DIFT: Dynamic Iterative Field Transforms for Memory Efficient Optical Flow

Risheek Garrepalli Jisoo Jeong Rajeswaran C Ravindran Jamie Menjay Lin† Fatih Porikli
Qualcomm AI Research*
{rgarrepa, jisojeon, rajeswar, jmlin, fporikli}@qti.qualcomm.com

Abstract

*Recent advancements in neural network-based optical flow estimation often come with prohibitively high computational and memory requirements, presenting challenges in their model adaptation for mobile and low-power use cases. In this paper, we introduce a lightweight low-latency and memory-efficient model, Dynamic Iterative Field Transforms (DIFT), for optical flow estimation feasible for edge applications such as mobile, XR, micro UAVs, robotics and cameras. DIFT follows an iterative refinement framework leveraging variable resolution of cost volumes for correspondence estimation. We propose a memory efficient solution for cost volume processing to reduce peak memory. Also, we present a novel dynamic coarse-to-fine cost volume processing during various stages of refinement to avoid multiple levels of cost volumes. We demonstrate first real-time cost-volume based optical flow DL architecture on Snapdragon 8 Gen 1 HTP efficient mobile AI accelerator with **32 inf/sec** and 5.89 EPE (endpoint error) on KITTI with manageable accuracy-performance tradeoffs.*

1. Introduction

Optical flow is the task of estimating pixel-level correspondences between video frames. In recent years, starting with [3, 8], deep learning-based optical flow methods have shown remarkable performance while demanding high memory and substantial computation [10, 25, 27]. Such computational requirements are not feasible for mobile or other resource-constrained use cases.

Inspired by RAFT [25] there are follow up works focusing on efficient approaches [26, 28], but primary focus of these approaches is to improve performance with some reduction in overall compute. These approaches too employ cost volumes, requiring significant computational resources and are not directly applicable for mobile or low compute deployment. Design principles of our approach in terms of cost volume processing, number of iterations and resolution of cost-volume can be extended to other modern architectures which employ cost volumes.

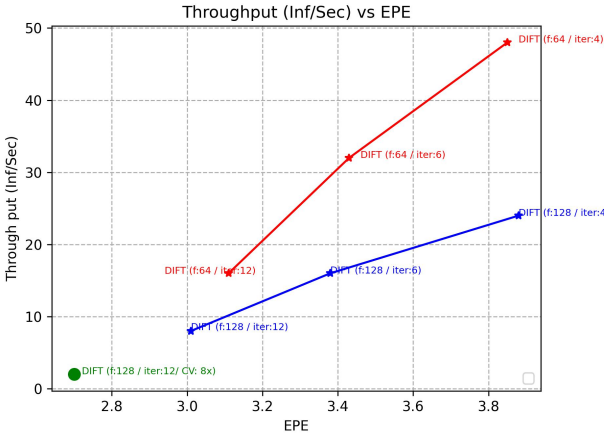

Figure 1. Accuracy (in endpoint error) vs. latency comparisons of DIFT over feature dimensions on Snapdragon HTP Platform. We observe that DIFT with a single-level cost volume at $1/16$X with feature dimension of 64 (f=64) out of 4 iterations achieves an on-target throughput of 48 inf/sec with an EPE of 3.85, while DIFT at $1/8$X and f=128 out of 12 iterations achieves 2 inf/sec with 2.7 EPE. The RAFT [25] baseline is not presented in this figure, as the original RAFT model cannot fit in the memory of commercial smartphones without major modifications in the network model architecture. More details can be found in Table 1.

Global vs Local Cost Volume Processing: If we review various architectures for optical flow, most competitive approaches leverage cost volumes or cross-attention. We can classify approaches by whether they can capture all-pairs similarity measure. If approaches have global cost volumes, then it import to understand whether down-stream cost volume processing is based on global operation such as attention, 3D convolution or local operators such as grid-sampling, 2D convolution.

RAFT [25] performs grid sampling only in the *local neighborhood* of current estimate of optical flow, but processes information in all-pairs cost-volume i.e, using spatially varying local look-up operation across iterations. Whereas approaches such as GMFlow with the *global* attention or Mobile Stereo [20] which applies 3D convolution for stereo correspondence need to operate atomically on global

∗ Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.
† This work was done while at Qualcomm AI Research.

cost volumes, thus requiring much higher computation and peak memory space.

Therefore, *we identify an unique opportunity for magnitude-order complexity reduction in the type of local processing particularly with RAFT*. RAFT has significant inductive biases inspired by traditional energy minimization approaches and inspired follow-up approaches that leverage transformers [27]. We perform additional modifications to RAFT and analyse various design choices and compare to our efficient version DIFT. This helps us in understanding latency vs performance trade-off as illustrated in Figure 1 and detailed evaluation is discussed in 5.1.

Objective of this work is to adopt cost-volume based complex DL architectures and understand critical design choices for mobile use-cases. We introduce *Dynamic Iterative Field Transforms (DIFT)* for optical flow or stereo estimation, a novel memory and computationally efficient architecture for real-time computation on extreme low compute mobile platform which can run with less than 4MB memory while preserving core inductive biases of recent approaches like RAFT [25].

