LEARNING OPTICAL FLOW WITH AUXILIARY COST AGGREGATION

by

Chengyao Xiao

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
November 2023

Table of Contents

List of Tables

List of Figures

Abstract

Optical flow represents motions for each pixel between two adjacent frames in a video sequence. Deep learning-based estimation approaches for optical flow have overshadowed the variational approaches over the past few years, as they achieve real-time estimation with reduced estimation error. The construction of deep learning-based estimation models heavily relies on the cost volume which is constructed through matrix multiplication and encodes the dense matching information between the given inputs. Long-range correlation and occlusion, however, remain challenging as information drawn from the cost volume is heavily weighted by the local correlation defined over a fixed window size. In this thesis, we propose to enrich the information used for the iterative residual flow decoding process with an Auxiliary Cost Aggregation (ACA) unit that constructs an auxiliary cost volume based on the top-k matches from the 4D cost volume and then augments it using Transformers. Additionally, a post-refinement module is also proposed to refine the predicted residual flow at the end of each iteration based on the feature's local coherence. Extensive experiments indicate that our model achieves better cross-dataset generalizability than two baseline models, RAFT and GMA. On the Sintel and KITTI benchmarks, our model outperforms RAFT and has comparable performance with other state-of-the-art (SOTA) models.

Chapter 1

Introduction

Optical flow is defined over 2D space and it represents the per-pixel motion within a given consecutive image pair. The correspondence information it provides facilitates many real-world applications, such as video frame interpolation [5, 54, 53], video super-resolution [69, 14, 60], and video inpainting [77, 38, 88].

The era of deep learning-based optical flow estimation models started with the work presented by FlowNet [18] that computes optical flow with stacks of end-to-end trainable convolutional neural networks (CNNs) [40]. PWC-Net [62] then achieved a notable performance increase compared to FlowNet by introducing an end-to-end model equipped with cost volume and backward-warping operations which are then applied by Zhao *et al.* [85], Hui *et al.* [28], and Jiang *et al.* [33]. The success of PWC-Net shows that cost volume computed by matrix multiplication between CNNs encoded feature maps is crucial to the architecture design for deep learning-based approaches as it encodes the feature matching information which provides guidance for the 2D motion estimation. Therefore, model design subsequent to PWC-Net all includes cost volume as part of the computation process. Compared to the PWC-Net, instead of using stacks of coarse-to-fine CNNs pyramid, a significant improvement was made by RAFT [66] which uses a convolutional gated recurrent unit (ConvGRU) as an iterative residual flow decoder and adopted table lookups through the cumulative residual flow at each iteration on the 4D cost volume of shape $H \times W \times H \times W$, where H and W are the spatial dimensions of the encoded feature maps.

As the RAFT model revolutionized the field of optical flow estimation, it is adopted as the baseline model for many of the following deep learning-based optical flow estimation models [61, 33, 86, 64, 34, 84, 27]. Works by Jiang *et al.* [33], Sun *et al.* [64], Zheng *et al.* [87], and Bai *et al.* [3] mainly focus on the recurrent decoder part of RAFT, whereas models proposed by Zhang *et al.* [84], Jiang *et al.* [34], Xu *et al.* [74], and Xiao *et al.* [72] emphasize the cost volume construction which is also the direction we have taken when constructing our model.

The RAFT model does not handle occlusion which occurs when the pixels in the first image do not have a matching target in the second image. Additionally, the full 4D cost volume introduces unnecessary matching information. The GMA [33] model augments RAFT by handling occlusion with information propagation based on the feature's intra-similarity, the proposed component is then applied by several approaches [61, 64, 27]. Zhang *et al.* [84] break the per-pixel matching 4D cost volume into two smaller 4D cost volumes of shape $H \times W \times U \times K$ and $H \times W \times V \times K$, where H and W are the spatial dimensions of the feature map, and U, V, and K are hyperparameters, encoding horizontal correlation and vertical correlation respectively. Cost aggregation is then performed on the two cost volumes. Xu *et al.* [74] reduces the cost volume to two 3D tensors of shape $H \times W \times D$, where one encodes horizontal matching information and the other one encodes vertical matching information by using 1D attentions and 1D correlation. Xiao *et al.* [72] keeps the full 4D cost volume and introduces a Caylay representation to learn the cost volume. Recent works demonstrated in SCV [34] and FlowFormer [27] reduce the size of the full 4D cost volume from the shape of $H \times W \times H \times W$ to a 3D shape of $H \times W \times K$ using k-nearest neighbor selection and a learnable latent tensor, respectively. SCV shows that iterative optical flow estimation can be done by only using a sparse cost volume, however, its performance is inferior to RAFT as the ambiguity caused by textureless and motion-blurred regions requires dense matching to determine the pixels' motion. On the other hand, FlowFormer preserves the full 4D cost volume while using the embedded 3D cost volume to provide extra information during the decoding process. It greatly boosted the estimation performance but it is computationally heavy as the 3D cost volume is computed by applying dot-product attention between the latent tensor and full cost volume.

As the cost volume is a crucial part of the deep learning-based approaches, the information presented in it should be captured as much as possible during the optical flow estimation process. Although current approaches have made efforts to enrich the cost volume by decomposition to encoding horizontal information and vertical information separately or through condensation which tries to capture only important information and filter out noise, the resulting model is either computationally heavy or has reduced performance compared to the RAFT model. Therefore, in this thesis, we propose an approach that preserves the benefits of SCV and FlowFormer. Inspired by SCV, we take a subset of correspondences from the full 4D cost volume by applying the *topk* operation which selects k most significant correlations, and linear self-attention is applied on the embedded *topk* matches to aggregate information presented by each of them. After the construction of the auxiliary cost volume, similar to FlowFormer, both the full cost volume and the auxiliary cost volume are used together during the residual optical flow prediction stage.

Compared to those two models, we preserve the full 4D cost volume computed using dot-product between two encoded feature maps to provide the full matching information between two images which benefits the prediction of textureless regions as well as motion-blurred regions. The auxiliary cost volume that contains the top matching candidate is used along with the full cost volume to enrich the information provided to the decoding stage dynamically.

Local refinement is often applied in the depth estimation [80, 41, 20] as depth maps are piecewise smooth. Additionally, based on the smoothness assumption widely made in variational approaches regarding optical flow estimation [26, 8, 81], where motion is shared within a local neighborhood, we also propose to propagate the hidden state based on the feature's local similarity at the end of each recurrent iteration.

We summarize our contributions and performance as follows:

- **An auxiliary cost aggregation unit** is introduced that efficiently constructs an auxiliary cost volume using the top-k correlations from the full 4D cost volume and dynamically integrates it into the residual optical flow decoding process.

- **A post-refinement process** is proposed that propagates the information of each updated hidden state based on the image's local coherence.

- **Compatible performance on cross-data generalization** to SOTA models which outperforms both RAFT [66] and GMA [33] on the KITTI [52] benchmarks while having performance that is on par with SOTA models on the Sintel [13] final benchmarks.

Chapter 2

Related Work

This chapter briefs the related approaches that have been applied to the optical flow estimation task. The first section mentions some of the most significant work done using variational approaches. Backward wrapping and hierarchical estimation approaches are detailed as these two techniques are still widely used since the early 80s. Then we move on to the deep learning approaches which have greatly boosted the estimation accuracy over the variational approaches. Methods in both supervised learning and unsupervised learning are described. Although supervised learning models perform better than unsupervised ones, unsupervised models provide insight into how occluded pixels can be handled explicitly within a deep learning framework. We also give a shallow description regarding attention models [68] for Natural Language Processing (NLP) as well as its variations that are more suitable for vision tasks as they achieve better performance on some tasks that were dominated by the convolutional neural network (CNN) based deep learning models.

2.1 Variational Approaches

The seminal work presented by Horn and Schunck (HS) [26] with brightness constancy and smoothness assumptions has built the foundation for the majority of non-deep learning-based optical flow estimation models. Given a pixel located at position (x, y) in an image I, the brightness constancy assumption states that optical flow does not change the intensity of a pixel over a small time interval,

$$I(x+u, y+v, t+\delta t) = I(x, y, t), \tag{2.1}$$

where u is the motion regarding x-axis (horizontal) and v is the motion regarding y-axis (vertical). Using first-order Taylor expansion, the above equation becomes

$$\begin{aligned} I(x, y, t) + I_x u + I_y v + I_t &= I(x, y, t) \\ I_x u + I_y v + I_t &= 0 \end{aligned} \tag{2.2}$$

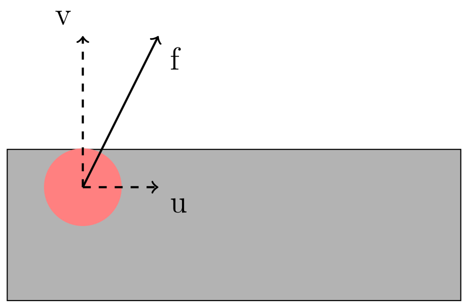

Figure 2.1: An illustration of the aperture problem: The red object has a motion in the direction of f over the grey background which can be decomposed to u in the horizontal direction and v in the vertical direction. For a small time step dt, the image gradient in the vertical direction changes along v and image gradient in the horizontal direction is perpendicular to u which resulted in $I_y v \neq 0$ and $I_x u = 0$. Thus, the motion f having its horizontal component u unregistered under dt.

where I_x and $I_y$are the gradient over the horizontal and vertical directions on the image I, respectively. I_t is the temporal gradient computed between the input sequence.

The model then tries to minimize the energy function below defined based on Equation 2.2,

$$E_b(x, y) = \phi(I_x u + I_y v + I_t) = (I_x u + I_y v + I_t)^2, \tag{2.3}$$

where ϕ is the penalty function and chosen to be $\phi(x) = x^2$ in their setting.

The brightness constancy assumption provides a single equation with two unknowns, additionally, the aperture problem stated in Hutchinson *et al.* [29] and illustrated in Figure 2.1 indicates that $I_x u$ and $I_y v$ may become invalid under certain conditions. Therefore, the local smoothness assumption E_s which states that neighboring points of (x, y) should possess similar motion is also added to the model,

$$E_s = u_{xx} + u_{yy} + v_{xx} + v_{yy}, \tag{2.4}$$

with the second-order horizontal and vertical motion gradients computed with the standard five-point stencil. The minimization of the summation of Equation 2.3 and Equation 2.4 is conducted in an iterative manner with u and v both initialized as 0s.

Before deep learning-based approaches, the basic formulation of optical flow estimation using variational approaches mostly follows the above two assumptions. The penalty function $\phi(x) = x^2$ used in HS is further replaced with the L1 norm

$\phi(x) = |x|$ [81], the Charbonnier penalty $\phi(x) = \sqrt{x^2 + \epsilon^2}$ [12], or the Lorentzian $\phi(x) = log(1 + \frac{x^2}{2\epsilon^2})$ [8]. In addition, the L1 norm is usually equipped with the total variation [58] and minimizing the following energy function,

$$E_\theta = \sum_d |\nabla u_d| + \sum_d \frac{1}{2\theta}(u_d - v_d)^2 + \lambda|\rho(u)|, \tag{2.5}$$

where d is the dimension of the flow vector, $|\nabla u_d|$ is the smoothness term, $\rho(u)$ is the brightness constancy term, and v is the introduced auxiliary variable that is a close approximation of optical flow u [81]. The median filter can be applied to discard outliers after each incremental optical flow result [71]. Furthermore, to mitigate lighting changes, Rudin-Osher-Fatemi (ROF) structure texture decomposition method [58] is used to pre-process the input sequence [71].

Although most of the optimization techniques developed under the variational approaches are not suitable for deep learning-based optical flow estimation models, two common practices are widely applied in state-of-the-art (SOTA) works: warping and iterative hierarchical (multi-resolution) structure.

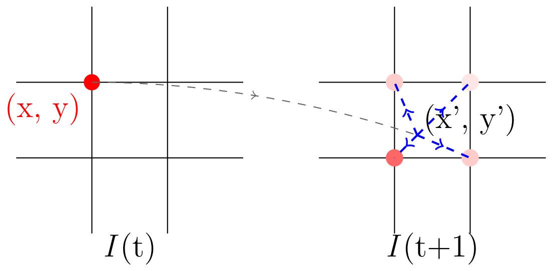

Figure 2.2: Forward warping: Based on the estimated flow $f_{t\to t+1}$, pixel located at (x, y) in $I(t)$ is mapped to pixel located at (x', y') in $I(t + 1)$ followed by bilinear splatting.

A warping operation is adopted in the majority of optical flow estimation models. Forward warping used by Bergen *et al.*[7] maps pixels located in $I(t)$ to those located in $I(t + 1)$ based on the estimated flow as shown in Figure 2.2. For a given pixel at position (x, y) in $I(t)$, it is mapped to (x', y') in $I(t + 1)$ according to flow vector $f_{t\to t+1}$. Since $f_{t\to t+1}$ are not guaranteed to be integer values, bilinear interpolation is used to divide the intensity of $I_t(x, y)$ into four parts with weighting $\sum_{i=1}^4 w_i = 1$, where each w_i is calculated as,

$$\begin{aligned}
x_1 &= \lfloor x' \rfloor \\
x_2 &= x_1 + 1 \\
y_1 &= \lfloor y' \rfloor \\
y_2 &= y_1 + 1 \\
w_0 &= (y_2 - y') * (x_2 - x') \\
w_1 &= (y' - y_1) * (x_2 - x') \\
w_2 &= (y_2 - y') * (x' - x_1) \\
w_3 &= (y' - y_1) * (x' - x_1),
\end{aligned} \tag{2.6}$$

with w_0 corresponding to the coordinate at the southwest corner and the rest arranged clockwise, the larger the w_i, the more contribution it will take from $I_t(x, y)$ (corresponding to darker color in Figure 2.2). Forward warping is having less impact on optical flow estimation models as the warped image will contain holes if there are multiple pixels mapped to the same location [65]. Additionally, the forward warping operation is not easily differentiable as a z-buffer is usually required [53], where the z-buffer stores all the contributing pixels and orders them based on their depth. Fully differentiable forward warping is proposed by Niklaus *et al.* [54] for video frame interpolation, but it is unclear how to apply this operation to optical flow estimation tasks.

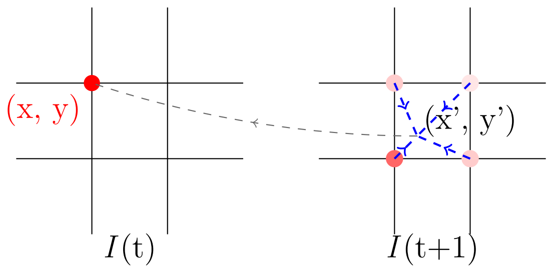

Figure 2.3: Backward warping: Based on the estimated flow $f_{t\to t+1}$, bilinear interpolation is first performed at location (x', y') in $I(t+1)$, the interpolated value is then mapped back to location (x, y) in $I(t)$.

