1-18-2023

MULTI-FRAME OPTICAL FLOW ESTIMATION USING SPATIO-TEMPORAL TRANSFORMERS

Fisseha Admasu Ferede

MULTI-FRAME OPTICAL FLOW ESTIMATION USING SPATIO-TEMPORAL TRANSFORMERS

by

Fisseha Admasu Ferede

A thesis

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Major: Electrical and Computer Engineering

The University of Memphis

December 2022

Acknowledgement

Special thanks to the *Herff graduate fellowship* and the department of Electrical and Computer Engineering (EECE) at the University of Memphis for providing me stipend and tuition supports during my graduate studies. I thank my academic advisor, Dr. Madhusudhanan Balasubramanian, for his continuous support and encouragement through out this research work. I thank my thesis committee members Dr. Eddie Jacobs from the EECE department and Dr. Deepak Venugopoal from the Computer Science department for their advice and feedback for this thesis work.

I also would like to express my deepest gratitude to my family (Admasu, Nunuye, Teddy and Hilina) for their unbounded love and believing in me through out my life. Special thanks to my dad (Admasu Ferede) who has always been a motivation for me to pursue science.

Abstract

Optical flow estimation is a computer vision problem which aims to estimate apparent 2D motion (flow velocities) of image intensities between two or more consecutive frames in an image sequence. Optical flow information is useful for quantifying dense motion field in numerous applications such as autonomous driving, object tracking in traffic control systems, video frame interpolation, video compression and structural biomarker development for medical diagnosis. Recent state of the art learning methods for optical flow estimation are *two-frame based methods* where optical flow is estimated sequentially for each image pairs in an image sequence. In this work, we introduce a learning based *spatio-temporal transformers* for multi-frame optical flow estimation (SSTMs). SSTM is a multi-frame based optical flow estimation algorithm which can learn and estimate non-linear motion dynamics in a scene from multiple sequential images of the scene. When compared to two-frame methods, SSTM can provide improved optical flow estimates in regions with object occlusions and near boundaries where objects may enter or leave the scene (out-of-boundary regions). Our method utilizes 3D *Convolutional Gated Recurrent* Networks (3D-ConvGRUs) and space-time attention modules to learn the recurrent space-time dynamics of input scenes and provide a generalized optical flow estimation. When trained using the same training datasets, our method outperforms both the existing multi-frame based optical flow estimation algorithms and the recent state of the art two-frame methods on Sintel benchmark dataset (based on a computer-animated movie) and KITTI 2015 driving benchmark datasets.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In computer vision, optical flow estimation is a task of estimating dense and apparent flow field in a scene from a sequence of input images. Given two or more input images, the goal of optical flow estimation algorithms is to estimate the pixel-wise velocities between the input images with the flow estimates invariant to occlusions, motion blur, out of boundary regions and small/large displacements.

Horn and Schunck [3] was a pioneering algorithm that introduced a variational energy minimization approach to estimate optical flow using the brightness consistency assumption (pixel intensities are assumed to be constant for small temporal change) and smoothness constraint (smooth or elastically deformed motion is assumed) as part of the objective function. More recent classical methods used a bank of hand-crafted motion filters tuned to capture moving patterns and texture characteristics from input images to estimate the direction and magnitude of optical flow fields [4, 5].

With the recent advancement of deep learning frameworks, and availability of large datasets with known ground truth, optical flow estimation task is reformulated as an end-to-end learning problem without requiring assumptions about the nature of the input images or the motion pattern. FlowNet [6] introduced a two-frame learning based optical flow estimation method, where a single flow field is estimated from a pair of input frames, based on convolutional neural networks (CNNs). Following FlowNet [7] various improvements were done on two-frame learning based techniques. These techniques include, coarse-to-fine pyramid network

[8], multiple intermediate flow estimates and warping based brightness error computation [7], receptive field guided motion feature extraction networks [9] and 4D all-pairs correlation volume with gated recurrent networks [1]. These two-frame methods can be seen as approaches that approximate flow dynamics between two corresponding pixels in an image pair linearly (with some recurrent information considered in some methods).

Another approach of solving the optical flow estimation task is by using multiple input frames simultaneously. Such multi-frame based optical flow estimation methods use three or more consecutive input images of the same scene to estimate one or more optical flows at each inference instants. Compared to two-frame methods, multi-frame methods can benefit significantly from the temporal information that are available in longer input sequences and thus such an approach may be able to capture non-linear flow dynamics among corresponding pixels in an image sequence. Motivated by Heeger's [4] approach to use hand crafted spatio-temporal Gabor filters for optical flow estimation, Teney *et al.* [10] developed a learning-based multi-frame optical flow estimation based on learnable 3D CNNs and signal processing concepts. Zhile *et al.* [11] extended the two-frame PWC-Net [8] method to a multi-frame method via fusion of multiple flow estimates in parallel and was able to achieve a better result compared to the original PWC-Net. As of now, there are not many state of the art multi-frame based optical flow estimation methods. We believe that this is mainly due to the limited availability of datasets for training multi-frame optical flow models.

Poor generalizations (low estimation accuracy) around occluded and out of boundary regions is one of the significant limitations of two-frame based optical flow estimation methods. This is likely because in these regions the flow dynamics are often complex and cannot be fully captured using only two frames. In other words, for regions of the input frames which can only be seen in one of the two frames, two-frame methods do not have a good context about these disappearing regions. This limited information about the disappearing regions causes erroneous flow estimates. Moreover, we believe that, the estimated flow in such methods is a linear approximation, such that given two input frames we estimate the pixel-wise trajectory that fits a pair of corresponding points in the two given frames. While this might give a good flow estimate in most cases, it cannot generalize the non-linear flow dynamics of

the motion over multiple time windows. For example, if we want to use the estimated two-frame based flow vectors to interpolate more frames in between, we are assuming a linear trajectory that do not hold true to generalize real world trajectories.

In this work, we develop a multi-frame optical flow estimation method with the goal of overcoming the limitations of two-frame methods near occluded and out of boundary regions. Our methods are designed on the basis of understanding the space-time nature of the input sequence in a wider temporal windows by taking multiple input frames, instead of two, and estimate their flows in parallel. Unlike most two-frame methods whose context features are limited to the spatial context of a single image, we design spatio-temporal filters and apply them over multiple input images to capture the space-time context of the input sequence. Moreover, we learn the recurrence structure of the input in space-time domain to understand the non-linear dynamics of optical flow. Our current method is based on three input frames and it has a quadratic nature as the estimated flows are over three corresponding pixels of three input frames. This gives our method the ability to understand the non-linear flow dynamics when compared to two-frame methods. We believe that, by understanding the non-linear flow dynamics we can predict the trajectory of each scene points with a better accuracy and can also be used for interpolating and extrapolating more realistic frames (in next-frame prediction problems).

Our proposed multi-frame method has the following features. Given three input images, we use 4D correlation volume to compute visual similarities between each of the neighbouring input frames. Concurrently, we extract context features from three consecutive frames in each image sequence using spatio-temporal 3D CNN network. With the weights of 3D Convolutional Gated Recurrent Units (3D conv GRUs) tied, we learn the non-linear recurrence of the input sequence given the context features, the correlation volumes, and intermediate optical flow estimates in each GRU level. Further, we introduce an error block network which computes a multi-level brightness error by warping intermediate flow estimates. This error was used as an additional information while generating motion features. Moreover, we used space-time attention mechanism with a cross-attention architecture wherein two attention networks were used to capture global dependencies among corresponding pixels in three

consecutive image frames.

The structure of this thesis is organized in the following order. In *chapter 2*, we describe the i) multi-frame optical flow datasets we used for training, validation and testing, ii) details of the network input and output, iii) evaluation metrics used during validation and testing, and iv) tools used for visualizing estimated flows. In *chapter 3*, we provide a brief review of works related to the key features of our methods; define a mathematical framework for multi-frame optical flow estimation; and discuss the main features and contribution of our multi-frame methods. In *Chapter 4*, we present our experiment details including training approaches, ablation experiments conducted to elucidate key features of our methods when compared to other methods; and present qualitative and quantitative results of our method. In *Chapter 5*, we summarize and discuss our results, scientific observations from each of our experiments, and our contribution. In *chapter 6*, we conclude this research work with a brief summary of our multi-frame optical flow estimation methods, their performance, and future research direction of this research work.