We identify that single level cost volume few iterations give reasonable performance even for large motion (KITTI). More detailed description of our method can be found in 3. Our proposed architecture is evaluated on standard optical flow benchmark datasets [1, 3, 5, 11, 15, 16].

In summary, inspired by RAFT we propose a novel memory and computationally efficient optical flow architecture (DIFT) for mobile use cases with novel components for better deployment.

- Introduce a novel coarse-to-fine approach (3.2) for correlation lookup with varying cost-volume resolution across iterations.
- Extend efficient correlation lookup approach (3.3) from RAFT [25] to fit within less than 4MB.
- We introduce an efficient algorithm, *bilinear shift*, (3.4) which achieves $8\times$ sampling throughput with arithmetic equivalent to the baseline bilinear sampling for HW efficient warping.
- On target deployment and latency evaluation of DIFT, on Snapdragon HTP platform with 8MB Tightly coupled memory (TCM) but also evaluated on platform with less than 4MB.

2. Related Work

Optical Flow: There has been lot of work deep learning based approaches for optical flow estimation, and an active research area. FlowNet [3] introduced the first end-to-end CNN based architecture for optical flow estimation, and FlowNet2 [8] developed a stacked architecture that includes a warping image with optical flow prediction. FlowNet2 shows improves performance by refining optical flow at multiple stages, and such iterative refinement framework is adopted later in [7, 25].

There are various approaches which leverage coarse-to-fine techniques with image or feature pyramids. They compute the large displacement at a coarse level and get refining at the finer level. SpyNet [17] combined traditional coarse-to-fine image pyramid methods with deep learning, and PWC-Net [24] replaced the fixed image pyramid with learnable feature pyramids. Additionally, PWC-Net conducted the warping operation at the layer level in the network. But these approaches process cost volume locally once & lack capability of iterative refinement architectures. In our work we introduce coarse-to-fine approach for cost volume processing within iterative refinement scheme, we will describe more in detail in Section 3

Recurrent All Pairs Field Transforms (RAFT) [25] demonstrated significant performance improvement over previous methods, and has influenced many subsequent research [6, 9, 10, 27]. RAFT generates multi-scale and all-pairs field 4D correlation volume from feature extraction outputs and iteratively updates the optical flow estimates through GRU with local grid sampling within 4D correlation volume.

RAFT has demonstrated good generalization performance and is more robust to adversarial attacks compared to prior works [19] and has very good inductive biases which is adopted by followup works. Motion is one of fundamental geometric cues applicable in autonomous driving, robotics and many other safety critical applications. Hence a robust and generalizable optical flow estimation is useful not just for down-stream tasks, but also to detect novel objects i.e., openset and Out-of-distribution detection [18] complementing appearance based features, representational learning and uncertainty measures based approaches [4, 12, 13].

Memory Efficiency Model: Flow1D [26] decomposed 2D correlation to 1D and DIP [28] proposed patchmatch framework. However, they still require large memory compared to their baseline RAFT algorithm. The Correlation Volume of Flow1D requires $O(HW(H+W))$ memory, which is less than RAFT $O((H+W)^2)$, but depending on resolution, it requires still large memory and pre-computes cost volume. This could also be an interesting direction to consider but we do not consider such decomposition in this work. DIP adopted previous warping method, and it reduced the memory to $O(HW)$. But, DIP computes the correlation volume even at 1/4 downsampled resolution features and has multiple computations to process cost volumes for each update. With low resolution (448×1024), DIP (1.56GB) requires about 3 times more memory compared to RAFT (0.48GB).