Backward warping or backward sampling as shown in Figure 2.3, on the other hand, performs sampling in $I(t+1)$ based on $f_{t\to t+1}$, the resulting image represents pixels in $I(t+1)$ moved backward by $f_{t \to t+1}$. For a pixel located at (x, y) in $I(t)$, its intensity value is obtained by performing bilinear interpolation at location (x', y') in $I(t+1)$ first and then this value is mapped back to $I(t)$ [12]. Different from the forward warping that splats the intensity of $I(x, y)$, all four integer coordinates surrounding (x', y') contribute to the intensity at (x, y), additionally, under the iterative hierarchical flow estimation approach, only the flow between the original $I(t)$ and the warped $I(t+1)$ are estimated [11] at each iteration.

Large motions presented in an image sequence are captured by using a coarse-to-fine (hierarchical) warping approach [51, 8]. Image pyramid with different resolution is constructed first, starting from the level with the lowest resolution where optical flow are initialized as 0s, the flow estimation at each level is computed between the first image and the warped second image, where warping operation uses the flow computed at the previous level (coarser level).

2.2 Deep Learning Approaches

Deep learning based per pixel prediction tasks gained attention since Convolutional Neural Network (CNN) based segmentation network [46] had great performance gain over previous works. In this section, a number of deep learning-based optical flow estimation models will be introduced.

2.2.1 Supervised Approaches

Flownet proposed by Dosovitskiy *et al.* [18] uses a stack of convolutional layers, and nonlinear activation functions to predict optical flow given two input images. Two different models are proposed in this work. FlowNetSimple which concatenates the two images in their channel dimension and then feeds to a sequence of CNNs for optical flow prediction and FlowNetCorr which extracts features maps which are the output of CNNs from two input images independently and then predicts optical flow based on their correlations. Both of them are based on the encoding and decoding design [57] where the spatial sizes on the xy-plane are reduced during the encoding stage, the encoded feature map is then enlarged during the decoding stage.

The encoding stage for FlowNetSimple is designed with nine convolutional layers with kernel size of 7×7 for the first layer, 5×5 for the following two layers, and 3×3 for the rest of the layers. Feature maps that are the output of convolutional layers are produced with reduced spatial dimensions and increased channel dimensions after each layer. The output from FlowNetSimple's encoding stage is a feature map that encodes information with respect to the concatenated inputs. Compared to FlowNetSimple, the number of encoding convolutional layers is the same for FlowNetCorr architecture but the encoding architecture is different. Two feature maps $f_1, f_2 \in R^{C_f \times H \times W}$ with C_f, H, and W representing the number of encoded channels, feature map's height, and feature map's width, separately, are extracted by using the first three convolutional layers in a Siamese fashion [9] where the same network is used on two different inputs for feature extraction. Given f_1 and f_2, correlations between them are computed in patch-wise manner,

$$c(x_1, x_2) = \sum_{o \in [-k,k]^T \times [-k,k]} \langle f_1(x_1 + o), f_2(x_2 + o) \rangle, \tag{2.7}$$

where $x_2 \in [x_1 - d, x_1 + d]^T \times [x_1 - d, x_1 + d]$ is defined to limit the search window size as well as reduce the number of computations required. The resulting correlation $Corr \in R^{H \times W \times (2d+1)^2}$ is the input to the rest of the encoding layers. The resulting feature map for the FlowNetCorr after the encoding stage represents the correlations between f_1 and f_2. In order to recover details from the reduced resolution, the decoding process uses deconvolution [82] which gradually increases the spatial resolution of the output feature map. Additionally, an intermediate optical flow estimation is produced at each deconvolution step using the concatenation among feature map f_i from the previous deconvolution, coarser flow from the previous deconvolution, as well as the feature map from the encoding stage whose resolution is the same as f_i.

The training data scheduling approach proposed in FlowNet2 [30] is used by all the SOTA optical flow approaches. The network trained on the easier Chairs dataset first[18] and then trained on the harder dataset FlyingThing3D [49]. This type of training where tasks become increasingly harder follows the curriculum learning approach [6]. To integrate the successful iterative estimation approach from variational approaches into their network, multiple networks are stacked together. FlowNetCorr is used as the start of the stack where it takes in two images as the input and then predicts optical flow f_i^c where i is the network stack index. FlowNetSimple follows the FlowNetCorr but instead of taking stacked images as the input, it takes the stack of first image I_1, second image I_2, back-warped second image $\hat{I}_2$, optical flow f_i^c, and the difference ΔI between I_1 and $\hat{I}_2$ as the input which then outputs optical flow f_i^s. Additionally, the network stack for estimating large motion displacement f_l consists of one FlowNetCorr followed by two FlowNetSimples. To produce more crisp motion boundaries, a deeper FlowNetSimple is applied as an additional branch, and it produces smaller motion displacement f_s. The final optical flow is produced with a shallow fusion network that takes the magnitude of f_l, the magnitude of f_s, optical flow f_l, optical flow f_s, the brightness difference between I_1 and $\hat{I}_2$ which is the backwarpped I_2 using f_l, and brightness difference between I_1 and $\hat{I}_2$ which is the backwarpped I_2 using f_s.

FlowNet shows that real-time optical flow estimation can be approached using deep learning models but their performance was inferior to the best variational approaches. FlowNet2 has better performance but it requires a large memory footprint as it requires five networks in total. PWC-Net [62] gained accuracy and reduced memory consumption by applying an end-to-end differentiable iterative hierarchical warping technique. The feature pyramid which contains feature maps of different resolutions is constructed with a stack of six convolutional layers that reduce the spatial dimension of the input by a factor of 2 at each layer. The warping operation is conducted among feature maps. Starting from the coarse level i (low resolution), optical flow f_{i-1} from the previous level is upsampled by a factor of 2 to $\hat{f}_{i-1}$, and $\hat{f}_{i-1}$ is used to warp feature map f_2^i produced by the second image at the level i to $\hat{f}_2^i$ using the bilinear interpolation operation provided in the spatial transformer network [31]. Cost volume cv_i that encodes correlation information between feature maps f_1^i and $\hat{f}_2^i$ is constructed the same way as the Equation 2.7 which outputs a 3D tensor of shape $H_i \times W_i \times (2d+1)^2$. The optical flow estimator is implemented as a multi-layer CNN. At each level, it takes the first feature map f_1^i, the upsampled flow from the previous level $\hat{f}_{i-1}$, and the cost volume cv_i as input and output optical flow f_i at the current scale. Additionally, to refine the estimated flow, a separate network is constructed using a sequence of dilated convolutional layers [78] which takes in the output from the last convolutional layer of the optical flow estimator and outputs refined optical flow. The refined optical flow is then summed with the optical flow with the highest resolution obtained through the optical flow estimator as the final result. Compared to FlowNet2 which has 162M parameters, PWC-Net has only 8.75M parameters and outperformed all the other models when it was released.

Since cost volume is constructed by matrix multiplication between two feature maps and it encodes the per feature matching information. This is essentially a search space defined between each pixel in the first image and every pixel in the second image. Xu *et al.* [76] restricted the search field in the feature map produced by the second image to a window of size $(2 * d + 1)^2$ for each feature in the feature map produced by the first image and then output optical flow by using semi-global matching (SGM) algorithm [25] on the cost volume of size $H \times W \times (2*d+1)^2$, where H and W are the spatial resolution of the first feature map. To obtain a cost volume with less noise, Hui *et al.* [28] applied affine transformation on the cost volume to filter outliers,

$$cv_m = \alpha \otimes cv \oplus \beta, \tag{2.8}$$

where all operations are performed element-wise and cv_m is the modulated cost volume which is used for optical flow estimation. α and β in the equation above are generated by passing the cost volume cv, confidence map, and the feature map of the first image to several layers of CNNs. Although all the aforementioned models require the use of cost volume to provide correlation information for the network to produce reasonable optical flow estimation, computing a matrix multiplication at each layer is a costly operation. The RAFT [66] optical flow estimation model which only computes the cost volume once and decodes it using Recurrent Neural Networks (RNN) [59] gained 30% error reduction on Sintel dataset compared to the best-published result at its publication.

RAFT model extracts feature maps $\{f_1, f_2, f_c\} \in R^{H\times W\times C_f}$ which are 1/8 of the size of the input images using residual convolutional layers with skip connections[24], where f_1 is the feature map for the first image, f_2 is the feature map for the second image, and f_c is the context feature extracted from I_1 that will be used in the flow decoding stage, moreover, all the three features are the output of a sequence of CNNs. Cost volume is then computed as the matrix multiplication $cv = f_1^T f_2$ which is a 4D tensor of shape $H \times W \times H \times W$, the multiplication is performed by first shaping both f_1 and f_2 to 2D tensors of size $HW \times C_f$, then producing a tensor of size $HW \times HW$ through matrix multiplication, and reshaping the multiplication result to a 4D tensor of shape $H \times W \times H \times W$ which encodes the matching information between every feature in f_1 and every feature in f_2. The RNN then iteratively decodes residual flow $f^i_{1\to 2}$ at each time step i using motion feature f_m which is computed by taking samples from cv, and the final optical flow is computed as $f_{1\to 2} = \sum_{i=0}^{T} f^i_{1\to 2}$, where T is the number of recurrent steps. Sampling locations from cv are determined based on the current optical flow $f^t_{1\to 2} = \sum_{i=0}^{t} f^i_{1\to 2}$ with $t \leq T$, given the sampling location, the cost volume is then reshaped to a size of $HW \times H \times W$ and the sampling is performed among the last two dimensions of the cost volume. The entire sampling process for each feature in f_1 can be considered as taking that single feature and move it around in the entire feature space of f_2, and the moving direction is based on $f^t_{1\to 2}$. At each time step, samples of size $H \times W \times (2d + 1)$ are drawn from the cost volume, so that the cost volume only needs to be computed once during the estimation pipeline, additionally, since parameters in RNN are shared across all time steps, the model only requires 5.5M number of parameters. The boosted estimation performance and reduced model size make RAFT the baseline model for many of the following works.

To improve the matching accuracy regarding the cost volume, Luo *et al.* [47] proposed to fuse information between f_c and f_m. Given $\{f_c, f_m\} \in R^{H\times W\times C}$, context node v_c and motion node v_m are created by projecting $\{f_c, f_m\}$ to a shape of $C \times K$ with K representing the number of nodes, the projection step is done by passing both of them a shallow neural network. The node adjacency matrix A between v_c and v_m allows image context information to be propagated to the motion nodes with the assumption that predicted flows can be constrained by the image structure, A is computed as,

$$A = Project(v_m, \Theta(v_c)), \tag{2.9}$$

where Θ computes the matrix multiplication $v_c^T v_c$ followed by softmax activation function and results in a tensor of shape $K \times K$, the project function here is also a shallow neural network. Then the refined motion node v'_m is obtained by passing v_m and A to adaptive graph convolutional network [42]. The enhanced context node is computed following Kipf *et al.* [39],

$$\begin{aligned} v'_c &= \phi(A' v_c^T w_G) \\ A' &= v_c^T v_c, \end{aligned} \tag{2.10}$$

where w_G are learnable graph convolution parameters and ϕ is again a projection layer implemented using a linear neural network. Both v'_m and v'_c are projected back to the original feature space of shape $H \times W \times C$ and used for the iterative flow decoding stage.

Cost Volume Reduction for Iterative Optical Flow Decoding Approach

To reduce the size of the cost volume and allow non-local information to be learned during the recurrent decoding stage, SCV [84] proposed to separate the cost volume construction. Instead of constructing a cost volume cv of size $H \times W \times H \times W$ as in RAFT [66], a cost volume $\hat{cv}$ of size $H \times W \times U \times V$ is constructed where U is the range of horizontal motion and V is the range of vertical motion, both of them are the hyperparameters representing the range of motion determined before the training. Two smaller cost volumes $cv_u \in R^{H \times W \times U \times K}$ for horizontal motion and $cv_v \in R^{H \times W \times V \times K}$ for vertical motion are extracted from $\hat{cv}$, with $K \ll U, V$. The first two channels of K are computed as,

$$cv_u^1 = \frac{1}{V} \sum_{v \in V} \hat{cv}(i, j, u, v), \tag{2.11}$$

$$cv_u^2 = \max_{v \in V} \hat{cv}(i, j, u, v). \tag{2.12}$$

The first channel for cv_u is computed as the average over all values in the vertical dimension and the second channel is computed as the max element from the vertical dimension, the same computation is done on cv_v to compute cv_v^1 and cv_v^2. For the construction of the rest of $K-2$ channels in cv_u, a weighting term $A_u \in R^{H \times W \times V \times (K-2)}$ is obtained by passing $[cv_u^1, cv_v^2]$ to a 3D convolutional layer followed by a softmax operation over the third dimension V, then a matrix multiplication between reshaped $A_u \in R^{H \times W \times (K-2) \times V}$ and $\hat{cv} \in R^{H \times W \times V \times U}$ resulting in a tensor of shape $H \times W \times (K-2) \times U$ which contains information for the $K-2$ channels, and the above processes are repeated again for the construction of the vertical cost volume. A stack of differentiable SGM layers [83] and 3D convolutional layers are applied to cv_u and cv_v which allows them to contain non-local information and reducing their size by removing the K channels which results in shapes $H \times W \times U$ and $H \times W \times V$, respectively. The recurrent optical flow decoding process is then applied on cv_u and cv_v to produce the flow estimation.

The cost volume constructed as a matrix multiplication between two feature maps requires storing a tensor of size $H \times W \times H \times W$, but only a few matching information in the second feature map are crucial to a single feature from the first feature map. Although the separable cost volume proposed by Zhang *et al.* [84] uses two smaller cost volumes, they still try to cover as much matching information as possible. Cost volume construction based on k-nearest neighbors (kNN) is then proposed by Jiang *et al.* [34] to only focus on matching a few points for each point in the first image. After feature extraction, kNN [35] is used to compute k points in the second feature map for each point in the first feature map. The cost volume then contains $H \times W \times k$ elements where $k \ll H \times W$ with $k = 8$. Thanks to the reduced size of the cost volume, this model can increase the size of the feature maps to 1/4 of the input image under the same computational power as RAFT. The approach taken by this work shows that optical flow estimation could be done with only k matches from the cost volume but the sparsity of the constructed cost volume failed to provide accurate correlation information when the input images contain noisy textures such as motion blur and shadow. This has inspired us to design a model that considers both the dense cost volume as well as the sparse but significant correlation pairs.

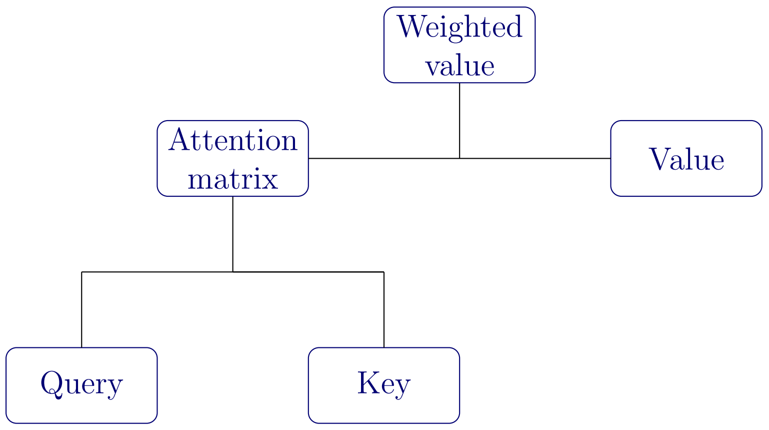

Figure 2.4: Basic attention model: Attention matrix is computed by performing matrix multiplication between the query and the key tensors followed by softmax operation and the output is computed by performing another matrix multiplication between the attention matrix and the value tensor.