Chapter 2

Datasets and Evaluation Methods

2.1 Datasets

In this section, we provide a brief overview of the datasets that were used for training, validating and testing all the optical flow algorithms studied in this research work. It should be noted that the ground truth of the final testing datasets were not available to us. Further, the accuracy of our methods on the final testing datasets were assessed blindly by the *leaderboards* of respective datasets.

2.1.1 MPI Sintel

MPI Sintel [12] is naturalistic synthetic dataset for optical flow estimation based on the CGI movie *Sintel*. MPI Sintel contains 1064 training images from 35 different scenes along with their ground truth optical flows and 564 testing images from 8 different scenes with no ground truths. Compared to other optical flow benchmark datasets such as Middleburry [13] and KITTI [14], Sintel contains frames with larger and non-rigid motions [15]. Sintel has two versions namely, Clean pass and Final pass. The Clean pass version contains computer rendered images with no image degradation effects. The Final pass dataset, on the other hand, contains frames with different degradation effects such as defocus blur, motion blur and artificial atmospheric effects [15].

2.1.2 KITTI 2015

KITTI 2015 [16] is a real world dataset for optical flow estimation and autonomous driving recorded from a moving vehicle. Compared to the previous KITTI 2012 version, KITTI 2015 contains more dynamic scenes. The moving objects in each scene are mostly rigid containing large displacement motions.

The multi-view version of KITTI 2015 that we used in this research work has 200 different scenes (sequences) and 20 training frames per scene. Only the ground truth flow between 10^{th} and 11^{th} frame in each sequence was available for training. The ground truth flows between each pair of images in every scene are sparse such that motion information of some objects (distant object like sky) were not available [6]. The multi-view version also has 200 testing scenes which we used to test our final methods. The "multi-view" terminology used by the KITTI dataset refer to "multi-frame".

2.1.3 Virtual KITTI2

Virtual KITTI2 [17] dataset for optical flow estimation is a synthetic dataset recreated based on the KITTI dataset. The dataset consists of 5 sequences which are clones of the KITTI tracking dataset. Each sequence has modified variants of the cloned sequences with different camera orientations, lighting and weather conditions. Different camera orientations include $15°$ and $30°$ rotations to the left and right. The dataset also includes five different image sequences with artificially modified weather and lighting conditions such as with rain, morning, overcast, fog and sunset.

Virtual KITTI2 is an updated version of Virtual KITTI [18] containing a second stereo camera view for each sequence. Virtual KITTI2 has two stereo camera views namely Camera0 and Camera1 such that Camera1 is positioned approximately 0.53m to the right of Camera0 coordinates [17]. This doubles the size of the dataset in Virtual KITTI2 compared to Virtual KITTI [18] which has only one camera view (Camera0). Moreover, Virtual KITTI2 utilizes advanced post-processing and lighting features available in the more recent version of the Unity game engine available for generating a more photo-realistic images [18] when

compared to Virtual KITTI [17].

2.1.4 Monkaa

Monkaa [19] is a synthetic dataset based on the short movie *Monkaa* created using a customized version of the open source 3D creation suite Blender. This dataset contains 8591 training frames from 8 different scenes with resolution size of 960 $\times$ 540. Similar to the MPI Sintel dataset, Monkaa contains two different versions of the input frames namely final pass and clean pass. The final pass version contains different effects such as motion blur, lighting and depth of field blur effects that degrade the colors and textures of the input frames [19]. The clean pass on the other hand contains clean frames with no added degrading effects. Furthermore, for data augmentation purpose, more sequences are rendered by varying camera orientations and motion paths. While both Monkaa and MPI Sintel were based on 3D animated movies, Monkaa dataset is less naturalistic and has fewer number of scenes and randomness when compared to the MPI Sintel dataset.

2.1.5 HD1K

HD1K [20] is a real world optical flow and autonomous driving benchmark dataset captured from a moving vehicle. The dataset consists of various traffic scenes taken relatively at a larger resolution of 2560×1080 pixels.

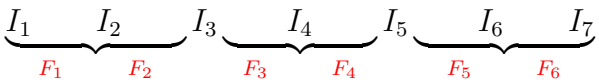
Table 2.1: Summary of training datasets used. For KITTI, we used the multi-view version which consists of 20 frames per scene or sequence (out of the 20 frames available in every scene, we used only three frames namely 9^{th}, 10^{th} and 11^{th} frames).

Datasets	Number of frames	Number of scenes	Number of Ground truths	Resolution (pixels)	Dataset type
Sintel	1064	23	1041	1024×436	Synthetic
KITTI	600	200	200	1242×375	Real
Virtual KITTI2	21,260	5	42,510	1242×375	Synthetic
Monkaa	8591	8	17,166	960×540	Synthetic
HD1K	1083	35	1083	2560×1080	Real

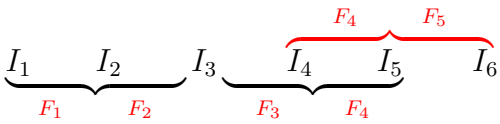
2.2 Input Frame/Output Flow Scheme

The aforementioned datasets were kept in their original structure and format. Each of the scenes of these datasets should contain at least three input frames. For every three consecutive input images of the same scene, I_1, I_2 and I_3, our method estimates two optical flow fields, F_1 and F_2, respectively between I_1 and I_2, and I_2 and I_3. For estimating optical flow for all frames in a sequences, three input frames were chosen as input to our multi-frame method. Depending on the number of frames available per sequence, the following strategies were used for selecting the three input frames for our methods.

1. For odd number of input image frames per sequence:



2. For even number of input image frames per sequence:



2.3 Performance Evaluation Methods

The following standard optical flow evaluation metrics were used for validating our models during training as well as for assessing the accuracy of our methods on the final test datasets reported by the KITTI and Sintel leaderboard.

2.3.1 Endpoint Error (EPE)

Endpoint error (EPE) is a standard evaluation metrics for optical flow estimation. EPE measures the point-wise squared error summed over all pixel positions between the estimated optical flow and its corresponding ground truth flow. Given the estimated flow vectors $\boldsymbol{u}_{est}(x, y)$

and $\boldsymbol{v}_{est}(x, y)$ and the corresponding ground truth vectors $\boldsymbol{u}_{gt}(x, y)$ and $\boldsymbol{v}_{gt}(x, y)$ at pixel location (x, y), EPE at pixel position (x, y) in the image plane is computed as:

$$EPE(x, y) = \sqrt{(\boldsymbol{u}_{gt}(x, y) - \boldsymbol{u}_{est}(x, y))^2 + (\boldsymbol{v}_{gt}(x, y) - \boldsymbol{v}_{est}(x, y))^2} \tag{2.1}$$

The EPE of an estimated flow of size $h \times w$ is computed as the sum of $EPE(x, y)$ over all pixel locations, (x, y). This is given as:

$$EPE = \sum_{x=0}^{w-1} \sum_{y=0}^{h-1} EPE(x, y) \tag{2.2}$$

2.3.2 EPE Near Occluded Regions (d_{m-n})

This evaluation metric measures the accuracy or EPE within a certain rage of distance from the occluded regions. d_{m-n} denotes EPE over regions with pixels that are m to n pixels away from the nearest occluded boundary. This metric helps us to understand how well our estimated flow result or in general our flow estimation method is performing near occluded regions. d_{m-n} is a standard measurement used in the MPI Sintel [15] leaderboard.

2.3.3 s_{m-n}

s_{m-n} is defined as EPE over regions with flow magnitude in the range of m to n pixels per frame. This helps us evaluate and understand the performance of our optical flow estimation methods in regions with various flow magnitudes..