Modern Optical Flow Architectures: Key components

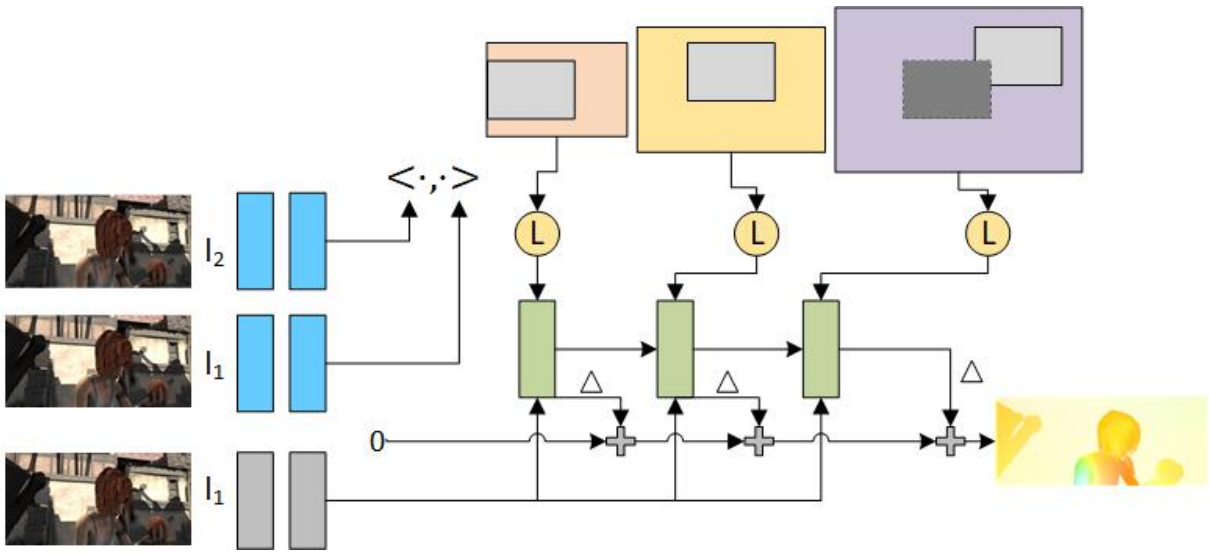

Figure 2. **DIFT with Coarse-to-Fine Cost Volumes and Lookup Concatenation**. CNN based encoder applied to images I_1, I_2 and additional context Encoder applied to I_1. Three different resolutions of cost volumes for various update stages are illustrated in orange, yellow and light purple. Each update block(green) takes output of lookup operator at corresponding cost volume resolution z_k, r_k. And overall flow is updated with element-wise sum across iterations as illustrated in bottom of figure. For visual clarity did not include Look-Up Concatenation in the architecture diagram.

of modern optical flow architectures such as RAFT include feature extraction, cost volumes or cross-attention i.e., to use some similarity metric, iterative or multiple stages of flow refinement informed by cost volumes or cross attention, and finally coarse to fine spatial information processing either having multi-scale features and/or multi-scale cost volumes, and having coarse to fine cost volume is potentially useful for capturing large motion.

There are many later works [6, 27] which adopt core design principles from RAFT by leveraging attention & transformers while adopting similar design in terms of cost volume or cross attention, (iterative) refinement of flow/correspondence estimation informed by cost volume.

Improving Training Pipeline and Data: Recent approaches such as Autoflow [23] and [22], which improve training strategy, or works such as [10] which leverage augmentations to impose consistency and improve performance, are orthogonal to our work. Our architecture can also leverage such training techniques, but we do not consider such techniques for this work.

3. DIFT

We introduce DIFT following core design principles of RAFT, an iterative refinement network which consists of a feature extractor, cost volume based iterative refinement scheme with novel components. DIFT adopts a single level of cost volume with potentially varying resolutions. The overall architecture is illustrated in Fig 2, which includes our dynamic coarse to fine lookup, and illustration of sample local grid for a region. For DIFT, we do not pre-compute or compute all-pairs cost volume for each update step but can leverage all-pairs information across updates.

Now lets review core components of RAFT and additional modifications for DIFT, *we adopt notation from RAFT paper*.

3.1. Architecture Overview

Feature Extraction: We adopt a simple convolutional feature encoder g_θ which is applied on I_1 and I_2. Output dimension of g_θ is $\mathbb{R}^{H/K \times W/K \times D}$, where 'K' is the scale of down-sampling 16 or 8 in our experiments and 'D' is the feature dimension, we try $D = 64, 128, 256$ in our experimental evaluation. Similar to RAFT we also have a separate context encoder applied on base image and output of context encoder is passed to update block. Unlike RAFT for final version we choose $K = 16$ to minimize peak memory and overall latency.

Cost Volume Computation & Lookup: Given images I_1, I_2 and image features $g_\theta(I_1), g_\theta(I_2)$ then *all-pairs* correlation or cost volume $\mathbf{C}(g_\theta(I_1), g_\theta(I_2))$ is computed by applying dot product where each element has its value defined by $C_{ijkl} = \sum_h g_\theta(I_1)_{ijh} \cdot g_\theta(I_2)_{klh}$.

Let L_C be the correlation lookup, which generates a correlation feature map, is obtained based on current estimate of flow $f^k = (f^1, f^2)$ denote flow in x,y directions, respectively. Then for each pixel $x = (u, v)$ in I_1 to obtain correspondence in I_2, we define a local neighborhood region around x defined by $\mathcal{N}(x')_r = \{x' + dx | dx \in \mathcal{Z}^2, ||dx||_1 \leq r\}$.

We consider different resolutions of cost volumes C^1, C^2, C^3 obtained by pooling on feature maps for corre-

lation lookup, or pooling along last two dimension in case of RAFT-small as practiced in [25]. For *coarse to fine lookup*, for each update step we only adopt one level of cost volume C^i and corresponding lookup L_{C_i}.

Update Block: Our default update block is adopted from RAFT, lets review components of update block. Each update block takes current flow estimate f^k, correlation lookup output, context encoder features. Update block consists of few pre-processing convolutional layers for processing current flow estimate, correlation features.

Let z_k, r_k be features obtained by after processing current flow f^k and correlation features, respectively. Then based on z_k, r_k and additional input from context encoder and previous hidden state are input to the update block. For *lookup concatenation* we do not just pass z_k, r_k but also z_{k-1}, z_{k-2} and r_{k-1}, r_{k-2} also as inputs to update block.

3.2. Coarse-to-Fine Lookup

For DIFT, we adopt a single level cost volume per-iteration and we choose $1/16$ as resolution for cost-volume (CV). To reduce overall latency while aiming to capture large motion, we introduce *Coarse-to-fine strategy for choosing resolution of cost volume* to have a varying effective receptive field across iterations. We achieve this by adopting a coarser resolution for CV for initial iterations, with an intuition that these initial steps should provide good initialization for further refinement. For later iterations, increase the resolution of cost volume so that now the refinement can focus on finer correspondence. And if needed, we can also have one/two iterations of coarser cost volume resolution after finer refinements to capture if there are any regions with larger displacements.