Attention-based Iterative Optical Flow Decoding Models

The multi-head attention model proposed by Vaswani *et al.* [68] and shown in Figure 3.5 where each head is a portion of the feature map in its channel dimension was designed for NLP that allows information to be propagated to different locations based on their similarities. The attention computation takes in a sequence of encoded tokens of shape $N \times K$, where N is the number of tokens and K is the size of the hidden dimension, the encoded tokens are usually the output of a sequence of linear projection layers with learnable parameters. Moreover, the attention computation can be categorized as self-attention and cross-attention. In self-attention, query, key, and value all come from the same input source, whereas in cross-attention, these three values may come from different inputs. The attention matrix has the shape of $N \times N$ which is the result of matrix multiplication between query tokens of shape $N \times K$ and key tokens of shape $K \times N$. The attention matrix encodes the matching score between each query token and each key token, and it is used to augment the value tokens. In the field of NLP, the token refers to each word in the input sequence, whereas in the field of computer vision, the token refers to each patch within the input image.

In recent years, there has been a surge in using transformers designed for image processing as the backbone model for vision-related tasks as they are able to capture long-range dependencies. ViT proposed by Dosovitskiy *et al.* [17] breaks the image into a sequence of patches $x_p \in R^{N \times (P^2 \cdot C)}$, where N is the number of patches, P is the patch size, and C is the channel dimension for the input image. Then all the patches are projected to a high dimension using Multiple Layer Perceptron (MLP) and passed to a position-embedded multi-head self-attention model (MSA). A single head x_p^i is computed as

$$\begin{aligned} x_p^i &= MLP(x_p^i) + PE(x_p^i) \\ MSA(X) &= MLP(softmax(X^T X)X) \\ \hat{x_p^i} &= MSA(x_p^i) + x_p^i \\ \hat{x_p^i} &= MLP(\hat{x_p^i}) + \hat{x_p^i} \end{aligned} \tag{2.13}$$

where PE is the position embedding to encode position information for each patch and the final result after MSA is the concatenated $\hat{x_p^i}, i \in [0, ..., N]$ with N as the number of heads.

To reduce the amount of computation during self-attention, Swin-transformer[45] divides the projected image to non-overlapping windows and computes attention within each local window, then a shift operation is performed that shifts pixel positions by half of the window size to achieve cross-window information propagation. Twins [15] reduces the self-attention computation by first computing attention within each windowed feature which is referred to as locally-grouped self-attention (LSA). LSA is then followed by global sub-sampled attention (GSA) which uses CNN to summarize each feature window of size $P \times P \times C$ to a shape of $1 \times 1 \times C$ and then using the summarized feature as key in the self-attention computation. The ability to encode long-range dependency information makes the attention module a widely used component during the design of optical flow estimation models.

Based on the iterative optimization step used in RAFT [66], Xu *et al.* [74] uses 1D attention to reduce the size of the cost volume over its last two channels. Regarding 1D attention in the vertical direction, the attention matrix is generated between the first feature map F_1 with respect to the first image and the second feature map F_2 with respect to the second image which resulted in a tensor of shape $W \times H \times H$, where H and W are the spatial dimensions of the feature maps which are the same for both of them. Using F_2 as value, the final tensor from 1D vertical attention has the shape of $H \times W \times C_f$, where C_f is the number of features, 1D horizontal attention is computed in the same manner except the resulted attention matrix has the shape of $H \times W \times W$. Then correlation is computed between F_1 and the vertically attended F_2 as well as the horizontally attended F_2, two cost volumes of the same size $H \times W \times (2R + 1)$, where R is the correlation radius, are concatenated over the last dimension for the iterative optical flow optimization stage. Although memory consumption can be reduced greatly when the image resolution increases, the performance is inferior to that of RAFT.

Since the cost volume construction proposed in RAFT [66] is performed in the same manner as the computation for the attention matrix in the Attention model [68], relative position embedding is added to the cost volume in CRAFT model [61]. The relative position bias [19] $B \in R^{(2r+1)\times(2r+1)}$ is a learnable tensor that is added to the cost volume cv with r being the lookup radius, for each pixel (i, j) in the first two dimensions of cv,

$$cv'(i,j,i+x,j+y) = \begin{cases} cv(i,j,i+x,j+y) + B(x,y) & \text{if } |x| \le r,\ |y| \le r \\ cv(i,j,i+x,j+y) & \text{otherwise} \end{cases} \tag{2.14}$$

where cv' is the cost volume encoded with relative position embedding. They also applied Expanded Attention from Li *et al.* [43] to further encode the second feature map F_2 before passing it to the cost volume computation.

GMflowNet proposed by Zhao *et al.* [86] uses patch-based overlapping attention to diversify the encoded information in both feature maps after features extracted by convolutional layers to reduce the estimation error regarding large motions. Each of those feature maps is divided into $M \times M$ non-overlapping patches, multi-head attention is performed between patch $A \in R^{M^2 \times C_f}$ and the 3×3 patches $B \in R^{9M^2 \times C_f}$ that are centered by it, using A as query and B as both key and value and output refined patch $A' \in R^{M^2 \times C_f}$. Then the refined patches are merged back to the shape of the original feature map for cost volume construction and recurrent flow prediction. To provide better starting sampling positions, they used the dual-softmax operation [63, 56, 67] to compute matching probability from the cost volume and use the matched position as the starting point for cost volume sampling.

A transformer-based model is proposed by Huang *et al.* [27] which encodes the cost volume to a more compact representation and then iteratively decodes residual flow from the original cost volume and the compressed one. A convolution layer with a stride of 2 is applied on the last two dimensions of the cost volume to produce an embedding for each 2×2 patch which resulted in a cost volume of size $cv_p \in R^{H \times W \times H/8 \times W/8 \times D}$ with D being the embedding dimension. Based on the same argument made by Jiang *et al.* [34] where cost volume contains redundant information, a latent tensor of size $C \in R^{K \times D}$, where K is a hyperparameter determined based on trail-and-error, is used to compress the information contained in the cost volume,

$$\begin{aligned} Key &= Linear(cv_p) \\ Val &= Linear(cv_p) \\ T &= Attention(C, Key, Val), \end{aligned} \tag{2.15}$$

where Key and Val are computed by applying linear projection on the position embedded cv_p, and T has a shape of $H \times W \times K \times D$. To reduce the computation complexity of the *Attention* operation, two self-attentions are applied. The first one computes the intra-self-attention on the reshaped tensor of $H \times W \times (K \times D)$ and the result is reshaped to $K \times (H \times W \times D)$ for the inter-self-attention computation. During the decoding operation, cross-attention is then applied between the sampled cost volume and T with samples being query and T being both key and value. To further push the performance of the model, CNN is changed to pretrained Twins [15] for better feature extraction. Although their approach provides a great way to enhance the correlation information, their attention model is computationally heavy and cannot be easily applied to a single GPU machine, we share a similar cost volume information propagation approach but using a lighter attention model. Additionally, since none of the above deep learning approaches considers the post-refinement after each recurrent iteration, we propose a local coherent post-refinement process in our model with the aim of a more accurate residual flow estimation.

2.2.2 Unsupervised Approaches

Large datasets with annotated ground truth labels are not easily acquirable which makes unsupervised learning of optical flow a prominent research direction. Supervised learning is constrained by a loss function that directly compares the difference between the predicted result and the ground truth, without the ground truth labels, loss functions used to constrain the unsupervised models are significantly different than the supervised ones.

The approach taken by Yu *et al.* [79] uses the FlowNetSimple [18] architecture and trains the network using the photometric loss as well as the smoothness loss to constrain the network, both loss functions are almost identical to the energy function used in HS approach [26],

$$\begin{aligned} L &= L_{pho} + L_{smo} \\ L_{pho} &= ((I_1 - W(I_2))^2 + \epsilon^2)^\alpha \\ L_{smo} &= u_{xx} + u_{xy} + v_{xx} + v_{xy}, \end{aligned} \tag{2.16}$$

where $\{\epsilon, \alpha\}$ are hyperparameters, L is the final loss function, $\{u_{xx}, u_{xy}, v_{xx}, v_{xy}\}$ are gradients from the predicted flow, and W is the the backwarping operation. The loss function is applied on every intermediate flow during the decoding part of the FlowNetSimple to provide tight constraints on the predicted flow.

The design flaw of the above approach is obvious as the brightness constraint can easily fail at occluded regions. The unsupervised learning approach taken by Wang *et al.* [70] explicitly models the occluded region by considering forward and backward flow consistency. The underlying model is a modified FlowNetSimple architecture with a warped image I_1' at a particular scale and its photometric error as additional inputs to each decoding stage. Two modified FlowNetSimple are placed side by side. One takes in images in the sequence $\{I_1, I_2\}$ and outputs forward flow $f_{1\rightarrow 2}$ while the other one takes in $\{I_2, I_1\}$ and outputs backward flow $f_{2\rightarrow 1}$. Given $f_{2\rightarrow 1}$, the occluded map is generated by,

$$\begin{aligned} &V(x,y) = \sum_{i=1}^{W}\sum_{j=1}^{H} max(0, 1-|x-(i+{f^x}_{2\rightarrow 1}(i,j))|)\cdot max(0, 1-|y-(j+{f^y}_{2\rightarrow 1}(i,j))|) \\ &O(x,y) = min(1, V(x,y)), \end{aligned} \tag{2.17}$$

where V is the range map [1] computed by backwarping $f_{2\rightarrow 1}$, and O is the occlusion map for I_1. Loss functions for brightness and smoothness are computed in the same manner as Equation 2.16 and both of them are multiplied with the occlusion map O and then divided by its sum for normalization. Beside the photometric loss and the smoothness loss, edge-ware loss [22] is also added,

$$\begin{aligned} L_1 &= \Psi(|\partial f_{1\rightarrow 2}|e^{-\alpha|\partial I_1|}) \\ L_2 &= \Psi(|\partial^2 f_{1\rightarrow 2}|e^{-\alpha|\partial I_1|}), \end{aligned} \tag{2.18}$$

with Ψ being the Charbonnier penalty function.

To learn optical flow without applying a loss function only in the non-occluded regions, Liu *et al.* [44] proposed to guide the flow estimation for the occluded regions using flow estimated from non-occluded regions. The network architecture is similar to PWC-net [62] and multiple consecutive images are given as the input to two network branches with the same architecture. Taking in the image sequence of $[I_{t-1}, I_t, I_{t+1}]$, the first branch warps the feature map F_{t-1} towards F_t using the backward flow $f_{t\rightarrow t-1}$, and the second branch warps the feature map F_{t+1} towards F_t using the forward flow $f_{t\rightarrow t+1}$. Then both warped results are used to compute the cost volumes with F_t and produce forward and backward flow for the next scale. Although the work done by Janai *et al.* [32] also uses multiple frames as the input, and constructs two cost volumes in the same manner, they predict occlusion maps by having an additional occlusion decoder branch. Whereas the Occlusion maps from the work done by Liu *et al.* [44] are produced after flow estimation by using forward-backward consistency check [50],

$$\begin{aligned} f'_{x\to y} &= f_{y\to x}(p + f_{x\to y}(p)) \\ |f_{x\to y} + f'_{x\to y}|^2 &< a_1(|f_{x\to y}|^2 + |f'_{x\to y}|^2) + a_2, \end{aligned} \tag{2.19}$$

where $f'_{x\to y}$ is the reverse of $f_{x\to y}$, and $\{x, y\} \in [t-1, t, t+1]$. To produce reasonable estimation regarding occluded regions, two same models are trained with the first generating occlusion map and optical flow in a self-supervised way and the second model taking in noised inputs where pixels become synthetically occluded with the added noise. The training for the second model uses the optical flow predicted from the first model as supervision and only the second model is used during the testing phase. This approach produced the SOTA result among unsupervised models, additionally, when the trained second model is fine-tuned with the ground truth label in a supervised manner, it also produced the SOTA result among supervised models by the time it was published.

Investigation regarding the important components in unsupervised optical flow estimation models was done by Jonschkowski *et al.* [36]. They stated that convergence and performance can be improved by computing the cost volume between normalized feature maps for PWC-net [62] based unsupervised optical flow estimation models. They found occlusion estimation using a range map as Equation 2.17 performs better than the consistency checking as in Equation 2.19, additionally, stopping the gradient backpropagation at the occlusion mask reduces the divergence.

Unsupervised occlusion estimation in supervised models

Occlusion poses a significant problem regarding optical flow estimation. Occlusion regions refer to those pixels in the first image that do not have matching pixels in the second image as shown in Figure 2.5. This can be due to object motion or the change of depth. Back when the variational method was popular for optical flow estimation,

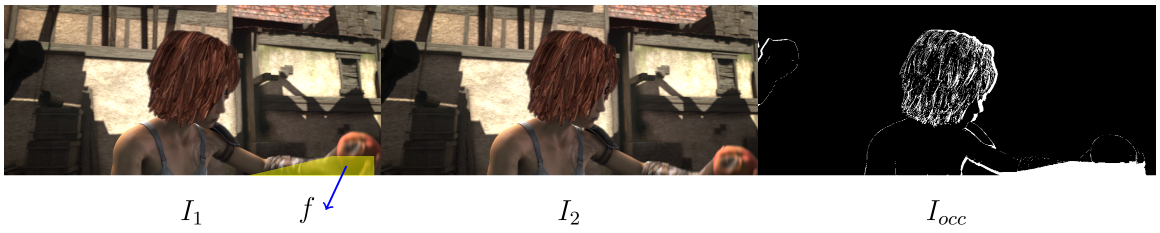

Figure 2.5: Example of occlusion: the occlusion mask is shown in I_{occ} with white pixels being the occluded ones. The yellow region in I_1 becomes occluded as it moves out of the scope of the image in the direction of f which resulted in having no matching pixels in I_2.

occlusion became a major problem for the underlying brightness consistency assumption, therefore the earlier works often model occlusion explicitly as outliers[10]. This approach is also applied when estimating optical flow with self-supervised neural networks, since the loss function for self-supervised models compares the photometric difference between the warped I_2 using the predicted optical flow with I_1 as shown in Equation 2.16.