2.3.4 Fl-all %

Fl-all (%) measures the percentage of outlier optical flow estimates whose EPE is above **3** pixels or **5**% of the ground truth averaged over all regions. In other word, each flow estimate at pixel location (x, y) is considered to be correct estimate if the $EPE(x, y)$ is less than **3** pixels or **5**%. Fl-all is a standard optical flow evaluation metric used by KITTI leaderboard.

2.3.5 Fl-fg % and Fl-bg %

Both Fl-fg% and F-bg% measure EPE outliers similar to Fl-all with the only difference being the outliers are measured over a specific region of interest (instead of the entire region). Fl-fg% and Fl-bg% measure EPE outliers whose EPE are above **3** pixels or **5**% of the ground truth in the foreground and background regions respectively. Fl-fg% and F-bg% are also the metrics used by KITTI leaderboard on the KITTI test dataset.

The KITTI leaderboard reports the aforementioned three optical flow outlier metrics (Fl-fg%, Fl-all% and F-bg%) in two groups namely for non-occluded regions only and for all regions for a total of 6 evaluation metrics.

2.4 Flow Color Coding

In this work, we use a standard optical flow color coding map to visualize the estimated optical flows and have qualitative comparisons with estimated flow results of other methods. The flow color coding scheme captures the magnitude and directional characteristics of flow vectors using a color wheel. Figure 2.1 shows the optical flow vector field as a quiver plot, color coded flow field and the flow color coding wheel. In the color coding wheel, the magnitude and direction of flow vectors are encoded as the saturation and color information respectively.

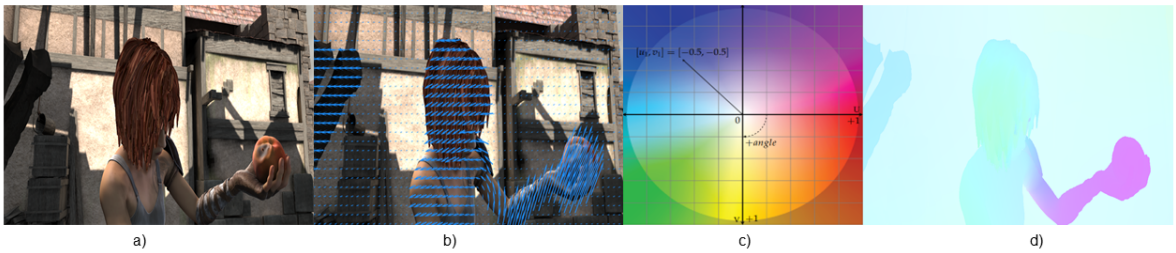


Figure 2.1: Flow visualization. a) Input frame, b) quiver plot of the optical flow field, c) Flow color coding map, d) Color coded optical flow field.

Chapter 3

Methodology

3.1 Problem Definition

Given multiple sequences of image frames, $I_t, I_{t+1}, I_{t+2}, ..., I_{t+n}$ for integer $n \geqslant 2$ such that an input frame, $I_t = E(x(t), y(t), t)$, at time t is defined as the brightness of point $(x(t), y(t))$, our goal is to estimate the intermediate flow velocities, $F_{t\rightarrow t+1}, F_{t+1\rightarrow t+2}, ..., F_{t+(n-1)\rightarrow t+n}$, between every two consecutive frames such that $F_{t\rightarrow t+1} = G(u(t), v(t), t)$ is the optical flow field between frame I_t and I_{t+1}. We call this multi-frame optical flow estimation as the flow estimation at each instance depends on not only a pair of images but three or more inputs images. Three-frame based optical flow estimation is the case where $n = 2$, such that given three consecutive frames I_t, I_{t+1}, I_{t+2} our goal is to estimate $F_{t\rightarrow t+1}, F_{t+1\rightarrow t+2}$.

Figure 3.1 shows the spatio-temporal orientation of an object moving in the x direction at a constant velocity (Figure 3.1 a). Figure 3.1 b) shows the space-time view of the motion trajectory over a time frame of length t such that a slice along xy- plane represents the spatial position of the object at time t. This slice is equivalent to a frame that captures the spatial orientation of the moving objects within the scene at that particular time instance. Figure 3.1 c) shows the slice of the space-time cubic orientation along xt-plane in which the slope of the bar's motion in the tx-plane captures the horizontal velocity of the bar's motion.

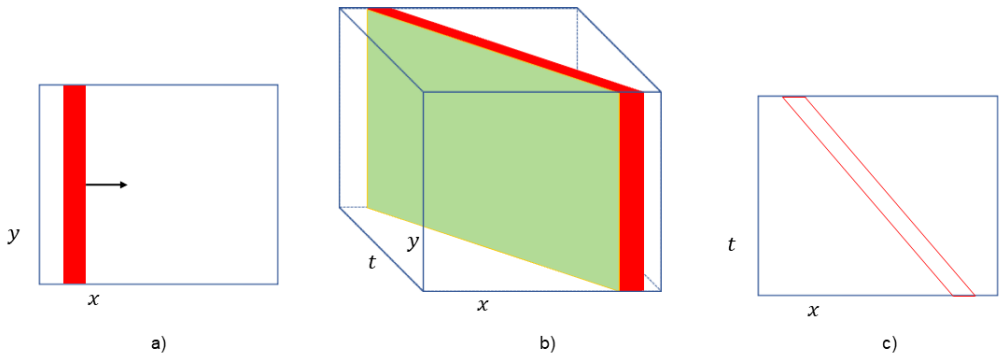


Figure 3.1: Multi-frame motion trajectory (Image redrawn from [2]). a) the red bar is moving with constant velocity along the x direction. b) illustrates the motion over spatio-temporal domain. c) shows the slice of (b) along $t - x$ plane such that the inverse of the slop of this bar represents the horizontal velocity.

3.2 Related Work

Heeger *et al.* [4] approached the optical flow estimation problem as the analysis of the space-time visual motion of a given scene. Heeger's multi-frame based method jointly captures 2D spatial motion dynamics and temporal dependencies over multiple frames. These space-time motion features and representations are captured by families of 3D Gabor filters tuned to respond to specific flow magnitudes and directions. By learning the space-time motion dynamics, Heeger's optical flow estimates were able to deal with the *aperture problem*.

Toshio *et al.* [21] developed one of the earliest classical methods for multi-frame optical flow estimation. In this work, Toshio *et al.* extended the two-frame Horn and Schunck [3] optical flow algorithm by introducing temporal coherence constraint to the existing smoothness constraint using Kalman filter. The smoothness constraint assumes the moving objects in the scene to be rigid or deforming elastically. The temporal coherence constraint introduced by Toshio *et al.*, on the other hand, fuses the smoothness constraint over multiple time frames to have a better generalization of the scene dynamics. Compared to the original Horn and Schunck two-frame algorithm, the resulting optical flow estimates from Toshio's multi-frame

method are more accurate and robust to noise.

A classical machine learning based method introduced by Kennedy *et al.* [22] showed the significance of using multiple frames for optical flow estimation. Kennedy computed multiple flow estimates based on neighbouring frames. These multiple flow estimates are then fused by a random forest classifier to capture a generalized temporal information of the dynamic scene that gives a better estimation of flows in occluded regions.

In a more recent work, Zhile *et al.* [11] introduced a multi-frame extension of the two-frame learning based PWC-Net [8] method. In this approach, multiple forward and backward optical flows are estimated from three input frames using PWC-Net. These flow estimates are then warped at different levels and fused together to give a single flow estimate between the second and third frames. Zhile's multi-frame method outperformed PWC-Net and other two-frame methods of that time.

Recent state of the art methods for optical flow estimation are two-frame methods which include RAFT [1] and GMA [23]. RAFT introduced a 4D correlation volume based on the Kronecker product of feature maps to capture visual similarities between all-pairs of pixels in the two input frames. RAFT also formulated the optical flow estimation problem as a recurrent problem by which convolutional Gated Recurrent Unit (convGRU) blocks are used to iteratively estimate a residual flow direction from the visual similarity captured by the 4D correlation volume and context features. GMA extended the work of RAFT by introducing transformer network to capture the global dependencies of motion features in the image sequence. GMA outperformed RAFT and thus signified the importance of attention mechanism to capture long term temporal dependencies among input frames.