To dynamically vary cost volume, we perform average pooling on the encoder's feature maps to modify the resolution of cost volume. Figure 2, demonstrates varying cost volume resolutions across iterations. We can also consider a method where we vary the lookup radius and potentially consider irregular sampling with the same cost volume resolution across updates. But such irregular sampling or larger neighborhood sampling is inefficient on hardware, and hence we do not investigate this approach.

Weight Sharing: As we have varying receptive fields w.r.t cost volume, we do not share parameters of update block across all iterations, but weight sharing is still present for update blocks which process same resolution of cost volume.

3.2.1 Look-Up Concatenation

Usually update block within RAFT only has local neighborhood information within cost volume, but GMFlow [27] and other works have shown performance improves with global processing of cost volume, but global processing would require significant memory. To retain more information than local neighborhood information captured in lookup operation. We concatenate the output of lookup operation, previous flow estimate (could also be equivalent to position encoding) over past few iterations as additional input to update block.

For simplicity we did not include concatenation part in 2 but darker square block illustrates additional region of correlation volume which can be given as additional input capturing more information than local look-up with less computation & memory.

Let L_{C_t} be the lookup operator at iteration 'k' which returns correlation features and z_k, r_k based current estimate of flow f^k. And similarly let z_{k-1}, r_{k-1} are obtained by previous lookup operations $L_{C_{k-1}}$ and z_{k-2}, r_{k-2} from $L_{C_{k-2}}$. When we adopt concatenation within lookup our update block does not just take z_k, r_k but also z_{k-1}, r_{k-1} & z_{k-2}, r_{k-2}.

3.3. Just-in-Time Construction & Lookup

Extending a computation alternative that samples for each pixel only by the required feature grids based on corresponding neighborhood as discussed in RAFT [25], we further decompose the linear-complexity construction and look-up operations in our *just-in-time* (JiT) approach in order to achieve low peak memory against the tightly-coupled memory (TCM) (4MB) constraints on typical smartphones.

Even during such on-demand lookups we technically need to construct a 3D cost volume based on a fix-sized look-up radius (R) and such approach would need $O(N \times 2R)$ memory complexity with $N = (H/16 \times W/16)$, which may not in practice be feasible to fit in a memory smaller than 2MB.

Specifically, to ensure a system of balanced pipelining, which involves memory read/write accesses and neural network processing on our target hardware, we optimize parameters including N_{slice} i.e., the number of tiles in $g_\theta(I_1)$ decomposition and R for the radius of look-up range in either direction for each pixel. DIFT with the JiT design successfully achieves a peak memory at only $2MB$ with $\{((H/16 \times W/16) \times (2R))/N_{slice}\}$. In the case of Sintel, we adopt $N_{slice} = 56$ for DIFT at $1/16$ cost volume resolution.

Instead of processing entire feature map $g_\theta(I_1)$ at once, *we process it few pixels at a time in a sliding window fashion* and based on neighborhood $\mathcal{N}(x')_r$,retrieve corresponding features from $g_\theta(I_2)$ to construct correlation volume for current slice, and then aggregating to generate overall correlation volume for current iteration. In our implementation *we decompose real-valued and sub-pixel part of current flow estimate*, to first use real-valued neighborhood to construct cost volume in a sliding window approach and later adjust for sub-pixel part using bilinear shift approach

Algorithm 1 Bilinear Shift

Let input tensor to bilinear shift be T
Let T_{x0}, T_{x1} be sliced tensors in width (X) axis
Let T_{y0}, T_{y1} be sliced tensors in height (Y) axis
Let l_{T_x}, l_{T_y} be length of Tensors in X, Y axis
$T_{x0} = T[0 : l_{T_x} - 2]$ and $T_{x1} = T[1 : l_{T_x} - 1]$
$T_{x0} = (1 - \Delta x) \times T_{x0}$
$T_{x1} = (\Delta x) \times T_{x1}$
$T_2 = T_{x0} + T_{x1}$
$T_{y0} = T_2[0 : l_{T_y} - 2]$ and $T_{y1} = T_2[1 : l_{T_y} - 1]$
$T_{y0} = (1 - \Delta y) \times T_{y0}$
$T_{y1} = (\Delta y) \times T_{y1}$
$T_{out} = T_{y0} + T_{y1}$
return T_{out}

discussed in next section.

3.4. Bilinear Shift

Bilinear Sampling is used within each iteration of GRU update to perform warping based on current estimate of flow. Bilinear sampling involves a series of pointwise grid operations followed by series of bilinear interpolation operations based on current estimate of flow. This operation is not efficient on hardware as it cannot leverage vectorized execution and hence adds significant bottleneck to latency. To alleviate this problem, we propose a mathematically equivalent Bilinear shift which replaces grid sampling with vectorizable element-wise operators, and we assume grid input is uniform which can work for dense optical flow estimation.

Based on a shift along x and y dimension i.e., $\Delta x, \Delta y$, the grid sample operation can be decomposed into a bilinear operation associated with a 2D shift, and such operations can be mapped to vectorizable elementwise operators on hardware, and can also allow flexibility w.r.t compilation/run-time as it can allow further decomposition into smaller operators/tiling.