Regarding supervised optical flow estimation, not many works have estimated occlusion without ground truth masks. The model proposed by Zhao *et al.* [85] learns

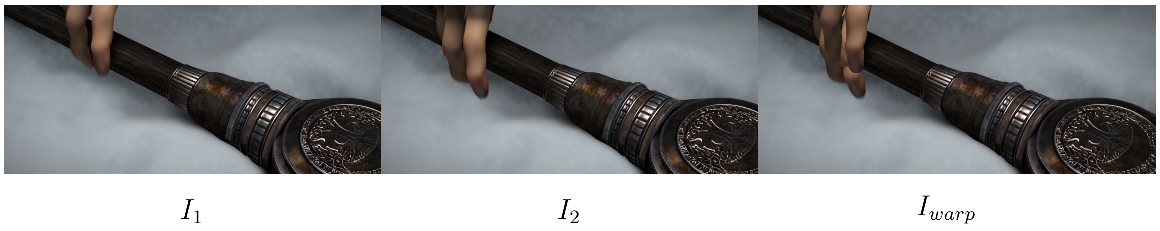

Figure 2.6: The repeated pattern occurs in I_{warp} as I_2 is backwarped to I_1 following the optical flow $f_{1\to 2}$

the occlusion mask without ground truth occlusion labels and the model structure follows the pyramidal structure proposed by PWC-net [62]. Due to the repeated pattern shown in Figure 2.6 when warping is applied, they argue that the cost volume may fail to provide accurate matching information. To remove the redundant information presented in the second feature map after warping, the cost volume at each level is constructed as,

$$c(x_1, x_2) = \sum_{o \in [-k,k] \times [-k,k]} \langle f_1(x_1 + o), D(f_2(x_2 + o)) \cdot \theta + \mu \rangle, \tag{2.20}$$

where D is the deformable convolution operation [16], θ and μ are both learnable parameters, with $\theta \in [0, 1]$ representing occlusion mask and μ providing extra information at the occluded areas.

More recent work done by Jiang *et al.* [33] models occlusion implicitly based on the fact that image regions with similar structures are more likely to have similar motion. Their model design follows that of the RAFT model [66]. Since samples drawn from the cost volume cannot provide matching information for the occluded regions, they propagate motion information within similar regions. Self-attention is applied to the model with features extracted from the first image as both query and key, and using motion features based on the samples as value. Then the propagated motion feature and the original motion feature are concatenated together for the iterative decoding process.

Chapter 3

Method

This section covers the approach we are taken for the optical flow estimation, the overall model architecture is shown in Figure 3.1. The entire model is built upon the RAFT model [66] as well as the GMA model [33]. The first three subsections cover the detailed implementation of the two base models. The last subsection describes two contributions we made, one is the auxiliary correlation aggregation (ACA) unit which uses linear attention model [37] to augment the sampled correlation information, and the other one is the post-refinement process that aggregates local hidden state information based on the local structure coherence.

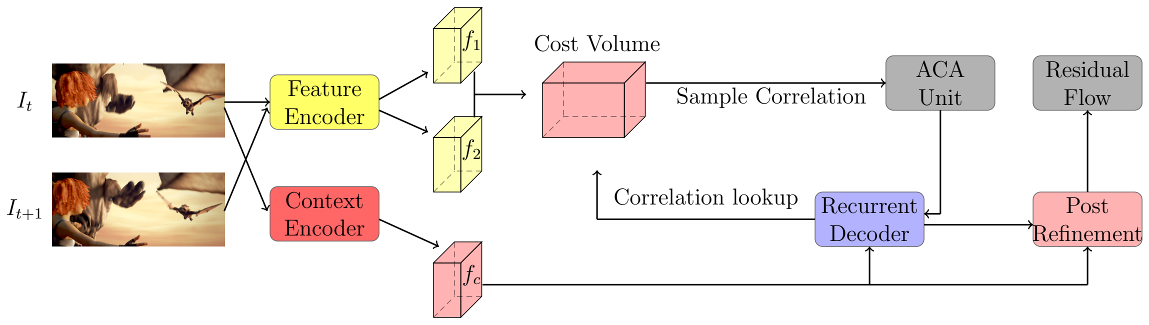

Figure 3.1: **Architecture of the proposed model.** We built our model upon the RAFT [66] model. The proposed model follows the feature extraction, cost volume construction, and the recurrent decoder from the prior work and added the auxiliary correlation aggregation (ACA) unit and post refinement to improve the performance of optical flow estimation. The ACA unit provides more diversified correlation information to the recurrent decoder layer using the linear attention model [37] without causing significant computational overhead and the post-refinement process propagates the hidden state information within a local window to predict a more local coherent residual flow.

3.1 Feature Extraction

Following the seminal work by Teed *et al.* [66], ResNet [23] is used for feature extraction as shown in Figure 3.2. The encoder is composed of three ResNet blocks and

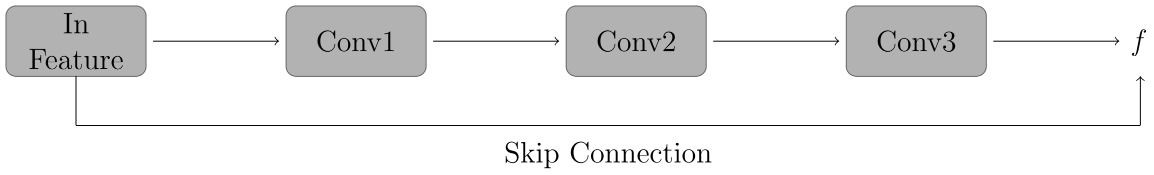

Figure 3.2: **Architecture of the ResNet.** Given the input feature map generated by passing the input image to a single convolutional layer with 7×7 kernel, the ResNet is composed of three convolutional layers. *Conv1* has a kernel size of 1×1, *Conv2* has a kernel size of 3×3, and *Conv3* has a kernel size of 1×1. The output of each convolutional layer is passed to a normalization layer followed by a ReLU activation function. The final output is the summation between the output from the convolutional layers and the input feature, the result of which is followed by a ReLU activation again.

each of them contains three CNN layers interleaved with activation and normalization functions. Two encoders with the same architecture are used, feature encoder E_c produces feature maps for cost volume computation, and context encoder E_r produces feature maps for iterative flow prediction. Given two normalized consecutive images I_1 and I_2, the encoder E_c outputs two feature maps $\{f_1, f_2\} \in R^{H \times W \times C_f}$ for I_1 and I_2 respectively. Both of them are 1/8 of the input resolution with $C_c = 256$. The context encoder E_r takes only I_1 as the input and outputs feature maps $\{f_c, f_h^0\} \in R^{H \times W \times C_c}$ with $C_c = C_f/2$. F_c is the context feature that only needs to be computed at the beginning and used through the iterative flow decoding process. F_h^0 is the initial hidden state for the RNN. Huang *et al.* [27] shows pretrained transformer-based image feature encoders such as Twins [15] produce more accurate optical flow estimation as the extracted features are more comprehensive. Since we have limited GPU memory, ResNet is used as the feature encoder in this work.

3.2 Cost Volume Construction

4D cost volume defines the matching similarity between f_1 and f_2 by computing their dot product $cv = \frac{f_1^T f_2}{\sqrt{C_f}} \in R^{H \times W \times H \times W}$, the denominator was introduced by Vaswani *et al.* [68] to avoid extremely small gradient. The cost volume cv is then reshaped to a 3D tensor $\hat{cv}$ of shape $HW \times H \times W$ for the following sampling process, where a slice at the location i over the first dimension represents the correlation between the ith pixel in f_1 and every pixel in f_2. Since the samples drawn from the cost volume are

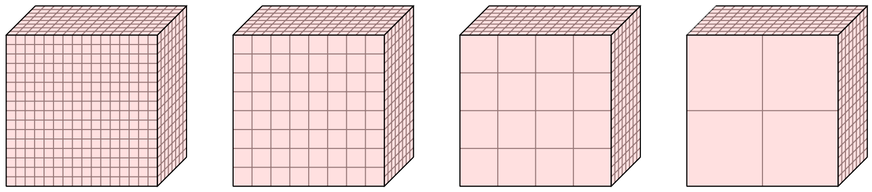

Figure 3.3: **An illustration of the cost volume computation.** From left to right, the spatial dimensions of $\hat{cv}^k$ are in decreasing resolution which is the result of the average pooling operation. Given the cost volume of shape $H \times W \times H \times W$, the cost volume is first reshaped to a size of $HW \times H \times W$, then the pooling operation with a window size of 2×2 and stride 2 is applied over the last two dimensions. Four pooling operations are applied iteratively which resulted in four cost volumes with decreasing scales as shown above.

limited to a relatively small 9×9 local window with radius 4 compared to the size of the feature map, to cooperate with motions that are larger than a range of 32 pixels in the input image, a hierarchical structure is proposed by Sui *et al.* [61]. 2D average pooling operation shifted with 2×2 non-overlapping window is used repetitively over the last two dimensions of the 3D cost volume which resulted in four cost volumes $\hat{cv}^k$ with the decreasing scale of $HW \times H/2^k \times W/2^k, k \in 0, 1, 2, 3$, as an example shown in Figure 3.3. Given the 9×9 sampling window, the correlation tensor used at each decoding iteration has a shape of $H \times W \times 324$, which could potentially capture motions within a range of 256 pixels in the input image. Although the correlation contributed by distant pixels may be diminished by the average pooling operation, compared to the warping operation the above approach is still advantageous.

Warping operation handles long-range motion by continuously warping the image in high resolution with flow predicted from a lower resolution, but it creates structure duplication as shown in Figure 2.6. Since methods that use warping operation only perform cost-volume construction within a small window [30], this doubling effect would create ambiguous matching which could potentially hinder the optical flow decoding process. On the other hand, taking samples from the full cost volume constructed by matrix multiplication essentially moves each pixel in the first image within the second image based on the current cumulative residual flow $f^t_{1\to 2}$ which generates more explicit matching between corresponding pixels.

3.3 Iterative Residual Flow Estimation

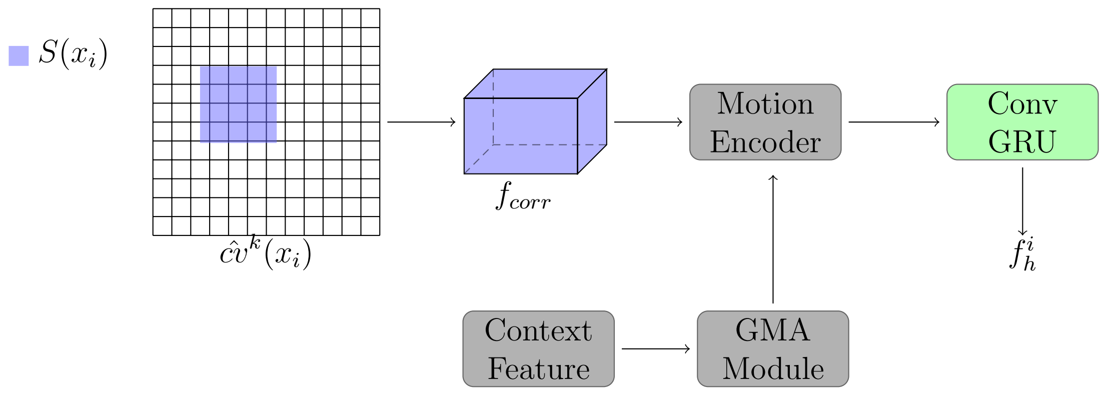

Figure 3.4: **Architecture of the iterative residual flow decoding module.** Given the sampling location obtained by the cumulative residual flow $f^t_{1\to 2}$ at the time step t, we take samples from the cost volume $\hat{cv}^k$ for each pixel x_i in the first feature map. The obtained correlation feature f_{corr} is passed to the motion encoder to produce motion feature f_m which is augmented with GMA module that takes both the context feature f_c as well as f_m as the input. The result of GMA module is then concatenated with f_m and used as the input to the Convolutional GRU model that updates the hidden state which is used to predict the residual flow.

The pipeline for the iterative residual flow decoding module is depicted in Figure 3.4. Given the cumulative residual flow $f^t_{1\to 2}$ estimated at time step t, for pixel x_i located at position i in the first feature map f_1, the correlation sampling location is drawn at $f^t_{1\to 2}(x_i)$ as,

$$\mathcal{S}(x_i) = cv'(x_i^x + f^t_{1\to 2}(x_i)_x + \Delta r, x_i^y + f^t_{1\to 2}(x_i)_y + \Delta r) \quad \text{with } \Delta r \in [-4, 4] \times [-4, 4], \tag{3.1}$$

where $\mathcal{S}(x_i)$ are the samples drawn from the last two dimensions of $\hat{cv}^k$. Over the hierarchical structure of the cost volume mentioned in the last section, the correlation feature f_{corr} is then a tensor of shape $H \times W \times 324$. A sequence of CNNs is then used to extract motion feature f_m from the correlation feature f_{corr} and the cumulative residual flow $f^t_{1\to 2}$. Two layers of CNNs are applied on f_{corr} to encode channel dimension from 324 to 192, $f^t_{1\to 2}$ is also encoded by two layers of CNNs from $H \times W \times 2$ to $H \times W \times 64$. Two encoded features are then concatenated over the channel dimension and reduced to a tensor with a channel dimension of 126 which is then concatenated with $f^t_{1\to 2}$ to form the motion feature f_m.

In this work, the global motion aggregation component proposed by the GMA model [33] is also applied after the motion feature computation. To aggregate motion features among similar image regions, attention is applied between the context feature f_c and the motion feature f_m,

$$f'_m = Attention(f_c, f_c, f_m), \tag{3.2}$$

with f_c being both the query and the key and f_m being the value. The aggregated motion feature f'_m is then concatenated with f_m as the input to the recurrent neural network.

Motion feature f_m is mainly constructed from the samples drawn from the cost volume where the occlusion information is not encoded. Since the occluded pixels do not have any matching pixel in the second image, to estimate the optical flow for those pixels, one way is to infer the motion from those pixels that are similar to those occluded pixels within the first image. This is achieved by the GMA model where the self-attention matrix is computed on the context feature map f_c which encodes the correlation information between all the pixel pairs within the first image. Then the motion feature for each pixel is aggregated by the feature similarity. With this approach, we can potentially recover the motion for those occluded pixels.

The iterative residual flow decoding process is then done with a ConvGRU unit[4] by taking the context feature f_c, the concatenated motion feature f_m^i at the time step i, and the hidden state $f_h^{(i-1)}$ from the previous time step,

$$\begin{aligned} z_i &= \sigma(Conv_z(f_h^{(i-1)}, [f_m^i, f_c])), \\ r_i &= \sigma(Conv_r(f_h^{(i-1)}, [f_m^i, f_c])), \\ q_i &= tanh(Conv_q(r * f_h^{(i-1)}, [f_m^i, f_c])), \\ f_h^i &= (1 - z_i) * f_h^{(i-1)} + z_i * q_i, \end{aligned} \tag{3.3}$$

where σ is the sigmoid activation function. To increase the receptive field without having too many parameters added to the model, ConvGRU is performed by two sets of convolutional kernels. The first set uses a kernel size of 1×5 which encodes the information over the horizontal direction, and the second set uses a kernel size of 5×1 which encodes the information over the vertical direction. All the learnable parameters within the ConvGRU are shared, thus the number of iterations used during the training and testing can be different.

Given the hidden state f_h^i at the time step i, the residual flow $f_{1\to2}^i$ is obtained by passing f_h^i through a two-layer CNNs interleaved with ReLU activation function.