The following subsections discuss model features, techniques and theoretical backgrounds which formed the basis for our design of multi-frame optical flow estimation methods namely SSTM and SSTM++.

3.2.1 Separable 3D Convolutions

Assuming symmetry among space and time dimension filter coefficients, a 3D convolutional filter with a dimension of $t \times h \times w$ can be separated into a spatial filer of size $1 \times h \times w$,

and a temporal filter of size $t \times 1 \times 1$; where t represents the temporal depth of the 3D filter and h and w represent the height and width of the filter in spatial dimensions. When the symmetry approximation is valid, this reduces the number of learnable parameters from hwt to $hw + t$. Moreover, Xie *et al.* [24] showed that such separable filters are not only computational efficient, but also bring more accurate results when compared to the Inception 3D model [25] where 3D convolutional filters of the form $t \times h \times w$ are used to learn spatio-temporal features of input videos. More recent work used such separable filters as building blocks of spatio-temporal filters designs [26, 27, 28, 29, 30].

3.2.2 Spatio-temporal Filters

Spatio-temporal filters are 3D motion filters that can be applied in space-time domain to capture 2-dimensional spatial motion features (motion directions and magnitudes) and 1-dimensional temporal features (interpretation and generalization of the motion from the temporal queue) [4]. Such spatio-temporal filters are widely used to learn and represent motion features in videos and sequence of images of the same scene [4, 31, 10, 32, 24, 30].

One way of designing such 3D spatio-temporal filters is by using handcrafted 3D Gabor filters. Heeger *et al.* [4], for example, used multiple separable handcrafted spatio-temporal Gabor filters tuned for extracting motion patterns from a sequence of images. These filters consist of two separable components namely 3D Gaussian and sinusoidal components. The Gaussian part serves as an envelope function which localizes the filter over a specific space-time window. The sinusoidal term, on the other hand, specifies the spaio-temporal frequencies at which the filter gives the maximum output. Such 3D Gabor filters are given as:

$$f(x,y,t) = \frac{1}{\sqrt{2}\pi^{3/2}\sigma_x\sigma_y\sigma_t} \times \exp\{-(\frac{x^2}{2\sigma_x^2} + \frac{y^2}{2\sigma_y^2} + \frac{t^2}{2\sigma_t^2})\} \times \sin\left(2\pi(\omega_{x_0}x + \omega_{y_0}y + \omega_{t_0}t)\right) \tag{3.1}$$

Where $\omega_{x_0}, \omega_{y_0}$ and ω_{t_0} are the spatio-temporal central frequencies and σ_x, σ_y and σ_t are Gaussian space-time windows. By designing a family of such 3D Gabor filters with different

temporal frequencies and spatial orientations, Heeger was able to capture motion features of sequences of images in different directions.

With the advancement of deep learning methods, more recent methods for video analysis and optical flow estimation use learnable spatio-temporal filters using 3D convolutional neural networks to extract motion features. Zhaofan *et al.* [31] introduced pseudo 3D CNN spatio-temporal filters which consist of 3D convolution filters decoupled into 2D spatial and 1D temporal filters. By combining such decoupled spatial and temporal filters in residual connections, Zhaofan designed spatio-temporal filters for motion feature representation of videos. Teney *et al.* [10] designed 3D spatio-temporal learnable filters that capture different motion patterns in a sequence of multiple input images. These motion features are further fed to CNN to output dense optical flow estimations.

3.2.3 Correlation Volume

Correlation volume in optical flow estimation and stereo matching problems refers to a measure of match or similarity between corresponding pixels (or patch of pixels) from any two images of the same scene. FlowNet [6] introduced a correlation layer that matched a patch of pixels of a given size from the feature map of the first input image with the feature map of the second input image within a specified range. In this framework, given two feature maps, $fmap_1$ and $fmap_2$, correlation between two patches, $x_1 \subseteq fmap_1$ and $x_2 \subseteq fmap_2$, is computed as convolution between patches x_1 and x_2. With such operations, FlowNet [6], computed a 3D correlation volume of size $h \times w \times P^2$, where h, w and P are the height of the feature maps, width of the feature maps, and the neighbourhood size respectively. Similarly, PWC-Net [8] introduced a 3D correlation volume based on convolution operations, however, instead of matching feature maps of the first and second images like FlowNet [6], the correlation is computed between the feature map of the first image and the warped feature map of the second image using upsampled intermediate flow estimate.

In more recent work, RAFT [1] developed a 4D correlation volume based on pixel-wise Kronecker products of all-pairs of pixels of feature maps. In RAFT [1], to compute visual similarity between the extracted feature maps of consecutive input frames, a multi-layered

correlation pyramid is designed using down-sampled all pairs Kronecker products. Given the feature maps of the input frames, $fmap_1 \in \mathbb{R}^{H \times W \times D}$, and $fmap_2 \in \mathbb{R}^{H \times W \times D}$, a correlation volume, $C \in \mathbb{R}^{H \times W \times H \times W}$, is defined as:

$$C_{ijkl} = \sum_{d=1}^{D} (fmap_1)_{ijd} \, (fmap_2)_{kld} \tag{3.2}$$

where $i, k \in \{1, 2, .., H\}$ and $j, l \in \{1, 2, .., W\}$. A correlation pyramid of four layers $\{\mathbf{C^1}, \mathbf{C^2}, \mathbf{C^3}, \mathbf{C^4}\}$ is constructed by down-sampling the last two dimensions of the 4D correlation volume at different rates. For example, the k^{th} layer of this pyramid is down-sampled at $1/2^{k-1}$:

$$\mathbf{C^k} \in \mathbb{R}^{H \times W \times H/2^{k-1} \times W/2^{k-1}} \tag{3.3}$$

Each of the layers of this correlation pyramid capture information about both large (lower resolution layers) and small (higher resolution layers) displacements in the input sequence. In later stages, these layers are set to have the same dimensionality using bi-linear interpolation and are concatenated for further processing.

3.2.4 Attention

Ashish *et al.* [33] recently introduced transformers, which are a type of neural networks that are capable of solving sequence transduction problems based on attention mechanisms only, without having to rely on convolutional neural networks (CNNs) or recurrent neural networks (RNNs). In this work, transformers were introduced with the goal of solving Natural Language Processing (NLP) problems, more specifically machine translation problems. These transformer utilizes self-attention mechanism, a type of attention network which learns the global relationships and dependencies of different parts of a single input sequence. In other words, the query, key and value vectors of the attention module are coming from a single input sequence. By building multiple of such self-attention modules which run in parallel, Ashish

et al. designed a multi-head self-attention module which can be trained faster and achieve higher accuracy in language translation problems compared to its counterparts namely the CNN and/or recurrent based encoder/decoder architectures.

Following the introduction of transformers by Ashish *et al.* and its astonishing results in machine translation problems, more recent work adapted the use of such neural network architectures to solve different computer vision problems and achieved state of the art results. Such computer vision problems include object detection [34, 35, 36], object tracking [37, 38, 39], image segmentation [40, 41, 42] and optical flow estimation [23, 43, 44].

GMA [23] is the first published method to make a good use of transformers for optical flow estimation. GMA [23] introduced attention-module to the existing RAFT [1] optical flow estimation method and achieved approximately 15.8% improvement on the EPE results of both Sintel final and clean test datasets. Instead of the self-attention, GMA used a mechanism similar to cross-attention where two different sequences were used to generate the key, value and query vectors. However, instead of using the same sequence for key and value as most cross-attention modules, GMA used the same sequence for key and query and a second sequence for value. More specifically, the context features are used to generate the key and query vectors, and the value vector from the motion features of the original RAFT architecture.