Algorithm 1 describes overall Bilinear shift operation given input tensor T, the x-shift (Δx) and y-shift (Δy). For bilinear shift, first input tensor T is split into two equal parts indexed by one pixel offset along x direction and then interpolated using x-shift Δx and $(1 - \Delta x)$. Then an analogous operation is repeated along y direction.
We compare bilinear shift with baseline grid sampling on our hardware simulation platform to analyze latency. For a given input dimension, bilinear shift has a throughput of 12277.3inf/s (81.45μs) whereas grid sample has throughput of 1483.2 inf/sec (674.22μs) i.e., 8$\times$ *improvement in throughput (inf/sec)*

4. Experimental Setup

4.1. Datasets

We evaluate our approach on common optical flow benchmarks, Sintel (S) [1], KITTI (K) [5, 16]. For evaluation on Sintel we use checkpoint pre-trained on on FlyingChairs (C) [3] + Flying Things(T) [15], following a common protocol. And for KITTI evaluation, we start with pre-train on (C+T) and fineune (T) checkpoint additionally on Virtual KITTI2 (VK) [2].

4.2. Architecture Details

We build our DIFT model on top of the RAFT [25] baseline [1] along with its original hyperparameters. We adopt RAFT-small CNN architecture for Image encoders as starting point, we add one additional convolutional layer to further down-sample input to $1/16x$ image resolution for cost volume processing. We choose lookup radius to be '3' for all our experiments based to maximize throughput on Snapdragon HTP hardware. And we don't adopt convex upsampling.

4.2.1 Weight Sharing of Update Block

For baseline single level cost volume variant of DIFT we adopt update block of RAFT-small and have shared weights across iterations i.e., we use a convolutional GRU based on GRU Cell. But for coarse to fine variant, we observe that weight sharing across iterations actually performs worse and *we share weights dependent on resolution of cost volume* across iterations.

For concatenation of lookup output, we take output of convolutional pre-processing layers before GRU cell within update block from previous iterations as additional information so if required GRU block can leverage such additional context. Even though we vary the cost-volume resolution across iterations but as the reference optical flow and cost-volume resolution is fixed and only the relative fidelity of each update operation changes the training is stable and convergence is fast on-par with RAFT compared to previous DL based approaches such as PWC-Net or FlowNet2.

4.3. Implementation Details

We pre-train DIFT for 100k iterations on FlyingChairs (C) [3] for 100k steps followed by training on FlyingThings3D (T) [15] initialized with previous checkpoint. We adopt batchsize of 12 with AdamW [14] optimizer for both the datasets. When trained on Virtual KITTI [2] we adopt RAFT fine-tuning hyper-parameters for KITTI. We do not perform any finetuning on Sintel or KITTI and both the datasets are only used for evaluation. Also, to have a fair

[1] https://github.com/princeton-vl/RAFT

Table 1. Analysis for RAFT & DIFT on Levels of cost volume, Cost volume resolution, feature encoder dimension, inference time iterations and corresponding latency. (L^{CV}: Cost Volume Levels, R^{CV}: Cost Volume Resolution (Downsampling), $\#F$: Encoder Feature Dimension, $\#I$: Test iterations and $\#radius$: lookup radius). C+T represents model trained on Flyingchairs and FlyingThings & VK indicates fine-tuned on Virtual KITTI for KITTI evaluation only. In below table best latency-aware version is highlighted with bold, overall best performing DIFT model is denoted via underline.

RAFT									DIFT (Ours)				
L^{CV}	R^{CV}	$\#F$	$\#I$	Sintel (train-EPE) ↓ Clean	Final	KITTI (Train) EPE↓	F1-all ↓	On-target (ms (Inf/s))	Sintel (train-EPE) ↓ Clean	Final	KITTI (Train) EPE↓	F1-all ↓	On-target ms ↓ (Inf/s) ↑
Cost Volume Level (C+T)													
4				2.65	3.94	11.87	38.02		N/A	N/A	N/A	N/A	N/A
3	$\frac{1}{16x}$	128	12	N/A	N/A	N/A	N/A	N/A	3.20	4.17	12.54	43.07	496.7 (2)
2				2.67	3.81	11.23	38.14		3.07	4.18	12.67	41.15	248.3 (4)
1				2.69	3.76	10.63	38.38		3.01	4.32	12.21	42.06	124.9 (8)
Cost Volume Resolution (C+T)													
1	$\frac{1}{8x}$	128	12	2.34	3.42	9.06	28.20	N/A	<u>2.70</u>	3.93	10.33	31.47	<u>498.7 (2)</u>
	$\frac{1}{16x}$			2.69	3.76	10.63	38.38		3.01	4.32	12.21	42.06	124.9 (8)
Feature Dimension (C+T)													
1	$\frac{1}{16x}$	128	12	2.69	3.76	10.63	38.38	N/A	3.03	4.32	12.21	42.06	123.2 (8)
		64		2.67	3.81	10.52	37.35		3.11	4.19	12.87	43.83	62.5 (16)
Number of Iterations (C+T/VK)													
1	$\frac{1}{16x}$	128	4	3.31	4.41	6.55	24.76		3.88	5.08	7.25	30.06	41.7 (24)
			6	2.95	4.03	5.91	22.29		3.38	4.63	6.45	26.69	62.5 (16)
			12	2.68	3.76	5.11	24.33		3.01	4.32	5.67	24.33	125 (8)
Proposed Method (Coarse2Fine) (C+T/VK)													
1	$\frac{1}{16x}$	128	4						3.61	4.71	6.66	30.41	42.1 (24)
			6						3.38	4.44	5.83	28.05	62.9 (16)
			12						3.09	4.13	5.56	25.76	126.3 (8)
1		64	6						**3.44**	4.57	**5.89**	26.81	**32.8 (32)**
Proposed Method (Coarse2Fine + Concat) (C+T/VK)													
1	$\frac{1}{16x}$	128/	4						3.64	4.91	**6.55**	31.1	45.45 (22)
			6						3.54	4.62	6.35	30.40	71.4 (14)
			12						3.04	4.15	5.51	25.76	142.85 (6)