3.4 Contribution

The predicted residual flow at each time step directly depends on the concatenated motion feature f_m, which then depends on the samples from the cost volume $\hat{cv}^k$. In their paper [66], the ablation study shows that although increasing the sampling radius from 2 to 4 reduces the estimation error by 8% when the image sequence presents no motion blur and atmosphere effects, the performance of the model even decreased slightly by 0.3% when the above noise are included as they are the main source of occlusion. Increasing the size of the sampling window could be beneficial when large motions are presented in the input sequence, but it may provide unnecessary matching information regarding pixels that are occluded. Therefore, to provide more comprehensive correlation information during the iterative decoding process without simply increasing the size of the sampling window, we introduce an auxiliary cost volume cv_a alongside the cost volume cv to provide additional matching information during the iterative decoding process.

As the sampling position depends on the cumulative residual flow, which is obtained through the hidden state, we also proposed a pose-refinement process that operates on the hidden state to generate a better sampling position for those occluded pixels.

3.4.1 Auxiliary Cost Aggregation Unit

The auxiliary cost volume cv_a of shape $HW \times k$ is obtained through the last two dimensions of the $\hat{cv}^0$. For each $x_i \in f_1$, a set $\mathbf{S}_{x_i}$ of length k is realized by,

$$S_{x_i} = \max_{A \in cv'(x_i), |A|=k} \sum_{a \in A} a, \tag{3.4}$$

where S_x contains k elements that have the highest correlation value.

Since $k \ll HW$, self-attention can be efficiently applied on cv_a to propagate non-local information.

The attention model [68] propagates information by computing a matrix multiplication between the value V and the attention matrix obtained using query Q and key

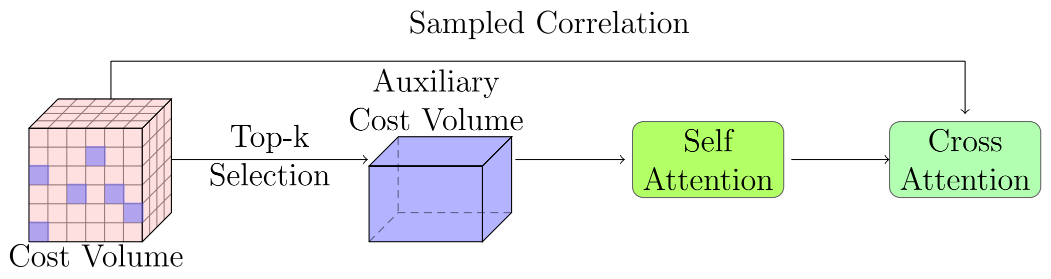

Figure 3.5: **Architecture of the proposed ACA unit.** The ACA unit constructs the auxiliary cost volume by selecting top-k correlations for each pixel in f_1, then self-attention is used to propagate information within the auxiliary cost volume. The sampled correlation from the full cost volume based on the sampling location $f^i_{1\to 2}$ is used as query to compute cross attention with the self-attention result which is used as both key and value. The result of the cross-attention is then concatenated with the sampled correlation for the recurrent flow decoding process.

K with the space and time complexity both being $O(N^2)$, where N is $H \times W$ in our case. Due to the limited GPU memory we have, if we directly apply the self-attention operation on cv_a using the dot product attention, the storage overhead would not allow us to increase the depth of the attention layers which could reduce the amount of information presented in the attention result. Linear attention [37] on the other hand, has reduced space and time complexity of $O(N)$ which is achieved through a low-rank kernel approximation. The formulation for the linear attention operation is defined as,

$$\begin{aligned} \phi(x) &= elu(x) + 1, \\ V' &= \frac{\phi(Q)(\phi(K)^T V)}{\phi(Q)(\sum_{i=1}^{N} \phi(K_i))^T}, \quad \text{with } Q, K, V \in R^{N\times D} \end{aligned} \tag{3.5}$$

where D is the hidden dimension and $\phi(x)$ is a non-negative feature mapping kernel that approximates the exponential kernel used in the dot product attention. By computing the matrix multiplication between $\phi(K)^T \in R^{D\times N}$ and $V \in R^{N\times D}$ first, the number of multiplication and summation operations is then reduced from $O(N^2 D)$ to $O(ND^2)$, as $D \ll N$, the attention layers becomes more time efficient and memory friendly. The denominator is used as a normalization term with a shape of $N \times 1$ that computes the matrix multiplication between $\phi(Q) \in R^{N\times D}$ and $\sum_{i=1}^{N} \phi(K_i) \in R^{1\times D}$.

We embed cv_a to reduce the channel dimension from k to 128 before passing it

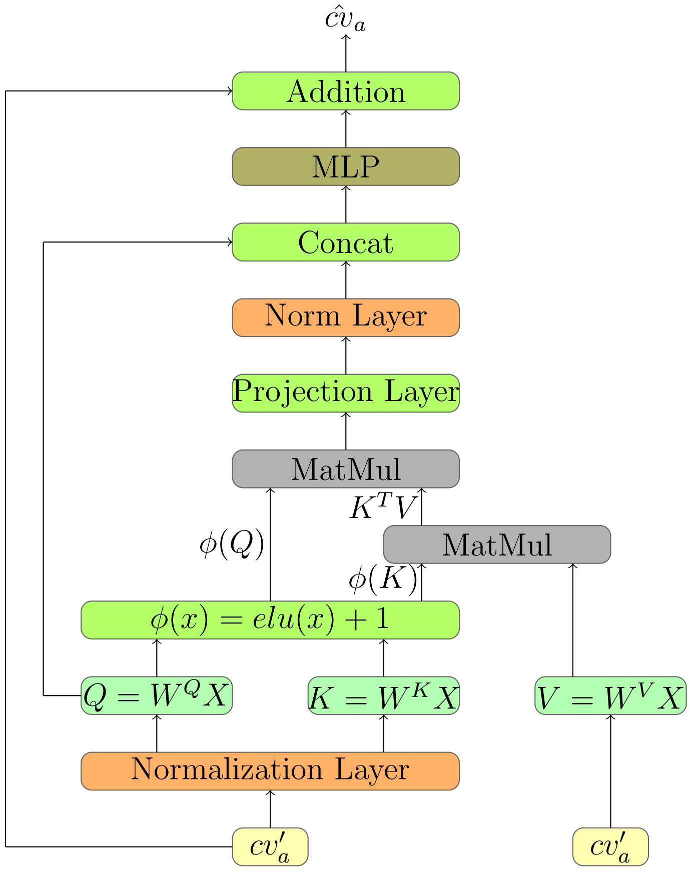

Figure 3.6: **Network structure for self-attention computation.** The linearized self-attention network takes in the embedded top-k elements cv'_a from $\hat{cv}_0$ as query, key and value. Layer normalization is applied to cv'_a before passing it to the query and key projectors, feature mapping is then applied on the resulting Q and K. Matrix multiplication is performed between K and V and then performed again between $\phi(Q)$ and K^TV, the result of which is passed to a linear projection layer. The result of the layer projection is normalized again and the result is concatenated with Q. The concatenation result is then passed to a multiple-layer perceptron (MLP), skip connection is applied between the result of MLP and cv'_a to produce the attention result $\hat{cv}_a$

through two layers of linear attention to compute the self-attention $\hat{cv}_a$. As the sampling position changes at each recurrent iteration, to cooperate with the information embedded in $\hat{cv}_a$ dynamically, we compute cross-attention between the correlation feature f_{corr} and $\hat{cv}_a$,

$$\hat{f}_{corr} = CrossAttn(f_{corr}, \hat{cv}_a, \hat{cv}_a), \tag{3.6}$$

two-layer linear attention is used for cross-attention computation that takes f_{corr} as query, and $\hat{cv}_a$ as both key and value, f_{corr} is then concatenated with $\hat{f}_{corr}$ over the channel dimension for the recurrent flow prediction. Additionally, layer normalization [2] is applied in two places, one before the linear projection and one after the attention computation [73] to stabilize the training, skip connection is also applied where the summation between query and the attention result is used in the following computation. The entire self-attention computation diagram is shown in Figure 3.6

Compared to the sparse cost volume used in the work done by Jiang *et al.* [34], we sample from the full cost volume $\hat{cv}$ and using $\hat{cv}_a$ to provide additional information with can provide more diversified correlation information to the iterative decoding stage. Different than the cost memory encoding proposed by Huang *et al.* [27] where a latent token of shape $D \times K$ is used to encode the patchified $\hat{cv}^0$ to $T_x \in R^{H \times W \times D \times K}$ and then apply self-attention on T_x, the result of which is further used to compute cross-attention between f_{corr}. We directly take top-k samples from $\hat{cv}^0$ to obtain cv_a which is then used for a sequence of attention operations with linear attention to reduce the computation burden.

3.4.2 Post Refinement

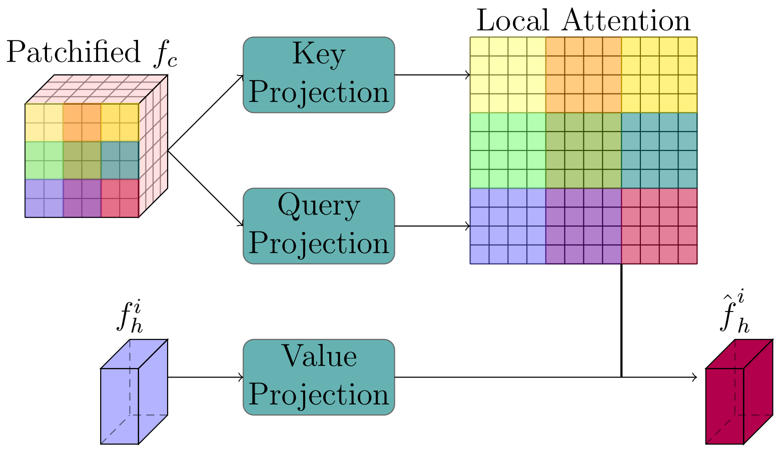

Figure 3.7: **Computation diagram for $\hat{f}_h^i$.** f_c are first broken into non-overlapping patches, then the local attention matrix is computed for each patch, the result of which is then multiplied with the hidden state f_h^i to propagate information within each local patch for generating more local coherent residual flow.

Sampled correlations are a crucial factor for accurate residual flow prediction, regarding the occluded pixel, their sampling locations maybe failed to provide useful correlation information as they do not have any matching pixels. Since sampling locations are determined by the hidden state f_h^i which is directly used to predict residual flow, we proposed a post-refinement model as shown in Figure 3.7 that is directly applied on f_h^i with the aim to produce local coherent optical flow which is formulated as,

$$\begin{aligned} f_c(i,j) &= Linear(f_c(i,j)) \\ f_c'(i,j) &= softmax(f_c(i,j)^T f_c(i,j)) \\ \hat{f}_h^i(i,j) &= f_c'(i,j) \times f_h^i(i,j) \\ f_{1\to 2}^i &= Conv([f_h^i, \hat{f}_h^i]) \end{aligned} \tag{3.7}$$

Specifically, we first break the context feature f_c into 5×5 non-overlapping patches, then the attention matrix within each local patch at location (i,j) is computed by linearly project $f_c(i,j)$, the result of which is followed by a softmax operation over matrix multiplication between the same local patch. The computed hidden state f_h^i is also broken into 5×5 non-overlapping patches and matrix multiplication between the local attention matrix and the local hidden state patch is performed to propagate the information among each 5×5 local window. The residual flow f_i is then computed by passing the concatenated tensor between the hidden state f_h^i and the local augmented hidden state $\hat{f}_h^i$ through a two-layer CNNs.

Chapter 4

Results

This chapter starts by detailing the loss function used to train the model and specifying hyperparameter settings. All the datasets used for training and evaluation are also presented. Evaluations are carried out on two aspects: inter-dataset generalization and intra-dataset generalization. The model's inter-dataset generalizability is obtained by training the model on less challenging datasets first and then testing the model on the training set of datasets that contain more complex scenes. To test the model's intra-dataset generalizability, the experiments are conducted by training and testing the model on datasets with diverse pixel motions. Ablation studies are also performed where all the proposed components from the previous chapter are tested. Visualization of failure cases and the model's limitations are also stated.

4.1 Loss Function

Following the work by Teed *et al.* [66], we use an $l1$ function to supervise the network's training with ground truth optical flow. The loss function is applied to all the predicted residual flow sequences since residual flow at the earlier iterations is less accurate, and a weighting term with exponential decay is applied, and

$$l = \sum_{t=0}^{N} \gamma^{N-t} |f_{1\rightarrow 2}^{t} - f_{gt}|, \tag{4.1}$$

where γ is set to be 0.8.

4.2 Implementation Detail

The model is implemented in PyTorch [55] with mixed precision to reduce memory consumption. We follow the standard training schedule where the model is pre-trained on the FlyingChairs dataset for 240k iterations with a batch size of 4 and then trained on the FlyingThings dataset for 900k iterations with a batch size of 2 while testing the model's generalizability on the Sintel dataset. The model is then fine-tuned with a batch size of 2 on the Sintel dataset as well as the Kitti dataset for 540k and 300k iterations, respectively. All the training and testing are done on a single NVIDIA 3060 GPU. We train the model with one-cycle learning rate policy to achieve faster convergency where the learning rate gradually increases to the maximum learning rate and then decreases to 0. The maximum learning rate for the FlyingChair dataset is set to 2.5×10^{-4} and 1.25×10^{-4} for the rest of the training. The weight decay coefficient for the AdamW optimizer is set to 1×10^{-4} for both FlyingChair dataset and FlyingThings dataset, and 1×10^{-5} is applied for the rest of the training. The number of recurrent iterations is set to 12 during all four training stages. Both sampled top-k elements and sampled correlations are embedded to a dimension of 128 for the attention computation and the number of heads is set to be 4. Since all learnable parameters are shared across the recurrent iterations, following the RAFT model [66], we increase the number of recurrent iterations to 32 during testing for better performance.

4.3 Dataset Descriptions

As optical flow is defined over the 2D space with horizontal motion and vertical motion, to have a better visualization of the pixel's motion, optical flow images below are obtained through color mapping from 2D motion to RBG values.

4.3.1 FlyingChair Dataset

Each image in the synthetic FlyingChair dataset is composed of a background and several foreground objects as shown in Figure 4.1. The background is chosen to be one of three types: city, landscape, and mountain. Foreground chairs are added to the background image. To generate motion, affine transformations are randomly applied to both background and foreground to render the second image for each image pair. Each image has a resolution of 512×384. Among 22,872 image pairs, 640 of them are used for validation and the rest are used for training.