3.3 SSTM: Model Features and Architecture

Figure 3.2 shows a schematic diagram of our SSTM architecture which is comprised of feature encoders and separable 4D correlation volumes for generating 3D motion features; and a spatio-temporal feature encoder for generating spatio-temporal context. Both the 3D motion features and spatio-temporal context information were used to update the hidden state of GRU modules and for generating an updated flow estimate. Detailed descriptions of these units as well as the loss function used for training the SSTM model are given below.

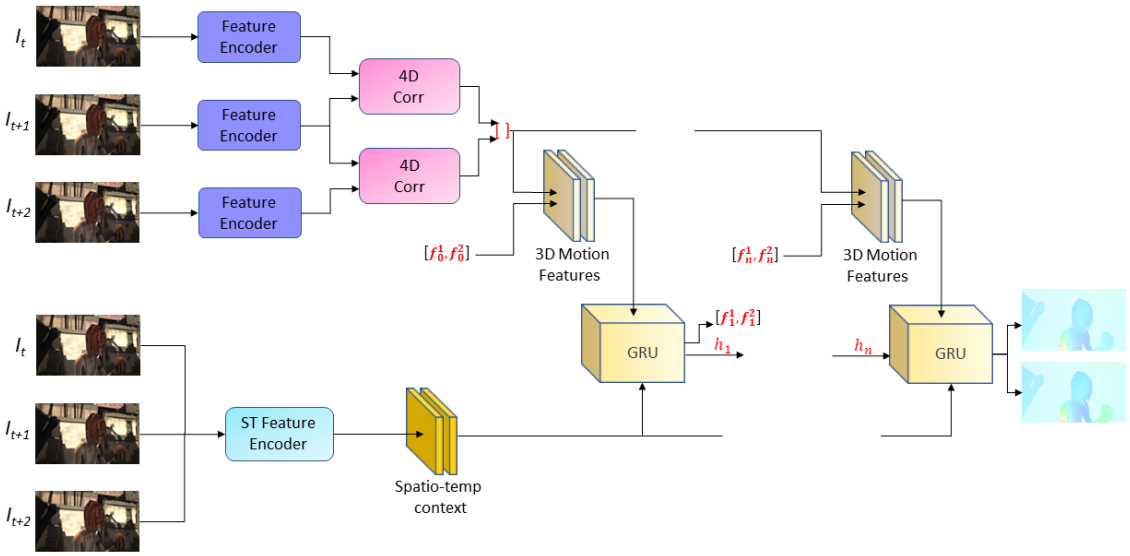


Figure 3.2: SSTM architecture: consists of 3D CNN Context feature encoder, two separate 4D correlation volumes, and 3D Conv GRU blocks. SSTM computes the correlation volume and context features from three input frames using the 4D correlation volume and context feature encoder. It then iteratively refines the optical flow estimate using the 3D conv GRU update blocks.

3.3.1 Feature Encoder and 4D Correlation Volume

, 3D motion features Given three input images I_1, I_2 and I_3, the feature encoder extracts spatial features of the input images at 1/8 resolution. The feature encoder is made of fully convolutional layers with different stride values such that the output feature maps are at 1/8 resolution of the input images. For input image $I_1 \in \mathbb{R}^{C \times H \times W}$, the feature encoder gives an output feature map, $fmap1 \in \mathbb{R}^{D \times H/8 \times W/8}$, for $D = 256$ number of feature maps. We then use a 4-D correlation volume as defined in RAFT [1] to measure two multi scale visual similarities, $\mathbf{C^1}$ and $\mathbf{C^2} \in \mathbb{R}^{D \times H/8 \times W/8}$ for D=324, between pairs of neighbouring feature maps in the sequence, $fmap1$ and $fmap2$, and $fmap2$ and $fmap3$, respectively. We further concatenate these two correlation volumes into, $\mathbf{C} = [\mathbf{C^1}, \mathbf{C^2}]$, along a temporal dimension such that, $\mathbf{C} \in \mathbb{R}^{D \times 2 \times H/8 \times W/8}$.

3.3.2 Spatio-temporal Context Feature Encoder

Our spatio-temporal context feature encoder learns both spatial and temporal features from multiple input images. We design such context feature encoder using four types of spatio-temporal feature extractor blocks, namely, SPT1, SPT2, SPT3 and SPT4. These blocks are made of 3D CNN layers decoupled into separate spatial and temporal layers with residual connections. Decoupling 3D convolutional layers in this fashion significantly reduced the high computational demands and memory cost that comes with using 3D convolutions. As we described earlier, the underlying filter coefficients were assumed to be symmetrical for decoupling 3D convolution operations. We minimized any approximation error due to our symmetric filter assumption by using multiple types of separable 3D convolutions with residual connections. For example, a $3 \times 3 \times 3$ 3D convolutional filter can be decoupled into 2D spatial filter of $1 \times 3 \times 3$ and 1D temporal filter of $3 \times 1 \times 1$. Similar types of blocks are used in spatio-temporal feature representations of videos [31, 29].

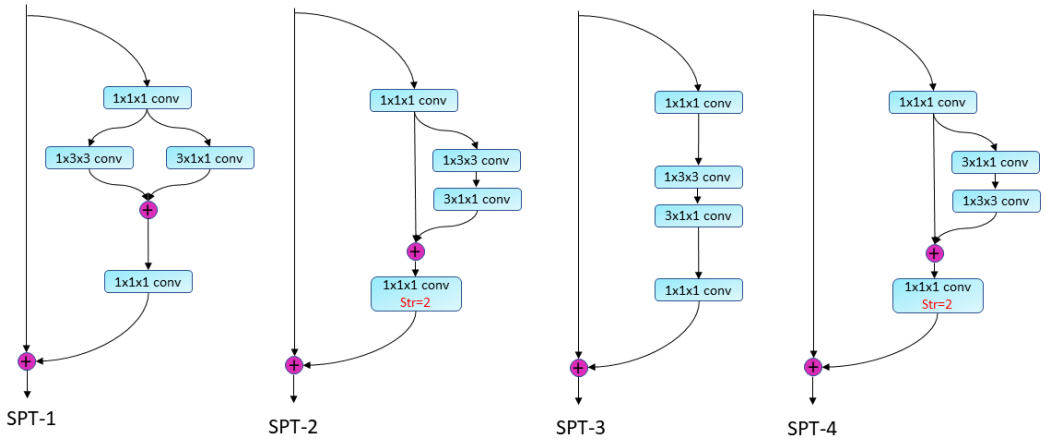


Figure 3.3: Four types of SPT blocks, SPT-1, SPT-2, SPT-3 and SPT-4 designed by different arrangements of the decoupled spatial and temporal filters with different residual connections and stride values.

The context encoder consists of 6 SPT blocks that are cascaded in a particular order with different stride values in such a way that the output feature map is at 1/8 spatial and 2/3

temporal resolution. The three input images, I_1, I_2 and I_3, are concatenated along a temporal dimension while retaining their temporal queue and fed to this encoder, $x \in \mathbb{R}^{C \times T \times H \times W}$, where C, T, H and W denote the number of color channels, the number of concatenated frames, height and width of each frame, respectively. Given the concatenated input, x, the cascaded SPT blocks output spatio-temporal context features, $context \in \mathbb{R}^{D \times 2 \times H/8 \times W/8}$ for $D = 128$ number of 3D feature maps that capture motion features. In this particular experiment, $T = 3$.

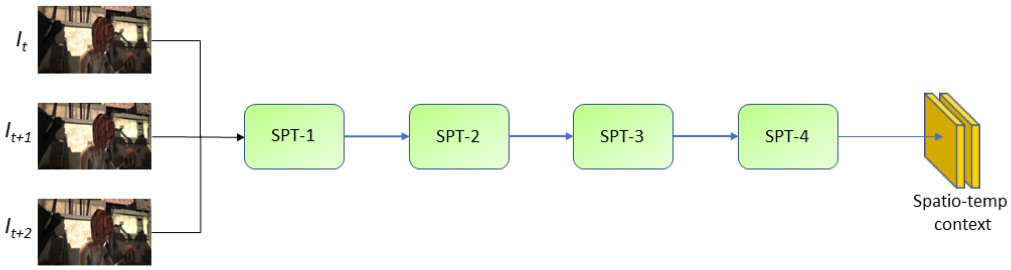


Figure 3.4: The spatiotemporal (SPT) blocks are cascaded to build the context encoder which extracts context features from three input frames.