comparision with DIFT we also adopt a maximum of 12 iterations/GRU Update steps for RAFT-small. But we also evaluate both DIFT & RAFT-small variants after 4,6 and 12 updates for various ablation study as latency is a key constraint for mobile deployment.

5. Results

Primary goal of our work is to obtain a good real-time optical flow, stereo or broadly correspondence estimation architectures with good inductive biases, generalization ability for mobile platforms. In this section we want to understand what are key bottlenecks & critical design choices in modern cost-volume based DL based architectures such as [25] for mobile or low compute settings. We analyse key design choices as discussed in 5.1 with in framework inspired by RAFT [25] to understand what matters in correspondence estimation for mobile/low-compute settings.

5.1. Ablation Study

To understand *performance vs latency trade-off*. In section 3, we discussed our architecture and motivation for our design choices. Here we present results for representative combinations of design choices and provide guideline recommendation for architecture exploration for correspondence estimation.

Coare-to-Fine Lookup: To understand effectiveness of our proposed approach we can compare Coare-to-fine variants with fixed resolution of cost volume across iterations. From Table 1 we can observe that both for fewer iteration setting coarse-to-fine lookup gives significant performance boost. At 4 iterations, our approach improves performance form 7.25 → 6.66 EPE i.e., 0.59 boost in EPE over baseline corresponding DIFT. Similarly *Delta 0.62 boost in EPE* for 6 iterations with final EPE of 5.83 only 0.16 EPE less than 12 iteration version of DIFT. In addition to Coarse-to-fine approach lookup-concatenation further improves the performance by 0.11 EPE for 4 iteration setting.

Overall we observe that our Coarse-to-fine approach consistently improves performance *without additional increase in latency* and is especially effective for large motion setting such as KITTI compared to Sintel, where even a single level cost volume would allow to provide good initialization

Number of Levels of Cost Volume: We observe that though additional levels of cost-volumes does not necessarily improve performance significantly but adds significant cost to latency. From table 1 we can observe that when go from 1 level cost-volume to 3-levels in case of DIFT, EPE actually gets from 3.01 to 3.20 in case of sintel-clean, similarly it worsens in case of KITTI. But performance im-

proves marginally in case of Sintel Final for DIFT and also in case of RAFT-small variants. But as the performance of single-level cost volume is good-enough it might be worthwhile to investigate such approaches towards efficient solutions.

Cost Volume Resolution: In our experiments we find that resolution of cost-volume is one of significant design choices w.r.t performance. When we go from $H/8 \times W/8$ to $H/16 \times W/16$ we can observe that EPE drops from 3.93 to 4.32 i.e., a 0.39 loss of EPE on sintel Final in case of DIFT. But as the latency is also reduced by 4x, we choose to pursue $1/16x$ resolution.

Effect of Iterations: Based on Table 1 we can observe that though increasing iterations improve performance, relative improvement is marginal compared to additional compute cost from 5.83 to 5.56 EPE for double the latency in case of KITTI.

Feature Dimension: In our experiments we find that reducing feature dimension to as low as 64 still gives reasonably good performance. This helps us get to close to real-time solutions by reducing the feature dimension.

Overall Performance: Compared a best-performing DIFT with 12 iterations and one $H/8 \times W/8$ cost-volume resolution we get a **16x** boost in latency with relatively minimal performance drop 2.70 (2inf/sec) $\rightarrow$ 3.46 (32 inf/sec) EPE on sintel clean where we adopt a coarse-to-fine variant of DIFT with 6 iterations and feature-dimension/number of encoder channels is 64. In case of KITTI we can get 5.89 EPE with real-time latency of 32 inf/sec.

To the best of our knowledge this is first real-time demonstration of cost-volume based deep learning solution for extreme low-compute mobile platforms. From 3 we can observe that DIFT output is on-par with RAFT-small while being significantly faster.

5.2. On-target Evaluation

In our experiments, we also evaluate DIFT on Snapdragon 8Gen 1 HTP (power efficient mobile AI accelerator for Neural Networks) to understand latency of different design choices. We have developed and optimized a deployment pipeline Snapdragon® 8 Gen 1 HTP accelerator platform, to leverage maximum out of hardware for this model. We adopt INT8 quantization (W8A8) based on AIMET quantization toolkit [21] and QNN-SDK [2]from Qualcomm® AI Stack. [3]

Depending on target platform few details might influence different final performance, but our experiments and observations should be informative and translate to other platforms.