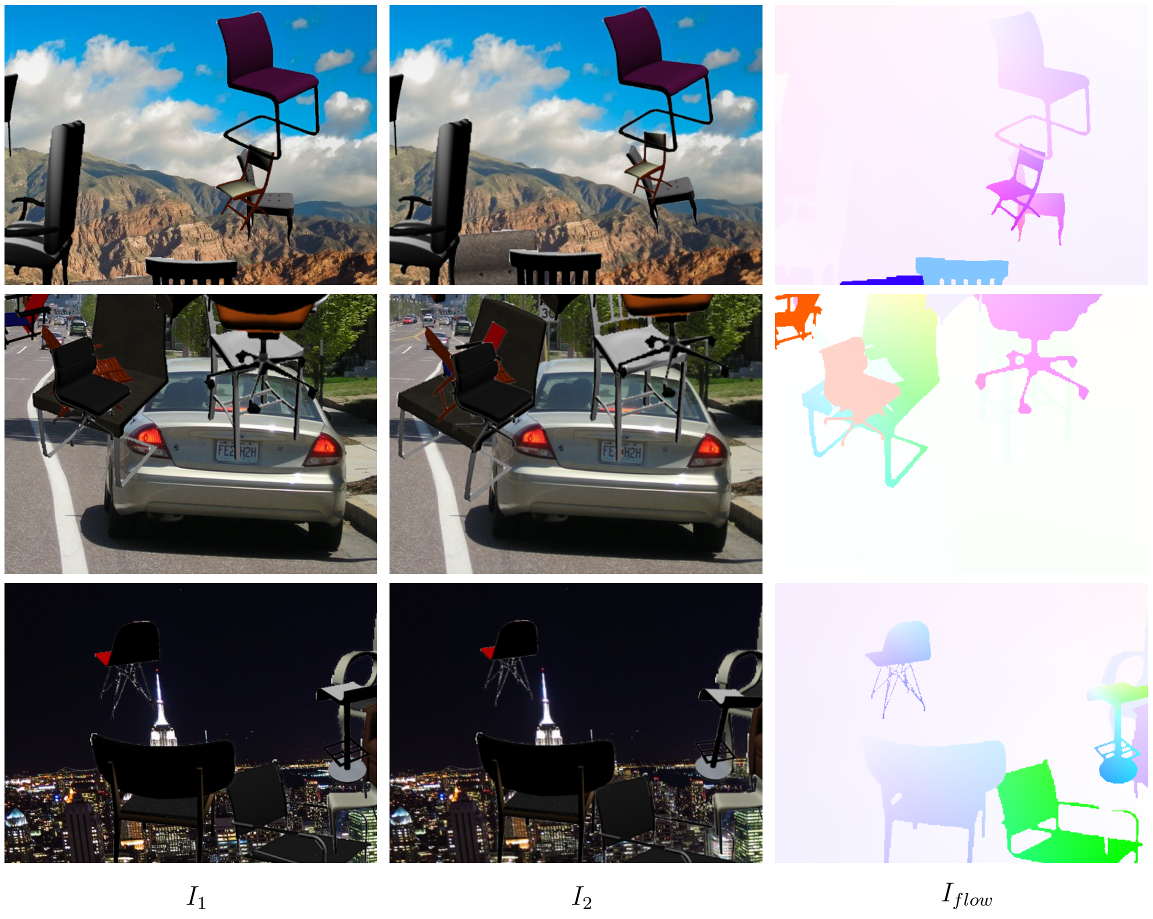

Figure 4.1: **Sample training images from the FlyingChair dataset.** Samples with different backgrounds are listed with their ground truth optical flow.

4.3.2 FlyingThings Dataset

The FlyingThings dataset contains a large set of training data that are generated using random objects as background chosen from cuboids and deformed cylinders while using 3D models from Stanford's ShapeNet with randomly textured material to populate the scene as shown in Figure 4.2. The motions are generated randomly for the background and each of the 3D models in the scene. The dataset has two versions, the **final pass** contains the simulated motion blur, depth-of-field blur, sunlight glare, and gamma curve manipulation, while the **clean pass** contains no image degradations. Both of them are used during the training process. Additionally, both the forward and the backward ground truth optical flow are generated for each pair of images which means a single pair of images can be used twice during the training. The entire

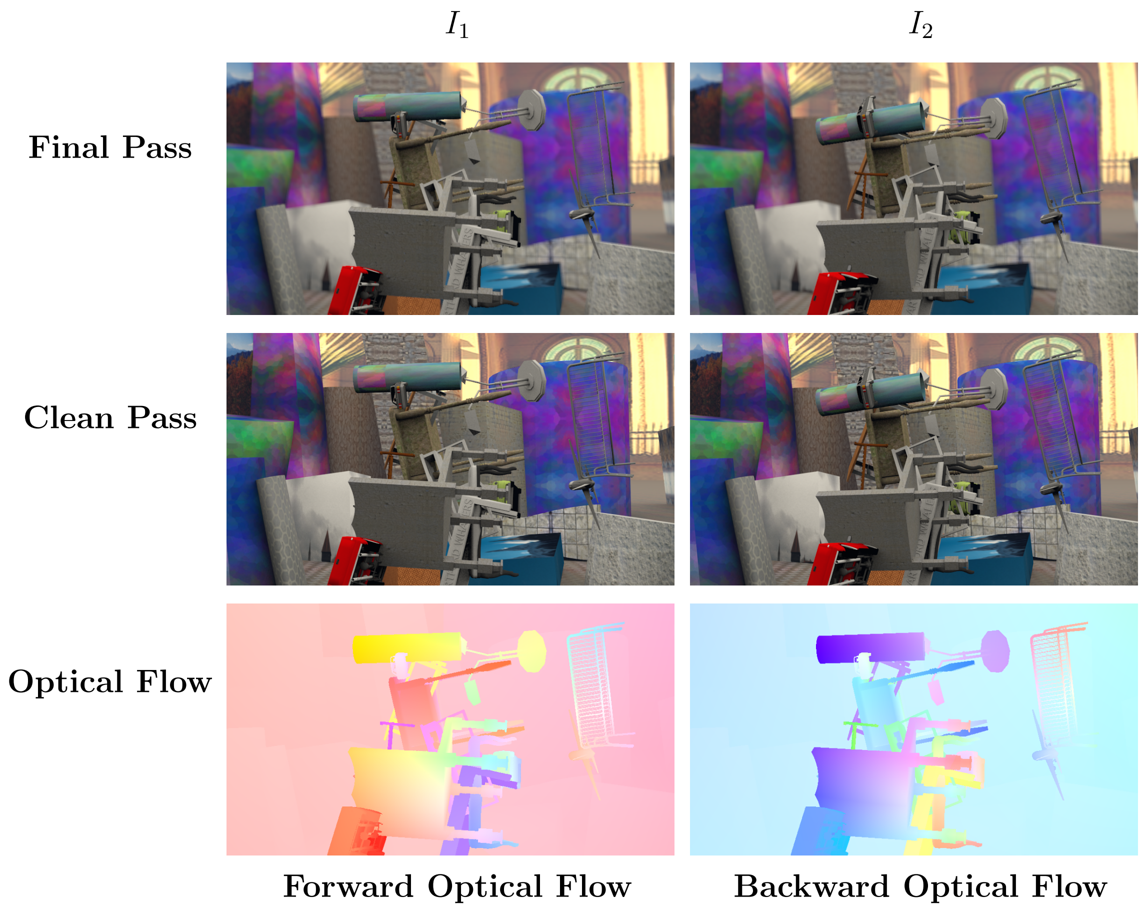

Figure 4.2: **Sample training image pairs from the FlyingThings dataset.** The first row shows the image pair from the final pass where the image degradation is applied. The second row shows the same set of image pair from the clean pass without any image degradation. The third row shows the corresponding optical flow of this image pair, since two ground truth labels exist each image pair can be used twice during the training.

training set for the FlyingThings dataset contains 80,604 image pairs with the ground truth optical flow at a resolution of 540 × 960. Following the standard practice, we only perform the training on this dataset while testing the model's generalizability on the Sintel dataset.

4.3.3 Sintel Dataset

The Sintel dataset is rendered from a real movie with a resolution of 1024 × 436 for each training image. The entire dataset contains 1,628 frames with 1,064 frames used

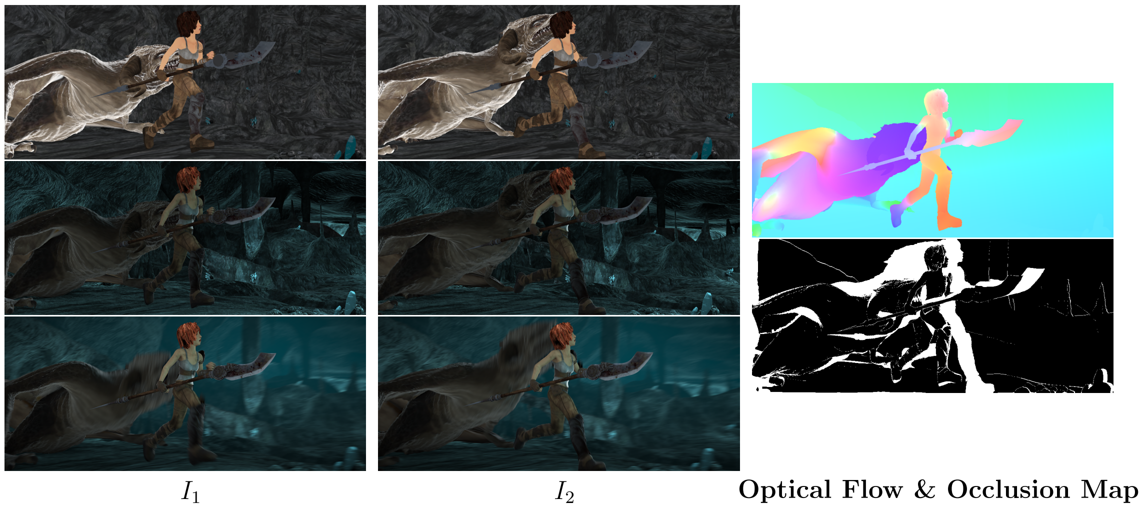

Figure 4.3: **Sample training images from the Sintel dataset.** The first raw of the first two columes are the **Albedo** images, the second raw are **Clean** ones, and the last raw are **Final** images. The last column shows the optical flow between the two images as well as the occluded pixels in the first image I_1.

for training and 564 frames used for testing. The dataset contains three versions. The **Albedo** version contains images with piecewise constant color without any illumination effects, the **Clean** version adds various illumination to the images without introducing any image degradation, and the **Final** version adds atmosphere effects, depth-of-field blur, motion blur, and other noise to the images as shown in Figure 4.3. The dataset also contains large motions with over 100 pixels per frame. Although all the image sequences are available to the public, the testing images are provided with no ground truth optical flow. To determine the model's performance, the predicted optical flows on the testing set are sent for online evaluation.

4.3.4 KITTI Dataset

The KITTI dataset contains 200 training image pairs and 200 testing image pairs with sample images shown in Figure 4.4. The static background is obtained by removing all dynamic objects from the KITTI raw dataset [21], while the moving objects are generated by adding 3D CAD models from Google 3D Warehouse to the scene. As this dataset is relatively small, it is employed as the last fine-tuning step in the training pipeline. Similar to the Sintel dataset, the ground truth labels are only available for the training image sequences while the predicted results on the test dataset are

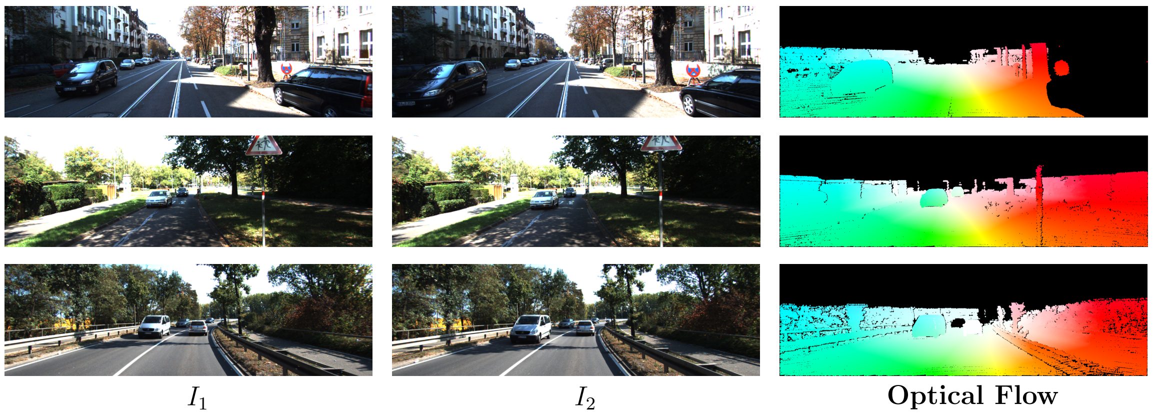

Figure 4.4: **Samples training images from the KITTI dataset.** The first column contains the first image from a training sequence and the second column contains the second training image. The last column contains the provided visualization of the ground truth optical flow labels with black indicating occluded regions. For all sequences, pixels above a certain level are not considered during the evaluation process.

submitted for online evaluation.

4.3.5 HD1K Dataset

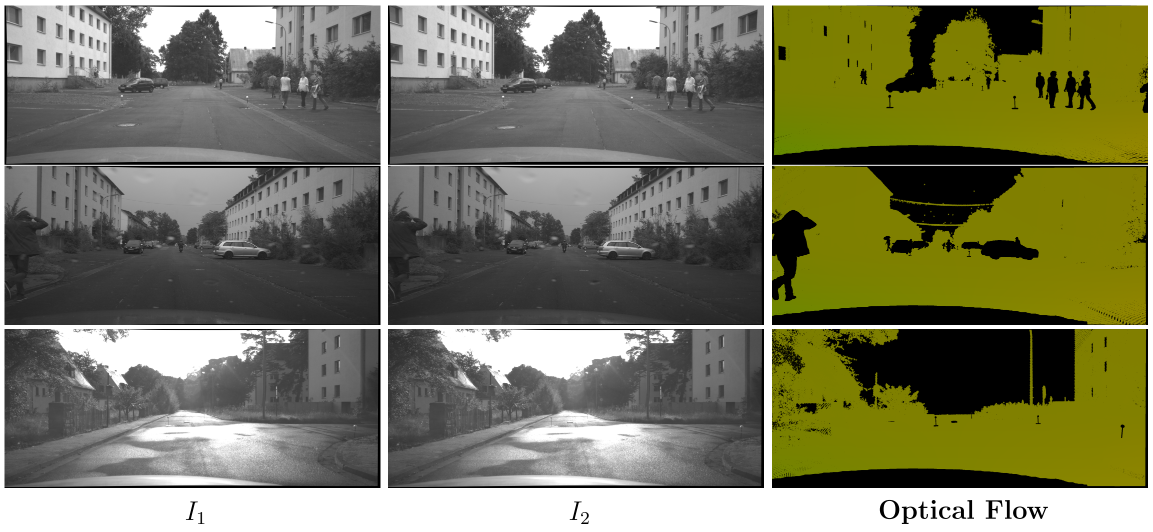

Figure 4.5: **Sample training images from the HD1K dataset.** The first row contain training images under normal weather condition and road condition, the second row shows sample sequences under light rain condition, and the last raw displays training sequence under wet road condition.

Similar to the KITTI dataset, the HD1K dataset also generates training sequences based on urban autonomous driving scenarios as shown in Figure 4.5. Compared to the KITTI, the HD1K dataset contains more diverse weather conditions such as day and night, light rain, as well as wet and dry road conditions. The dataset contains a total of 1,082 image pairs for training with ground truth labels. Following the SOTA models where the benchmark evaluation is only done on the Sintel and KITTI datasets, the HD1K dataset is only included during the training process when the model is fine-tuned for the Sintel dataset.