3.3.3 3D Convolutional GRU Update Block

We introduce a 3D Convolutional GRU update block which uses a 3D spatio-temporal gating system. This update block contains N GRU blocks with tied weights such that the hidden state of the n^{th} 3D GRU, $h_n \in \mathbb{R}^{D \times M \times H/8 \times W/8}$, contains $D = 128$ number of 3D spatio-temporal hidden units with M number of motion features or optical flows estimated in parallel. In this particular setting, $M = 2$ as two optical flows are estimated from three input images. We believe that such spatio-temporal hidden units learn recurrent dependencies in space and time among neighbouring 3D points.

This GRU update block takes an input vector X_n at the n^{th} GRU block defined as the concatenation of the correlation volume, $\mathbf{C} = [\mathbf{C^1}, \mathbf{C^2}]$, previously estimated flows $F_{n-1} = [F^1_{n-1}, F^2_{n-1}]$, extracted context features, and multi-level brightness errors ϵ^{123}_{n-1} from previous

estimate flows as follows.

$$X_n = [\mathbf{C}, F_n, context, \epsilon_n^{123}] \tag{3.4}$$

Thus, the modified Update and Reset GRU equations are given as follows:

$$Z_n = \sigma(Conv3d([h_{n-1}, X_n], W_z)) \tag{3.5}$$

$$r_n = \sigma(Conv3d([h_{n-1}, X_n], W_r)) \tag{3.6}$$

where $[h_{n-1}, X_n]$ represents the concatenation of the input, X_n, and the previous state, h_{n-1}; and W_z represents filter coefficients (weights) in the 3D convolution layer. These 3D convolutional layers are decoupled in to 3 1D convolutional layers corresponding to x, y and t directions. Figure 3.5 shows our 3D Convolutional GRU comprised of two 1D spatial filters and one 1D temporal filter.

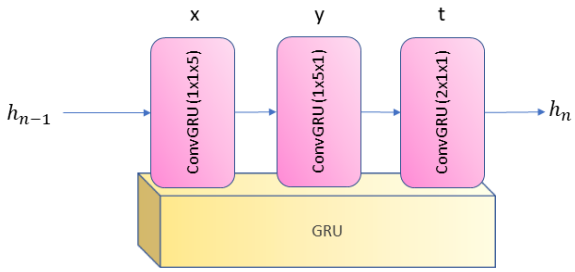


Figure 3.5: A 3D Convolutional GRU block built from two 1D spatial filters (along x and y directions) and one 1D temporal (along t direction) CNN filters.

The candidate hidden state and the output vector from level n are then calculated as follows:

$$h'_n = tanh(WX_n + r_n \odot h_{n-1}) \tag{3.7}$$

$$h_n = Z_n \odot h_{n-1} + (1 - Z_n) \odot h'_n \tag{3.8}$$

The output hidden state of the n^{th} GRU, $h_n \in \mathbb{R}^{D \times M \times H/8 \times W/8}$, then splits into two equal parts, h_n^1 and h_n^2, along the temporal dimension such that $h_n^1, h_n^2 \in \mathbb{R}^{D \times H/8 \times W/8}$. These two hidden states then pass through two convolutional layers with shared weights to give flow updates ΔF_n^1 and ΔF_n^2 respectively at level n. Both flow estimates at the nth level are updated as

$$F_n^1 = F_{n-1}^1 + \Delta F_n^1 \tag{3.9}$$

$$F_n^2 = F_{n-1}^2 + \Delta F_n^2 \tag{3.10}$$

$$F_n = [F_n^1, F_n^2] \tag{3.11}$$

We use $N = 12$ number of such GRU blocks with tied weights during training. Furthermore, the hidden states of these GRU blocks have residual connections at a certain interval. We believe that such residual connections play a major role in mitigating the vanishing gradient problem as the number of GRU blocks, N, grows. We define residual GRU connection at r interval as follows:

$$h_{n+k} = \begin{cases} GRU(X_{n+k-1}, h_{n+k-1}) + h_n & k = r, 2r, 3r, ... \\ GRU(X_{n+k-1}, h_{n+k-1}) & \text{Otherwise} \end{cases} \tag{3.12}$$

Figure 3.6 shows such residual GRU connections with a skip factor of $k = 2$.

3.3.4 Loss Function

The network generates a sequence of N optical flow estimates, $\{(\boldsymbol{f}_1^1, \boldsymbol{f}_1^2), (\boldsymbol{f}_2^1, \boldsymbol{f}_2^2), ..., (\boldsymbol{f}_N^1, \boldsymbol{f}_N^2)\}$ to arrive at the final optical flow estimate of $(\boldsymbol{f}_N^1, \boldsymbol{f}_N^2)$. Intermediate flow estimates from the

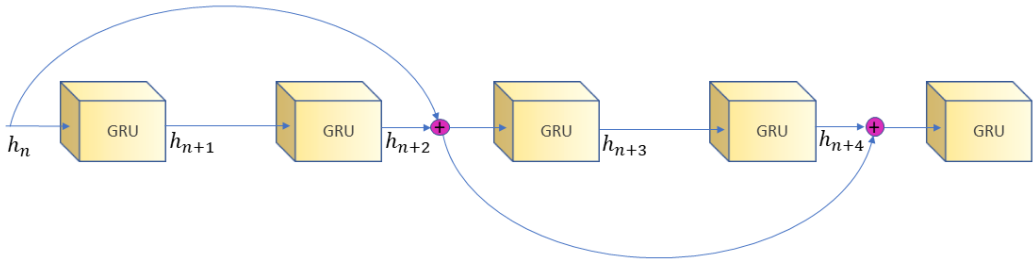


Figure 3.6: Illustration of the residual GRU connection with a skip factor of $k = 2$ in which the GRU hidden states are fed forward every two steps.

ith level $(\boldsymbol{f}_i^1, \boldsymbol{f}_i^2)$ is comprised of an intermediate flow estimate $\boldsymbol{f}_i^1$ between images I_1 and I_2; and an intermediate flow estimate $\boldsymbol{f}_i^2$ between images I_2 and I_3. Loss or objective function for supervised training was defined as the average L_1 norm between the estimated pair of optical flows and their ground truth values weighted by an exponential factor γ. The following loss function was used for end-to-end training of our networks.

$$Loss_1 = \sum_{i=1}^{N} \gamma^{N-i} \frac{(||\boldsymbol{f}_{gt}^1 - \boldsymbol{f}_i^1||_1 + ||\boldsymbol{f}_{gt}^2 - \boldsymbol{f}_i^2||_1)}{2} \tag{3.13}$$

For KITTI 2015, ground truth flows are available only for a single pair of images in every sequence with 20 images / sequence. To handle this limitation, we used a modified loss function while fine-tuning our models on KITTI 2015 training datasets. For image pairs I_2 and I_3 with ground truth flow f_{gt}^2, three images I_1, I_2 and I_3 were fed as input to the models for optical flow estimation. Therefore, we modified the loss function for fine-tuning on KITTI 2015 datasets as follows.

$$Loss_2 = \sum_{i=1}^{N} \gamma^{N-i} ||\boldsymbol{f}_{gt}^2 - \boldsymbol{f}_i^2||_1 \tag{3.14}$$