[2] https://developer.qualcomm.com/software/qualcomm-ai-stack

[3] Snapdragon and Qualcomm branded products are products of Qualcomm Technologies, Inc. and/or its subsidiaries.

5.3. Analysis

From Table 1, we can observe that latency is largely dependent on *size/memory of cost volume*, as the bottleneck is largely w.r.t memory access for such large tensors (cost volumes) after optimizing inference pipeline. Based on these on-target latency evaluations, we can infer following dependencies w.r.t key design choices

- Latency is proportional to levels of Cost Volume as the number of lookups we perform is dependent on number of levels of cost-volume per GRU Update.
- Latency is proportional on maximum resolution of cost volume as this determines the memory transfer and the also GMACs within lookup.
- Latency is proportional to number of iterations , as this also effects total number of lookups per inference.
- Latency is proportional to feature/channel dimension as this would also effect memory size of cost volume.

In summary, we observe that *latency is proportional to number of lookup operations and memory requirement of each lookup,* as this is inherently a memory bounded operation and hence latency is significantly determined by peak-memory of hardware module and cost-volume size.

If there is an efficient way to decompose cost-volume or cross-attention construction and processing more broadly then that would put less constraints on our network architectures.

We can observe from the Table 1 that, as iterations decrease from 12 $\rightarrow$ 4 throughput triples from 8 inf/sec $\rightarrow$ 24 inf/sec, or latency drops from 126.3ms to 42.1ms. Reducing feature dimension from 128 to 64 at 4 iterations further improves throughput from 24 inf/sec to 48inf/sec or 20.8ms latency.

5.3.1 Peak Memory

In this section, we discuss the effect of peak memory on overall latency, as determined by number of lookups which is dependent on size of cost volume and number of levels of cost-volumes for each GRU iteration.

From table 2 we can observe that $1/8x$ variant of RAFT needs peak memory of 56.3 MB which is a significant challenge for mobile platforms as they have very limited TCM memory, often less than 10MB. As further decomposition of JiT in DIFT allows to decompose overall operation, if needed DIFT supports higher resolution of cost-volume.

In case of Sintel, even after 16x down-sampling instead of 8x (for efficient latency), and **radius = 3**, we will have 440 x 1024 original image down-sampled, then effective cost volume including padding would result to $28 \times 64 \times (2 \times (radius = 3 + 1))^2 \times 128 = 28 \times 64 \times 64 \times 128 = 14,680,064/1024 = 14.3MB$ and similarly at 8x and

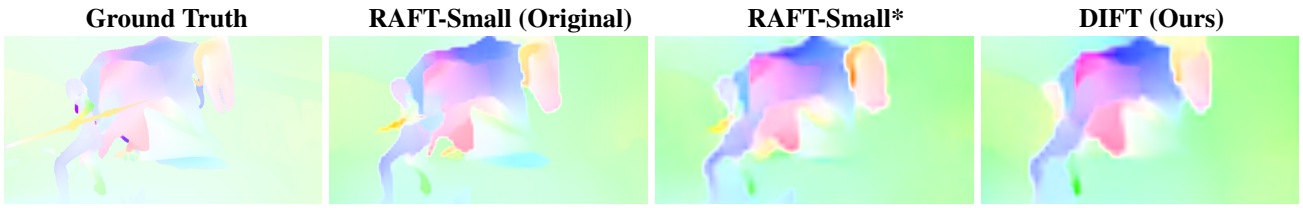

Figure 3. In this figure we compare optical Flow output on a Sintel test-case comparing DIFT to RAFT Variants. RAFT-small is default version with 4 level cost volume, 24 GRU iterations and $1/8\times$ cost-volume resolution. Where is DIFT and RAFT-small* are single-level cost-volume versions with 6 GRU iterations.

Table 2. Peak Memory Comparison DIFT & RAFT

Model	**Down sampling**	**Peak Memory**	L^{CV}	**Latency for 12 Iterations**
DIFT	16x	$< 1MB$	1	124.9 ms
DIFT	16x	$< 1MB$	4	501.3 ms
RAFT	16x	$14.3MB$	4	N/A
DIFT	8x	$< 1MB$	1	499.2 ms
DIFT	8x	$< 1MB$	4	2010.4 ms
RAFT	8x	$56.3MB$	4	N/A

$radius = 3$ its 56.32MB. For DIFT with 4 levels of cost volume (16x), which is similar to RAFT it would take approximately ($124.9 \times 4 = 499.6ms$) for one iteration which is equivalent to DIFT at 8x down-sampling but less effective.

Without further decomposition of cost-volume construction and processing as discussed earlier we cannot run RAFT on-target because of peak-memory constraints i.e., as mobile smartphones have limited (Tightly coupled Memory), often less than 10MB. Instead without decomposition if we try running RAFT on-target based on DDR-memory bound inference we observe that latency of single iteration is less than 0.2 inf/sec. So we do not report latency w.r.t RAFT in our experiments.

