on-the-fly using the implementation prposed by authors from RAFT. We take advantage of the significant reduction in the memory requirements that this implementation brings for our dynamic cost volumes.

$$\left.\begin{aligned}(\mathbf{F}_1^{[j]}, \mathbf{F}_2^{[j]}) &= \text{Transformer}(\mathbf{F}_1^{[j-1]}, \mathbf{F}_2^{[j-1]}; \theta^{p(j)}) \\ \mathbf{C}^{[j]} &= f(\mathbf{F_1}^{[j]}, \mathbf{F_2}^{[j]}) \\ \mathbf{f}^{[j]} &= \text{IRR}(\mathbf{C}^{[j]}, \mathbf{f}^{[j-1]})\end{aligned}\right\} \text{j=1,2,..,L}$$
$$\mathbf{f}^{[k]} = \text{IRR}(\mathbf{C}^{[L]}, \mathbf{f}^{[k-1]}), \quad k = L+1, L+2, .., K$$

3.2 Cost Volume LookUp based on Matching Probability

The GRU based update operator in the IRR is responsible for regressing the residual flows based on the previous flow estimate, context features, hidden state and local matching costs sampled from the cost volume. The parameters of this update operator are weigh-tied to mimic its functionality as a optimization solver. For static cost volumes, the distribution of the similarity score values encountered by the update operator does not change over the refinement iterations. On the other hand, the distribution of our feature map activations differs from one iteration to the other since they are the output of transformer blocks with non-shared parameters. This leads to unstable training of the parameters of update operator. To mitigate this issue, we propose to transform the sampled local matching costs into their corresponding matching probabilities. We accomplish this by applying *softmax* to the matching costs over the sampled co-ordinate locations. The update operator therefore receives such probability measures as input, irrespective of the distributions of matching costs generated by distinct transformer blocks.

Training Data	Method	Sintel		KITTI	
		Clean	Final	F1-epe	F1-all
C + T	HD3	3.84	8.77	13.17	24.0
	LiteFlowNet	2.48	4.04	10.39	28.5
	PWC-Net	2.55	3.93	10.35	33.7
	LiteFlowNet2	2.24	3.78	8.97	25.9
	S-Flow	1.30	2.59	4.60	15.9
	RAFT	1.43	2.71	5.04	17.4
	FM-RAFT	1.29	2.95	6.80	19.3
	GMA	1.30	2.74	4.69	17.1
	GMFlow	1.08	2.48	-	-
	GMFlowNet	1.14	2.71	4.24	15.4
	CRAFT	1.27	2.79	4.88	17.5
	SKFlow	1.22	2.46	4.47	15.5
	FlowFormer	**0.94**	**2.33**	4.09	14.72
	DCV-Net (Ours)	0.99	2.43	**3.91**	**14.27**

Table 1: Results on Sintel and KITTI datasets. We test the generalization performance after training on FlyingChairs(C) and FlyingThing(T)

4 Experiments

Following previous works, we train our models first on the FlyingChairs [Dosovitskiy et al., 2015] and then on the FlyingThings3D [Mayer et al., 2016]. Please refer to the supplementary material for the implementation details and the training schedules. We then evaluate our model on the training splits of Sintel [Butler et al., 2012] and KITTI [Geiger et al., 2013] to demonstrate cross-domain generalization.

4.1 Quantitative Experiments

Table 1 shows the cross-domain generalization performance of our model in comparison with the recent works from the literature. Although our DCV-Net has two orders of complexity than that of *FlowFormer*, we outperform it on the KITTI dataset, while staying highly competitive on the Sintel dataset. Also, we outperform both *GMFlow* and *GMFlowNet* by a significant margin on all the benchmarks, even though we employ transformer blocks of similar complexity on the visual feature maps.

4.2 Qualitative Experiments

We visualize flows estimated by our DCV-Net and *FlowFormer* for four examples in Fig. 3. We show that the flow estimated by our DCV-Net is similar in quality to that of *FlowFormer*, since we are able to recover high quality motions in overlapping object regions, occluded areas and regions with large displacements, with a scalable model of much lower complexity.

4.3 Ablation Study

We perform a set of ablations to demonstrate the effectiveness of our proposed contributions. We start with *RAFT* as the baseline with a CNN backbone and a static cost volume being decoded by a recurrent decoder to iteratively regress the optical flow. We then gradually alter different components of the baseline with our proposed components and show their significance on the performance of flow estimation.

Static vs Dynamic Cost Volume: We demonstrate the importance of having the cost volumes to be dynamic over the iterations of flow refinement and compare it with the case with the RAFT-style static cost volume. It is