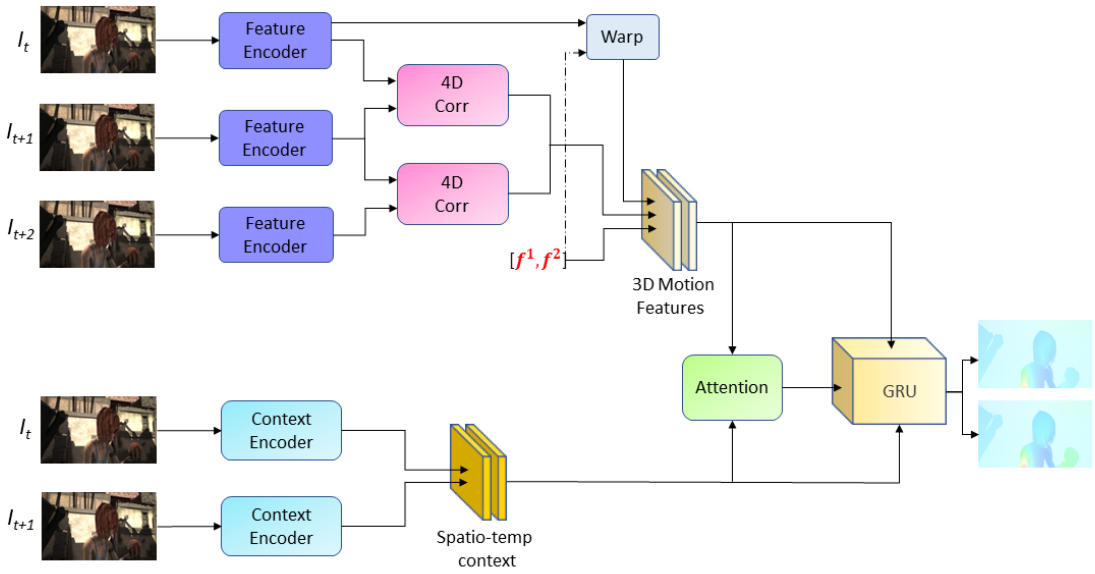


Figure 3.7: SSTM++ architecture: consists of two additional features compared to the original SSTM. The first addition was a *brightness error block* which computed using feature maps warped using intermediate flow estimates. The second addition was a *space-time attention mechanism* which extracted global dependencies of motion features in a given input scene. In SSTM++, we used two separate spatial context feature encoders from the first two frames.

3.4 SSTM++: Model Features and Architecture

Figure 3.7 shows the schematic diagram of our second method called as SSTM++. The core architecture of the SSTM++ model is same that of SSTM. In addition, SSTM++ has two additional modules namely a space-time attention network and a brightness error block. Our first hypothesis is that the SSTM++ method can improve over the SSTM method by using global motion features and identifying feature dependencies in both space and time using a space-time attention module. The second hypothesis is that the brightness error block can serve as an intermediate way of computing the optical flow estimation error at each GRU blocks and hence can improve the performance of the SSTM method. We believe that SSTM++ will benefit from these two additional designs compared to the original SSTM and achieve better optical flow estimates.

A detailed description of the additional attention network and the brightness based error block added to the SSTM++ model are given below.

3.4.1 Attention

We use space-time attention mechanism to learn the global motion dependencies over the input sequence. Similar to GMA [23], we used cross attention module where the query, key and value vectors come from two different input sequences. In this setting, the context feature is used to generate the query and key vectors, and the 3D motion features are used to generate the value vectors.

As stated in Section 3.3.2, context feature, $Context \in \mathbb{R}^{D\times T\times H/8\times W/8}$, extracted by the *spatio-temporal context encoder* have temporal length of $T = 2$. This context feature is split into two spatio-temporal context features along the temporal dimension. Let these split context features be X^1 and X^2. Given the context features, $context = [X^1, X^2]$, and the 3D global motion features, $M = [M^1, M^2]$, built from the intermediate flow estimates and the correlation volume, our attention module captures global dependencies of the motion sequence in two time windows that correspond to the optical flow between I_1 and I_2 and I_2 and I_3.

We first flatten both the context and 3D motion features along the spatial dimensions such that at a time window, t, the flattened context and 3D motion features are given as $X^t \in \mathbb{R}^{N\times L_c}$, and $M^t \in \mathbb{R}^{N\times L_m}$, for $N = HW$, L_m and L_c denoting the flattened channel dimension and M represents the 3D motion features.

For projection functions $\theta(.)$, $\phi(.)$ and $\sigma(.)$, we define the query, key and value vectors as:

$$\theta(C^t) = \mathbf{W_q} C^t \tag{3.15}$$

$$\phi(C^t) = \mathbf{W_k} C^t \tag{3.16}$$

$$\sigma(M^t) = \mathbf{W_v} M^t \tag{3.17}$$

where, W_q, W_k and W_v are the query, key and key projection vectors respectively. For input frames I_t and I_{t+1}, the aggregated global motion feature Y^t is given as

$$Y^t = M^t + \alpha \sum_{j=1}^{N} f(\theta(C^t), \phi(C^t))\sigma(M^t) \tag{3.18}$$

Finally, the context C, global motion features (Y), and motion features (M) are concatenated as

$$output = \big[[C^t|Y^t|M^t], [C^{t+1}|Y^{t+1}|M^{t+1}]\big] \tag{3.19}$$

and fed as a input to the GRU block between two time windows t and $t+1$.

3.4.2 Error Block

Our error block uses warping operations to compute multiple brightness errors at different levels. FlowNet2 [7] showed how feeding brightness errors to multiple stacks of networks improve the accuracy of estimated optical flows. In FlowNet2, brightness error is computed as the difference between the first image in the sequence and an estimate of the second image obtained by warping the first image using the current flow estimates. In our method, however, we compute brightness error as the difference between the feature map $fmap1$ and an estimate of the feature map $fmap1$ obtained by warping $fmap2$ using the optical flow estimate from level n computed as

$$\hat{fmap1} = W(fmap2; F_n^1) \tag{3.20}$$

$$= fmap2(x(t) + u(t), y(t) + v(t)) \tag{3.21}$$

where, $fmap2$ represents the feature maps extracted from the second input image; and $u(t)$ and $v(t)$ represent the components of the n^{th} flow estimate F^1 in the x and y directions respectively (x and y components of F_n^1). This warping operation gives us an estimate of $fmap1$ based on the current estimated flow. Therefore, we can compute the brightness error

ϵ_n^1 at level n as:

$$\epsilon_n^1 = \left\| W(fmap2; F_n^1) - fmap1 \right\| \tag{3.22}$$

By taking advantage of the fact that our method takes multiple input images and estimates two output optical flows, we compute the following additional brightness errors at each level n as:

$$\epsilon_n^2 = \left\| W(fmap3; F_n^2) - fmap2 \right\| \tag{3.23}$$

$$\epsilon_n^3 = \left\| W(W(fmap3; F_n^2); F_n^1) - fmap1 \right\| \tag{3.24}$$

We feed this multi-level brightness errors $\epsilon_n^{123} = [\epsilon_n^1, \epsilon_n^2, \epsilon_n^3]$ to the GRU update block described in the next section.

Chapter 4

Experiments

Our multi-frame methods for optical flow estimation, SSTM and SSTM++ were implemented in PyTorch. These models were trained and validated using two RTX 8000 GPUs. In this section, we present details about the procedures used for training our models, and the details of ablation experiments conducted to validate the significance of various modules and intermediate features. In addition, we present qualitative and quantitative results of our models and compare them with other recent state of the art methods (RAFT, GMA, and MFF).

4.1 Training Schedule

We used the standard training approach similar to recent optical flow estimation methods [6, 8, 1] in which we trained and validated our model in four training stages. All models were trained with goal of conducting a final test on Sintel and KITTI 2015 datasets (target datasets). In the first two training stages, we used training datasets (Virtual KITTI and Monkaa) which are different from target test datasets (Sintel and KITTI 2015). These datasets are larger in size and they differ in their image characteristics when compared to the target datasets. This helps our model to have a better generalization on unseen input images / sequences with differing features and motion patterns.