We also did not run alternate solutions for optical flow on Snapdragon HTP, because unlike bench-marking on NVIDIA GPUs with CUDA, these mobile platforms ML deployment workflows are still under-development and usually does not support all operations on hardware. Typically such operations have to be identified, substituted and end to end model also might need additional optimizations w.r.t effective usage of hardware for a fair Comparison. As the goal of this study is predominantly to understand to what extent we can adopt cost-volume based solutions for correspondence estimation for mobile, and understand key design choices we start with RAFT and similar intuitions can be extended to various recent follow up works with cost-volumes or cross-attentions, iterative reasoning, etc.

5.4. Recommendations for Architecture Design:

- As Cost-volume computations are memory-bound, adopt architectures with local cost-volume processing so that overall operation can be decomposed for mobile platforms.
- For most settings single-level or coarse-to-fine lookup seem to work with very minimal performance drop with significant boost in latency.
- Optimize overall size of cost-volumes by choosing appropriately low resolution of cost-volume at-least for initial iterations to save memory.
- Given a choice between number of levels of cost-volumes vs maximum resolution of cost-volume, its better to adopt a maximum resolution one with coarse-to-fine lookup strategy.
- Reduce Feature dimension/number of output channels in encoder as much as possible as for many settings within such iterative refinement architectures.
- If required increase encoder complexity as it only needs to run-once and doesn't introduce significant bottlenecks w.r.t latency.

6. Conclusion

In this paper, we have introduced DIFT, a computation and memory efficient real-time optical flow algorithm with competitive performance and good inductive biases . To the best of our knowledge, this is first work to adopt and perform various on-target analysis of such complex cost volume based refinement architectures for mobile use-cases. Based on our experiments, we observe that DIFT is feasible for real-time mobile solutions and shows good performance. Nevertheless, any lightweight design choices come at the cost of an expected performance drop. This means, there is a scope for further research to improve correspondence estimation for mobile use-cases as this is a fundamental problem across various vision problems.

References

[1] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *European conference on computer vision*, pages 611–625. Springer, 2012. 2, 5

[2] Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual kitti 2. *arXiv preprint arXiv:2001.10773*, 2020. 5

[3] 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. 1, 2, 5

[4] Risheek Garrepalli. Oracle analysis of representations for deep open set detection. *arXiv preprint arXiv:2209.11350*, 2022. 2

[5] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 2, 5

[6] 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 *European Conference on Computer Vision*, pages 668–685. Springer, 2022. 2, 3

[7] Junhwa Hur and Stefan Roth. Iterative residual refinement for joint optical flow and occlusion estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5754–5763, 2019. 2

[8] 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. 1, 2

[9] Jisoo Jeong, Hong Cai, Risheek Garrepalli, and Fatih Porikli. Distractflow: Improving optical flow estimation via realistic distractions and pseudo-labeling. *arXiv preprint arXiv:2303.14078*, 2023. 2

[10] Jisoo Jeong, Jamie Menjay Lin, Fatih Porikli, and Nojun Kwak. Imposing consistency for optical flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3181–3191, 2022. 1, 2, 3

[11] Daniel Kondermann, Rahul Nair, Katrin Honauer, Karsten Krispin, Jonas Andrulis, Alexander Brock, Burkhard Gussefeld, 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. 2

[12] Si Liu, Risheek Garrepalli, Thomas Dietterich, Alan Fern, and Dan Hendrycks. Open category detection with PAC guarantees. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3169–3178. PMLR, 10–15 Jul 2018. 2

[13] Si Liu, Risheek Garrepalli, Dan Hendrycks, Alan Fern, Debashis Mondal, and Thomas G Dietterich. Pac guarantees and effective algorithms for detecting novel categories. *J. Mach. Learn. Res.*, 23:44–1, 2022. 2

[14] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 5

[15] 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. 2, 5

[16] 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. 2, 5

[17] 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. 2

[18] Lukas Ruff, Jacob R Kauffmann, Robert A Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109(5):756–795, 2021. 2

[19] Simon Schrodi, Tonmoy Saikia, and Thomas Brox. Towards understanding adversarial robustness of optical flow networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8916–8924, 2022. 2

[20] Krishna Shankar, Mark Tjersland, Jeremy Ma, Kevin Stone, and Max Bajracharya. A learned stereo depth system for robotic manipulation in homes. *IEEE Robotics and Automation Letters*, 7(2):2305–2312, 2022. 1

[21] Sangeetha Siddegowda, Marios Fournarakis, Markus Nagel, Tijmen Blankevoort, Chirag Patel, and Abhijit Khobare. Neural network quantization with ai model efficiency toolkit (aimet), 2022. 7

[22] Deqing Sun, Charles Herrmann, Fitsum Reda, Michael Rubinstein, David Fleet, and William T Freeman. What makes raft better than pwc-net? *arXiv preprint arXiv:2203.10712*, 2022. 3

[23] Deqing Sun, Daniel Vlasic, Charles Herrmann, Varun Jampani, Michael Krainin, Huiwen Chang, Ramin Zabih, William T Freeman, and Ce Liu. Autoflow: Learning a better training set for optical flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10093–10102, 2021. 3

[24] 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. 2

[25] 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. 1, 2, 4, 5, 6

[26] 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

[27] 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. 1, 2, 3, 4

[28] 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. 1, 2