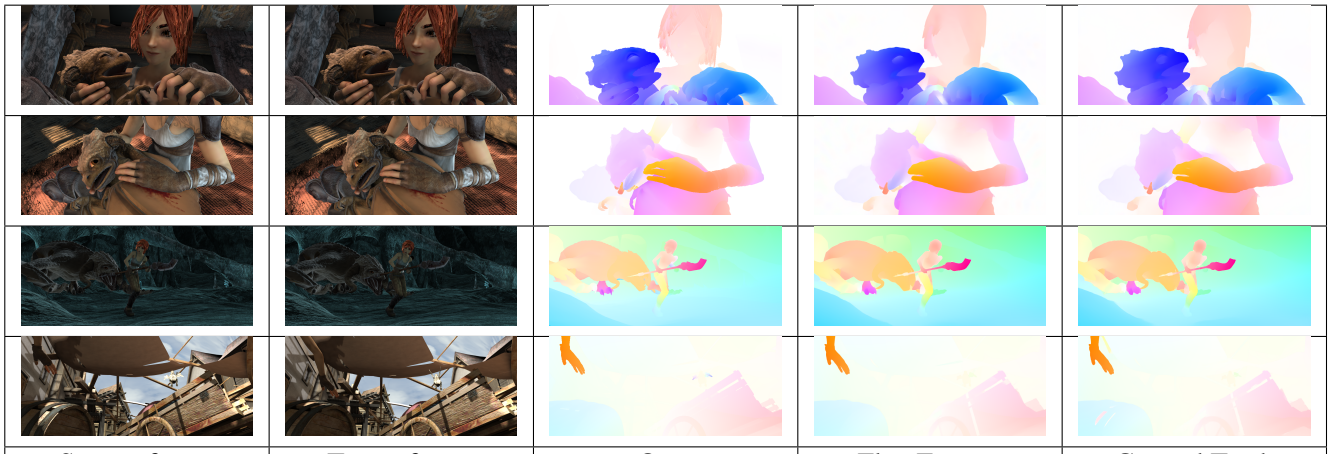


Figure 3: Qualitative comparison on the Sintel training set

Cost Volume	Sintel		KITTI	
	Clean	Final	F1-epe	F1-all
Static	1.51	2.65	4.61	16.11
Dynamic	**1.23**	**2.63**	**4.21**	**15.99**

(a) Static vs Dynamic Cost Volume

Transformer Blocks	Sintel		KITTI	
	Clean	Final	F1-epe	F1-all
No weight sharing	1.05	2.47	4.05	14.77
Shared weights	**1.02**	**2.41**	**3.99**	**14.58**

(b) Transformer blocks with and without weight sharing

Feature Enhancement	Sintel		KITTI	
	Clean	Final	F1-epe	F1-all
Before IRR	**1.00**	2.44	4.29	15.46
Inside IRR	1.02	**2.41**	**3.99**	**14.58**

(c) Feature enhancement: before refinement vs concurrent to refinement

Cost Representation	Sintel		KITTI	
	Clean	Final	F1-epe	F1-all
Matching Cost	1.22	2.76	4.56	16.24
Matching Probability	**1.02**	**2.41**	**3.99**	**14.58**

(d) Matching cost vs matching probability based cost volume lookup

Table 2: Ablation studies

evident from Table 2a that having the cost volumes inside the IRR improves the flow estimation accuracy, with upto a reduction of 18.5 % of the EPE on the Sintel Clean benchmark.

Transformer Weight-sharing: We compare the nature of our cost volume updates using the transformer architecture with both weight-tied and distinct transformer blocks. From Table 2b, it is evident that weight-sharing consistently improves the cross-domain generalization on all the benchmarks while maintaining the number of parameters of the transformer architecture to nearly half of the one without weight sharing.

Feature enhancement relative to IRR: Here, we test the effectiveness of contextualizing feature maps concurrent to flow refinement iterations as compared to before IRR. Model parameters, computational overhead and memory requirements remains exactly the same in both the scenarios. However, the cost volumes are dynamic inside IRR in the former setting, whereas the update operator decodes the same cost volume over the refinement iterations in the latter setting. We observe an EPE reduction of 7 % for the non-occluded pixels and a reduction of 5.7 % for all the pixels of the KITTI dataset (Table 2c).

Matching probability based Lookup Transforming the matching costs into their corresponding matching distributions stabilizes the training of the weight-tied update operator of IRR. From the performance point of view, we see a reduction in the EPE of 16.7 % and 12.4 % on the Clean and Final passes of the Sintel dataset, whereas the error reduction for all the pixels in the KITTI dataset is 10.2 % (Table 2d).

5 Conclusion

We have demonstrated a new transformer based architecture for Optical flow with dynamic cost volumes, which delivers optical flows with near state-of-the-art cross-domain generalization, but at the same time maintaining very good scalability of the overall model. Having a deep transformer architecture inside the refinement stage, however presents opportunities for future works to use the sparse residual flows to further design efficient self and cross attention layers for visual feature enhancement.

Acknowledgments: This research has been partially funded by the German BMBF project SocialWear (01IW20002) and EU project CORTEX2 (Grant Agreement: Nr 101070192).

References

[Butler et al., 2012] Butler, D. J., Wulff, J., Stanley, G. B., and Black, M. J. (2012). A naturalistic open source movie for optical flow evaluation. In *European conference on computer vision*, pages 611–625. Springer.

[Dosovitskiy et al., 2015] Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., and Brox, T. (2015). Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766.

[Geiger et al., 2013] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237.

[Horn and Schunck, 1981] Horn, B. K. and Schunck, B. G. (1981). Determining optical flow. *Artificial intelligence*, 17(1-3):185–203.

[Huang et al., 2022] Huang, Z., Shi, X., Zhang, C., Wang, Q., Cheung, K. C., Qin, H., Dai, J., and Li, H. (2022). Flowformer: A transformer architecture for optical flow. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVII*, pages 668–685. Springer.

[Mayer et al., 2016] Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., and Brox, T. (2016). 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.

[Sun et al., 2018] Sun, D., Yang, X., Liu, M.-Y., and Kautz, J. (2018). 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.

[Teed and Deng, 2020] Teed, Z. and Deng, J. (2020). Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

[Xu et al., 2022] Xu, H., Zhang, J., Cai, J., Rezatofighi, H., and Tao, D. (2022). Gmflow: Learning optical flow via global matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8121–8130.

[Zhao et al., 2022] Zhao, S., Zhao, L., Zhang, Z., Zhou, E., and Metaxas, D. (2022). 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.