4.4 Augmentation on Training Inputs

Following RAFT [66], the same augmentation is done across all the training datasets. The training image's brightness, contrast, saturation, and hue are changed based on random selection. Based on the random probability, for a given training image sequence, a bounded region inside the second image will be filled with the average pixel intensity value obtained from the second image to provide occlusion augmentation. Random scaling, random flipping, and random cropping are also applied to the training inputs.

4.5 Evaluation

Evaluation Metrics

Average end-point error (EPE) computes the sum of the squared difference between the predicted flow and the ground truth flow which is calculated for both the Sintel and the KITTI datasets. Additionally, F1-All is only applied for the KITTI dataset which computes the percentage of predictions with EPE greater than 3 pixels and the ratio between EPE and the sum of square of the ground truth is greater than 5%.

4.6 Comparisons with the State-of-the-Art Models

Cross Dataset Generalization. Table 4.1 contains our model's performance on Sintal and KITTI compared to other SOTA models. Our model's generalization performance is obtained by training it on "C+T" and then performing the evaluation

Training Data	Method	Sintel(train)		KITTI(train)		Sintel(test)		KITTI(test)
		Clean	Final	AEPE	F1-all(%)	Clean	Final	F1-all(%)
C+T	PWC-Net[62]	2.55	3.93	10.35	33.7	-	-	-
	AGFlow[47]	1.31	2.69	4.82	17.0	-	-	-
	SCV[34]	1.29	2.95	6.80	19.3	-	-	-
	GMFlowNet[86]	1.14	2.71	4.24	15.4	-	-	-
	GMFlow[75]	1.08	2.48	7.77	23.40	-	-	-
	CRAFT[61]	1.27	2.79	4.88	17.5	-	-	-
	SeparableFlow[84]	1.30	2.59	4.60	15.9	-	-	-
	KPAFlow[48]	1.28	2.68	4.46	15.9	-	-	-
	DIP[87]	1.30	2.82	4.29	**13.7**	-	-	-
	FlowFormer(small)[27]	1.20	2.64	4.57	16.62	-	-	-
	FlowFormer[27]	**1.01**	**2.40**	**4.09**	14.7	-	-	-
	SKFlow[64]	1.22	2.46	4.27	15.5	-	-	-
	RAFT[66]	1.43	2.71	5.04	17.4	-	-	-
	GMA[33]	1.30	2.74	4.69	17.1	-	-	-
	Ours	1.18	2.51	4.58	17.2	-	-	-
C+T+S+K+H	PWC-Net[62]	(1.71)	(2.34)	(1.50)	(5.3)	3.45	4.60	7.72
	AGFlow[47]	(0.65)	(1.07)	(0.58)	(1.2)	1.43	2.47	4.89
	SCV[34]	(0.79)	(1.70)	(0.75)	(2.1)	1.72	3.60	6.17
	GMFlowNet[86]	(0.59)	(0.91)	(0.64)	(1.51)	1.39	2.65	4.79
	GMFlow[75]	-	-	-	-	1.74	2.90	9.32
	CRAFT[61]	(0.60)	(1.06)	(0.58)	(1.34)	1.45	2.42	4.79
	SeparableFlow[84]	(0.69)	(1.10)	(0.69)	(1.60)	1.50	2.67	4.64
	KPAFlow[48]	(0.60)	(1.02)	(0.52)	(1.1)	1.35	2.36	4.60
	DIP[87]	-	-	-	-	1.44	2.83	**4.21**
	FlowFormer[27]	(0.48)	(0.74)	(0.53)	(1.11)	**1.20**	**2.12**	4.68
	SKFlow[64]	(0.52)	(0.81)	(0.51)	(0.94)	1.28	2.23	4.84
	RAFT(warm start)[66]	(0.76)	(1.22)	(0.63)	(1.5)	1.61	2.86	5.10
	GMA[33]	(0.62)	(1.06)	(0.57)	(1.2)	1.39	2.47	5.15
	Ours	(0.51)	(0.90)	(0.56)	(1.24)	1.58	2.57	4.85

Table 4.1: **Quantitative results on Sintel and KITTI 2015 datasets.** "C+T" indicates the models are pre-trained on the FlyingChairs and FlyingThings3D dataset and then cross-data generalization performance is evaluated on Sintel and KITTI where results are shown by average end point error (AEPE) and F1-all(only on KITTI). "C+T+S+K+H" presents models that are fine-tuned on a combination of T, S, K, and H after the "C+T" stage. Results inside parentheses are obtained by evaluating the model on the training set after fine-tuning. Following RAFT, we also use the "warm start" approach during the evaluation. The cross-dataset generalization result indicates that our model surpasses both RAFT and GMA by a large margin achieving performance that is on par with the state of the art approaches.

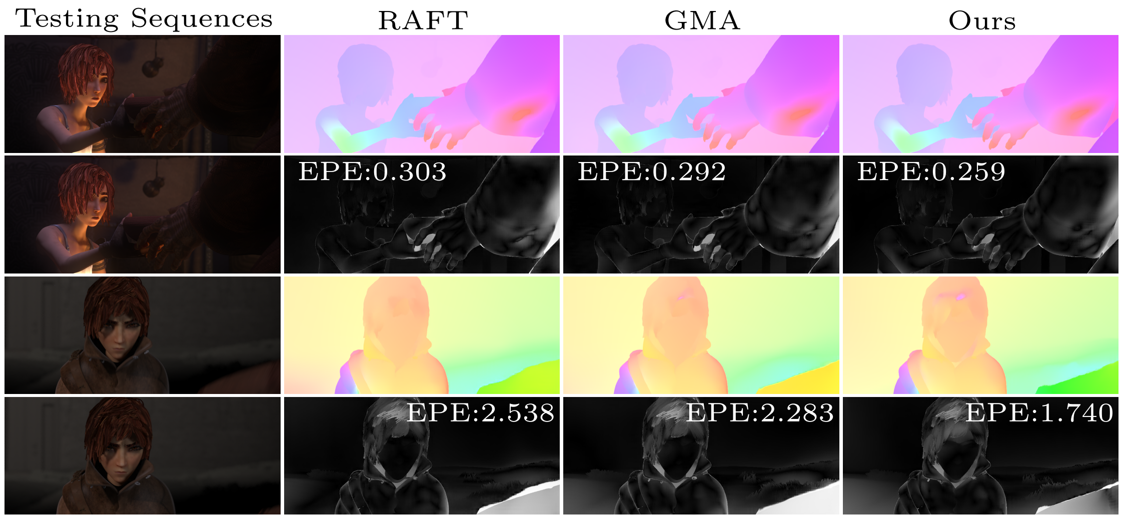

Figure 4.6: **Visualization on results from Sintel benchmark.** Images with only black and white color are the error images obtained from the Sintel benchmark, regions with darker color indicate more accurate optical flow estimation, and regions with lighter color indicate prediction errors. Testing sequence from the top two rows shows that our model is able to capture more fine-detailed motion on small occluded regions which occurs on the bowl that is moving in between the fingers. The bottom two rows indicate that our model is able to capture motions for pixels that completely disappear in the second image which can be seen as the stick on the bottom right corner disappears from the second image.

on Sintel's training data as well as the KITTI's training data, Sintel provides their data in *clean* and *final* version. The *final* version is more challenging compared to the *clean* as it contains noise such as motion blur and atmosphere effects.

The proposed model achieves an AEPE of 1.18 on the clean pass which surpasses SeparableFlow [84], SCV [34], both of which emphasize cost volume manipulation. We improve the estimation performance by 17.5% from 1.43 to 1.18 compared to RAFT [66] and by 9.2% from 1.30 to 1.28 compared to GMA [33] where both of them are widely applied models by state-of-the-art approaches. On the final pass we have a generalization performance of 2.51 which reduces errors by 3% compared to SeparableFlow, 14.9% compared to SCV, 7% compared to RAFT, and 8% compared to GMA. Regarding the generalization test on the KITTI dataset, we obtain AEPE of 4.58 which has a 9% improvement compared to RAFT and F1-all(%) are compatible to both RAFT and GMA.

Our model is inferior to the state-of-the-art model FlowFormer [27] as their model

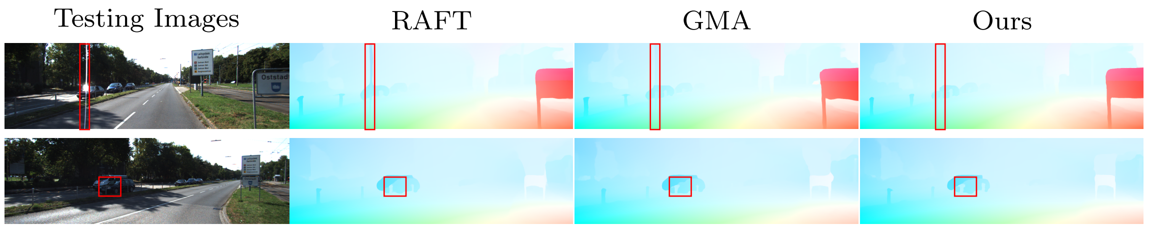

Figure 4.7: **Sample predictions from KITTI dataset.** The first row shows that RAFT and GMA treating the light reflected from the windshield preserves motion. The second row shows that our model can generate a better contour on the motion boundary. Please zoom-in for a more detailed view.

has 18.2M parameters whereas we only have 7.5M parameters. Additionally, they use Twins-Transformer as the encoder model and CNNs are used for feature encoding in our approach. However, when compared to the smaller version of FlowFormer which also uses CNNs for feature encoding, we have similar performance on the clean split and better performance on the final split regarding Sintel. Therefore, we think that given sufficient computational power, our model's performance can be increased further when Transformers are used for feature extraction.

Benchmarks Evaluations. We submitted the generated optical flow on testing images for both Sintel and KITTI-2015 for online evaluation. We outperform the RAFT model on both benchmarks, but the performance is inferior to GMA on Sintel and better than it on KITTI-2015. The Sintel benchmark provides a diverse list of metrics as shown in Table 4.2, the **EPE matched** indicates the endpoint error over the regions that are visible in both frames, the **EPE unmatched** computes the endpoint error over the regions that are visible only in one image, the **d0-10**, **d10-60**, and **d60-140** compute endpoint error for regions closer than 10 pixels, in between 10 to 60 pixels, and in between 60 to 140 pixels from the nearest occlusion boundary, respectively. Endpoint error for regions with movement less than 10 pixels, in between 10 to 40 pixels, and more than 40 pixels are represented by **s0-10**, **s10-40**, and **s40+**, separately.

On the **Final** version of the Sintel benchmark, our model outperforms the RAFT model [66] on all the above metrics, and we also outperform the GMA model [33] on the majority of metrics but our performance is worse than the GMA model regarding **EPE unmatched** and **s40+**. Regarding the **Clean** version of the Sintel benchmark, our model still outperforms the RAFT model, but the result on these metrics are not comparable with the GMA model except for **s0-10** and **s10-40**. Possible reasons for the performance drop are analyzed in Section 4.6.3.

Sintel Dataset	Metric	RAFT	GMA	Ours
Final(test)	EPE matched	1.405	1.241	**1.228**
	EPE unmatched	14.680	**12.501**	13.543
	d0-10	3.112	2.863	**2.718**
	d10-60	1.133	1.057	**0.958**
	d60-140	0.770	0.653	**0.646**
	s0-10	0.634	0.566	**0.488**
	s10-40	1.823	1.817	**1.627**
	s40+	16.371	**13.492**	15.203
Clean(test)	EPE matched	0.623	**0.582**	0.668
	EPE unmatched	9.647	**7.963**	9.035
	d0-10	1.621	**1.537**	1.572
	d10-60	0.518	**0.461**	0.624
	d60-140	0.301	**0.278**	0.313
	s0-10	0.341	0.331	**0.292**
	s10-40	1.036	0.963	**0.880**
	s40+	9.288	**7.662**	9.680

Table 4.2: Quantative comparison between our approach with baseline models under different matrices from the Sintel benchmark.

4.6.1 Occlusion Analysis

After finetuning the model on the clean and final versions of Sintel datasets, following GMA [33], we tested our model's performance against occlusion on the *Albedo* version which contains images with piecewise constant color without any illumination effects. As the brightness constancy is an inherent property in the albedo version, the error caused by occlusion becomes more dominant. Evaluation is carried out by splitting pixels to 'Noc' (non-occluded) and 'Occ' (occluded) based on the provided occlusion map and computing the AEPE on different schemes. Our model shows superior performance compared to RAFT [66] and GMA, results are illustrated in Table 4.4.

Experiment	Method	Sintel(train)		KITTI(train)		Parameters
		Clean	Final	F1-epe	F1-all	
Baseline	GMA	1.27	2.75	4.68	17.21	5.9M
Number of Self and Cross Attention Layers	0-self, 0-cross	1.29	2.67	5.59	18.55	6.0M
	1-self, 1-cross	1.31	2.60	5.00	17.94	6.4M
	2-self, 2-cross	1.28	2.65	4.82	17.85	6.7M
Number of Topk Samples	$k = 512$	1.28	2.64	4.82	17.85	6.7M
	k=1024	1.25	2.61	4.77	17.32	6.8M
Post Refinement	No	1.25	2.61	4.77	17.32	6.8M
	Yes	1.27	2.54	4.79	17.28	7.5M
Number of Training Iterations	300k	1.27	2.54	4.79	17.28	7.5M
	450k	1.21	2.54	4.70	17.54	7.5M
	900k	1.18	2.51	4.58	17.22	7.5M

Table 4.3: Ablation studies, where the configurations applied in the final model are highlighted.

Sintel Dataset	Type	RAFT (AEPE)	GMA (AEPE)	Ours (AEPE)
Clean(train)	Noc	0.32	0.29	**0.24**
	Occ	5.36	4.25	**4.24**
	Occ-in	4.45	3.81	**3.38**
	Occ-out	7.01	5.03	**4.48**
	All	0.74	0.62	**0.53**
Final(train)	Noc	0.66	0.59	**0.50**
	Occ	7.09	**6.22**	6.38
	Occ-in	6.21	5.30	**4.81**
	Occ-out	8.71	7.90	**7.26**
	All	1.19	1.06	**0.93**
Albedo(test)	Noc	0.34	0.32	**0.28**
	Occ	6.35	5.58	**5.50**
	Occ-in	5.83	5.23	**4.57**
	Occ-out	7.29	6.20	**5.45**
	All	0.84	0.76	**0.66**

Table 4.4: Performance on different versions of Sintel datasets with evaluation done using the provided occlusion map.

4.6.2 Ablation Studies

We also performed a sequence of ablation studies on ACVFlow. We first trained on FlyingChairs for 240k iterations and then on FlyingThings3D for 300k iterations while testing the generalization performance on Sintel and KITTI datasets to find the best configuration on the proposed model. The results are shown in Table 4.3. For a fair comparison, we also trained GMA [33] using the setting with reduced batch size and an increased number of training iterations on our machine.

Number of Self and Cross Attention Layers. We experimented with different settings for the number of attention layers used to generate long-range dependencies within the auxiliary cost volume as well as the number of attention layers used to query information from the auxiliary cost volume. When no attention is applied and the embedded auxiliary cost volume is statically concatenated to the sampled correlations at each iteration, the performance on KITTI dataset drops significantly while the performance on Sintel final increases compared to GMA. This finding is in line with other models that leverage transformers [27, 75] where the model's performance benefits from the increasing depth of the attention layers.

Number of Top-k Samples. Our model resulted in a lower error by increasing the number of topk samples with a slight increase in the number of learnable parameters. By increasing the number of correlations encoded in the auxiliary cost volume, information can be shared in a wider image region which benefits the estimation of regions with long-range motion and occlusion.

Post Refinement. On the Sintel dataset, the use of the post-refinement process causes the performance on the clean version to drop negligibly, but it sets a new high on the final version. We argue that due to the amount of noisy information presented in the final version, the piece-wise smoothness encoded in the post-refinement module captures the matching information on motion-blurred regions by sharing hidden state among similar local features, but it may become over-smooth on the clean version. This finding is consistent with the failure case provided in Section 4.6.3.

Number of Training Iterations As the result shows, increasing the number of training iterations on FlyingThing3D gradually reduces the prediction error on cross-dataset generalization and the reduction magnitude decreases as more training iterations are applied.

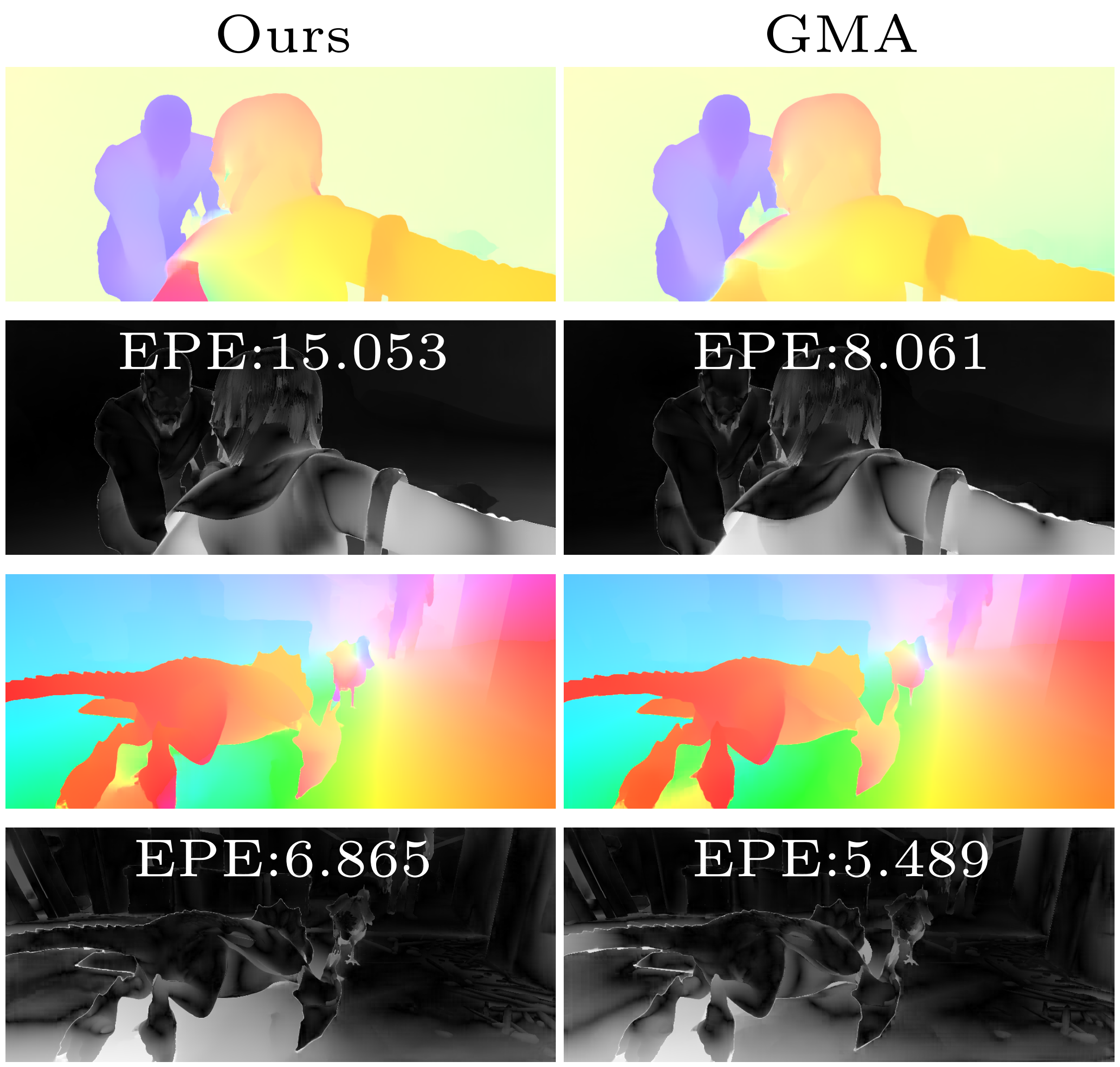

Figure 4.8: **Failure cases on Sintel benchmark.** Failure cases are obtained through the Sintel benchmark, with colorized images being the optical flow estimation while black and white images indicate error mapping. Our model fails to generate accurate optical flow on large textureless regions that also present with occlusion.

4.6.3 Limitations

Despite our model achieving excellent cross-data generalization performance and outperforming both RAFT and GMA on the KITTI benchmark, it can be trapped by textureless regions that are occluded as shown in Figure 4.8. For the first testing sequence, large areas on the right arm are wrongly predicted, for the second testing sequence, our model failed to generate accurate flow prediction for regions under the dragon's body. We suspect that this is caused by the pose-refinement process that over-smoothing the textureless region as the colorized optical flow map shows that color is more uniform on the right arm than the GMA's prediction. We argue that this phenomenon can be mitigated if a more dynamic version of the proposed post-refinement module can be implemented.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

We have proposed an ACA unit that constructs an auxiliary cost volume by simply extracting the top-k matches from the full 4D cost volume computed from the feature dot product. Linear attention is then applied to the auxiliary cost volume to create long-range intra-feature dependencies, compared to the dot product attention, linear attention has lower space and time complexity. The aggregated correlation result between the sampled correlation and auxiliary cost volume is then dynamically integrated into the recurrent residual flow decoding stage. Based on the local piecewise smoothness assumption, a post-refinement module is proposed to aggregate the hidden state based on the local contextural feature similarity. The ablation study performed on the model structure shows that the performance on cross-dataset generalization can be increased by stacking multiple attention layers, having larger top-k selections, including the post-refinement process, and increasing the number of training iterations. The occlusion analysis also shows that our model handles unmatched regions better than the baseline models. Based on the extensive experiments our model shows compatible cross-data generalization performance with SOTA methods while outperforming the baseline models by a large margin.

5.2 Future Work

As transformers have been widely adopted in the field of computer vision, one potential research direction is to change the recurrent residual flow decoder with a transformer-based module. Work by Xu *et al.* [75] have the decoding part of their model implemented using dot product attention but their model's performance is inferior to recurrent decoder based models as shown in Table 4.1. Transformer-based encoders can also be applied to our model and it can potentially boost the performance as shown in FlowFormer [27] where a transformer-based encoder outperforms a CNN-based encoder. One additional research direction is to change the pooling operation used in cost volume construction to an adaptive pooling operation which could be transformer-based.

Bibliography

[1] Luis Alvarez, Rachid Deriche, Théo Papadopoulo, and Javier Sánchez. Symmetrical dense optical flow estimation with occlusions detection. *International Journal of Computer Vision*, 75:371–385, 2007.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Shaojie Bai, Zhengyang Geng, Yash Savani, and J Zico Kolter. Deep equilibrium optical flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 620–630, 2022.

[4] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*, 2015.

[5] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-aware video frame interpolation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3703–3712, 2019.

[6] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[7] James R Bergen, Patrick Anandan, Keith J Hanna, and Rajesh Hingorani. Hierarchical model-based motion estimation. In *Computer Vision—ECCV'92: Second European Conference on Computer Vision Santa Margherita Ligure, Italy, May 19–22, 1992 Proceedings 2*, pages 237–252. Springer, 1992.

[8] Michael J Black and Paul Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer vision and image understanding*, 63(1):75–104, 1996.

[9] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. *Advances in neural information processing systems*, 6, 1993.

[10] Thomas Brox, Christoph Bregler, and Jitendra Malik. Large displacement optical flow. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 41–48. IEEE, 2009.

[11] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *Computer Vision-ECCV 2004: 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part IV 8*, pages 25–36. Springer, 2004.

[12] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International journal of computer vision*, 61:211–231, 2005.

[13] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, October 2012.

[14] Kelvin CK Chan, Shangchen Zhou, Xiangyu Xu, and Chen Change Loy. Basicvsr++: Improving video super-resolution with enhanced propagation and alignment. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5972–5981, 2022.

[15] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. *Advances in Neural Information Processing Systems*, 34:9355–9366, 2021.

[16] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017.

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[18] 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.

[19] Philipp Dufter, Martin Schmitt, and Hinrich Schütze. Position information in transformers: An overview. *Computational Linguistics*, 48(3):733–763, 2022.

[20] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, 27, 2014.

[21] 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.

[22] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 270–279, 2017.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 630–645. Springer, 2016.

[25] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2007.

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

[27] 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.

[28] Tak-Wai Hui and Chen Change Loy. Liteflownet3: Resolving correspondence ambiguity for more accurate optical flow estimation. In *European Conference on Computer Vision*, pages 169–184. Springer, 2020.

[29] James Hutchinson, Christof Koch, Jin Luo, and Carver Mead. Computing motion using analog and binary resistive networks. *Computer*, 21(3):52–63, 1988.

[30] 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.

[31] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015.

[32] Joel Janai, Fatma Guney, Anurag Ranjan, Michael Black, and Andreas Geiger. Unsupervised learning of multi-frame optical flow with occlusions. In *Proceedings of the European conference on computer vision (ECCV)*, pages 690–706, 2018.

[33] 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.

[34] 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.

[35] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[36] Rico Jonschkowski, Austin Stone, Jonathan T Barron, Ariel Gordon, Kurt Konolige, and Anelia Angelova. What matters in unsupervised optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 557–572. Springer, 2020.

[37] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

[38] 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.

[39] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[40] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[41] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326*, 2019.

[42] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Adaptive graph convolutional neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[43] Shaohua Li, Xiuchao Sui, Xiangde Luo, Xinxing Xu, Yong Liu, and Rick Goh. Medical image segmentation using squeeze-and-expansion transformers. *arXiv preprint arXiv:2105.09511*, 2021.

[44] Pengpeng Liu, Michael Lyu, Irwin King, and Jia Xu. Selflow: Self-supervised learning of optical flow. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4571–4580, 2019.

[45] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[46] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[47] Ao Luo, Fan Yang, Kunming Luo, Xin Li, Haoqiang Fan, and Shuaicheng Liu. Learning optical flow with adaptive graph reasoning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 1890–1898, 2022.

[48] Ao Luo, Fan Yang, Kunming Luo, Xin Li, Haoqiang Fan, and Shuaicheng Liu. Learning optical flow with adaptive graph reasoning. *arXiv preprint arXiv:2202.03857*, 2022.

[49] 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.

[50] Simon Meister, Junhwa Hur, and Stefan Roth. Unflow: Unsupervised learning of optical flow with a bidirectional census loss. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[51] Etienne Memin and Patrick Perez. A multigrid approach for hierarchical motion estimation. In *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, pages 933–938. IEEE, 1998.

[52] 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.

[53] 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.

[54] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5437–5446, 2020.

[55] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and

Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[56] Ignacio Rocco, Mircea Cimpoi, Relja Arandjelović, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Neighbourhood consensus networks. *Advances in neural information processing systems*, 31, 2018.

[57] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[58] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[59] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning internal representations by error propagation, 1985.

[60] Mehdi SM Sajjadi, Raviteja Vemulapalli, and Matthew Brown. Frame-recurrent video super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6626–6634, 2018.

[61] 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.

[62] 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.

[63] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8922–8931, 2021.

[64] Shangkun Sun, Yuanqi Chen, Yu Zhu, Guodong Guo, and Ge Li. Skflow: Learning optical flow with super kernels. *Advances in Neural Information Processing Systems*, 35:11313–11326, 2022.

[65] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022.

[66] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 402–419. Springer, 2020.

[67] Michał Tyszkiewicz, Pascal Fua, and Eduard Trulls. Disk: Learning local features with policy gradient. *Advances in Neural Information Processing Systems*, 33:14254–14265, 2020.

[68] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[69] Longguang Wang, Yulan Guo, Li Liu, Zaiping Lin, Xinpu Deng, and Wei An. Deep video super-resolution using hr optical flow estimation. *IEEE Transactions on Image Processing*, 29:4323–4336, 2020.

[70] Yang Wang, Yi Yang, Zhenheng Yang, Liang Zhao, Peng Wang, and Wei Xu. Occlusion aware unsupervised learning of optical flow. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4884–4893, 2018.

[71] Andreas Wedel, Thomas Pock, Christopher Zach, Horst Bischof, and Daniel Cremers. An improved algorithm for tv-l 1 optical flow. In *Statistical and Geometrical Approaches to Visual Motion Analysis: International Dagstuhl Seminar, Dagstuhl Castle, Germany, July 13-18, 2008. Revised Papers*, pages 23–45. Springer, 2009.

[72] Taihong Xiao, Jinwei Yuan, Deqing Sun, Qifei Wang, Xin-Yu Zhang, Kehan Xu, and Ming-Hsuan Yang. Learnable cost volume using the cayley representation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*, pages 483–499. Springer, 2020.

[73] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR, 2020.

[74] 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.

[75] 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.

[76] Jia Xu, René 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.

[77] 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.

[78] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[79] Jason J Yu, Adam W Harley, and Konstantinos G Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*, pages 3–10. Springer, 2016.

[80] Weihao Yuan, Xiaodong Gu, Zuozhuo Dai, Siyu Zhu, and Ping Tan. Neural window fully-connected crfs for monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3916–3925, 2022.

[81] Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Pattern Recognition: 29th DAGM Symposium, Heidelberg, Germany, September 12-14, 2007. Proceedings 29*, pages 214–223. Springer, 2007.

[82] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *2010 IEEE Computer Society Conference on computer vision and pattern recognition*, pages 2528–2535. IEEE, 2010.

[83] Feihu Zhang, Victor Prisacariu, Ruigang Yang, and Philip HS Torr. Ga-net: Guided aggregation net for end-to-end stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 185–194, 2019.

[84] Feihu Zhang, Oliver J Woodford, Victor Adrian Prisacariu, and Philip HS Torr. Separable flow: Learning motion cost volumes for optical flow estimation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10807–10817, 2021.

[85] Shengyu Zhao, Yilun Sheng, Yue Dong, Eric I Chang, Yan Xu, et al. Maskflownet: Asymmetric feature matching with learnable occlusion mask. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6278–6287, 2020.

[86] 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.

[87] 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.

[88] Xueyan Zou, Linjie Yang, Ding Liu, and Yong Jae Lee. Progressive temporal feature alignment network for video inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16448–16457, 2021.