Due to the multi-frame nature of our methods, we require the training datasets to have multiple input images of each scene or sequence. The training datasets, FlyingChairs [6]

and 3DFlyingThings [19], commonly used for training two-frame methods contain only two images per sequence. Therefore, instead of using the FlyingChairs and FlyingThings3D datasets, we used the Virtual KITTI2 and Monkaa datasets in the first two stages of the training. These two datasets are sufficiently larger in size for initial training, however, they contain much fewer number of scenes compared to the FlyingChairs and FlyingThings datasets. The last two stages of our training schedule include finetuning on target datasets for final evaluation. On the third stage, we finetune the result from first two stages (V+M) using Sintel training dataset. Finally, we finetuned the result from these three stages on KITTI 2015 datasets. In each of the four training stages, validation is done after every 5k training iterations. A summary of these training stages is shown in Table 4.1.

4.1.1 Stage i: vkitti

In the first training stage, we used Virtual KITTI2 (V) datasets for both training and validation. We split the Virtual KITTI2 dataset into non-overlapping training (90%) and validation (10%) datasets. The training is done for 100k iterations on 38,035 triplet images with a resolution of 288×960. During this training stage, we used eq. 3.13 to compute the loss function. Moreover, we used a batch size of 8 and a learning rate of 4×10^{-4} which is relatively larger compared to the later stages of training.

4.1.2 Stage ii: monkaa

The second stage (V+M) of our training is done on Monkaa dataset by initializing the network weights from Stage i. At this training stage, we used Sintel training dataset for validation. The monkaa training data contains 34,384 triplet input samples with a resolution size of 456×720. Since each triplet input in this set has two ground truth optical flow values, we used the loss function described in eq. 3.13. After we train this stage for 100k iterations with a batch size of 6 and a learning rate of 1.2×10^{-4}, we tested and reported the result on unseen target data (Sintel and KITTI 2015) as shown in Table 4.4.

4.1.3 Stage iii: sintel

In the third training stage, we initialized the network weights from Stage iiand finetuned using a mixture of datasets from Sintel training, Virtual KITTI2 validation, Monkaa and HD1K datasets (V+M+S+H) with the ratio of 0.13, 0.13, 0.72 and 0.02 respectively. As all these benchmark datasets have ground truth for multiple input images of the same scene, we used the loss function in eq. 3.13. In this stage, we used a batch size of 6 with a learning rate of 1.2×10^{-4} for 120k iterations. The sintel-finetuned model was evaluated using testing datasets provided by the Sintel leaderboard (http://sintel.is.tue.mpg.de/results). We reported the leaderboard results in Table 4.4.

4.1.4 Stage iv: kitti

In the last stage of our training schedule (V+M+K), we initialized the network weights from Stage iiiand finetuned using KITTI2015 training dataset. We trained this stage with a batch size of 6 and a learning rate of 2×10^{-4}. KITTI training dataset has ground truth flows only for a pair of images among all 20 images available per sequence. To accommodate this difference in the KITTI training dataset, we used the loss function defined in 3.14 which computes the training loss based on only one of the two flow estimates and the available ground truth. The kitti-finetuned model was evaluated using testing datasets provided by the KITTI leaderboard (https://www.cvlibs.net/datasets/kitti/eval_flow.php)

Table 4.1: Summary of training schedule. $Loss_1$ and $Loss_2$ refer to loss functions defined in eq. 3.13 and eq. 3.13. V, S, M, K and HD1k refer to Virtual KITTI2, Sintel, Monkaa, KITTI2015 and HD1k datasets respectively.

Stage	Training data	Validation data	Loss function	Input size	Batch size	Initialized weights	Training samples (triplets)
vkitti	V	V	$Loss_1$	[288, 960]	8	-	38,035
monkaa	M	S	$Loss_1$	[456, 720]	6	vkitti	34,384
sintel	V+M+S+HD1K	S	$Loss_1$	[368, 768]	6	monkaa	147,836
kitti	K	K	$Loss_2$	[288, 960]	6	sintel	200

4.2 Ablation Experiments

To determine the significance of each of the main parts of our design, we conducted ablation experiments by training different versions of our methods on Virtual KITTY and Monkaa datasets and evaluated the result on Sintel (Clean and Final) and KITTI2015 training datasets. In brief, various ablated models were created to assess the significance of using 1) 3D convolution in place of 2D convolution while building the context features, 2) attention features, and 3) feature warping for estimating training loss. The list of features used to build different versions of our methods is shown in Table 4.2.

We first trained each of these variants of our methods for 50k iterations with batch size of 8. We then finetuned this result on Monkaa dataset with a batch size of 6 while validating on Sintel training dataset. The validation result on Sintel dataset is then used to compare these features and their significance on the final model. Moreover, we also compared the total number of learnable parameters with or without the ablated features listed in table 4.2. These ablation experiments are conducted on two RTX 8000 GPUs. Quantitative performance of the models with and without ablated features is shown in Table 4.2. A detailed description of the ablted features are presented in the following subsections.

4.2.1 Context

We designed two types of spatio-temporal encoders to capture the context information from the image sequences. The first type of encoder utilized 2D Convolution to capture the spatial features of the first two images in the sequence, I_1 and I_2. These images pass through 2D CNN blocks with shared weights. The output spatial feature maps are then concatenated along an addition temporal dimension to form a 3D spatio-temporal features. In the second context encoder design, on the other hand, we used a 3D CNN based architecture as described in Figure 3.4 to capture the spatio-temporal context features. In this configuration, all the three images were concatenated and fed to the cascaded SPT blocks.

4.2.2 Warping

Our Error block defined in Section 3.4.2 introduces a brightness error by warping features using multi-level optical flow estimates. In our ablation experiment, we compared the effect of feeding a multi-level brightness error to the 3D motion extractor block.

4.2.3 Attention

We assessed two models to analyze the effect of our space-time attention module on the number of learnable parameters, inference time and standard results on Sintel and KITTI2015 training datasets. In the first model, we used a spatio-temporal attention module and a 2D CNN context encoder. In the second model, we used a 3D CNN context encoder without any attention module.

Table 4.2: Ablation experiment results. Trained on V+M (80k on Virtual KITTI and 50k on Monkaa) and evaluated on Sintel and KITTI 2015 training datasets. Inference time is measured using Sintel sequences (clean and final). Underlined features are used in the final SSTM++ (with attn) method, and features with '*' are used in the final SSTM (without attn) method.

Experiment	Feature	Sintel (train)		KITTI-15 (train)		Parameters (M)	Inference time (s)
		Clean	Final	AEPE	Fl-all(%)		
Context	2D Convolution	2.17	2.94	5.92	18	5.61	0.38
	3D Convolution*	1.88	2.81	5.58	19	5.12	0.34
Attention	No attention*	1.88	2.81	5.58	19	5.12	0.34
	Spatio-temp	2.17	2.94	5.92	18	6.09	0.38
Warping	No warping*	2.56	3.19	5.32	17.1	5.89	0.32
	Warping	2.17	2.94	5.92	18	6.09	0.38

4.3 Inference Time and Number of Learnable Parameters

Inference time in this experiment refers to the average amount of time our trained model took to estimate an optical flow field between two given images at a specified resolution / size. As our method takes three input images and estimates two optical flow fields at each

inference, we calculated inference time as the total inference time divided by the number of flows estimated. This experiment is done using 2 RTX 8000 GPUs on Sintel-clean and Sintel-final training datasets with resolution size of 368×768. We also reported the number of learnable parameters in our model and compared them with the recent state of the art methods for optical flow estimation. These results are shown in Table 4.3.

Table 4.3: Average inference time for a single flow estimate from a pair of input frames from the Sintel training dataset (clean and final), learnable parameter count and GPU memory from the training stage "sintel" with a batch size of 6.

Method	Parameters (M)	Inference time (s)	GPU Memory (GB)
RAFT [1]	5.3	0.38	**11**
GMA[23]	5.9	-	-
Ours (no attn)	**5.2**	**0.34**	23
Ours (attn)	6.1	0.38	23

4.4 Training and Testing Results

We evaluated the performance of our multi-frame optical flow estimation models on the current benchmark datasets using the standard optical flow evaluation metrics. We used Sintel and KITTI 2015 training and testing datasets to evaluate our methods and reported the EPE and Fl-all evaluation results. For Sintel training dataset, we reported the average EPE on both clean and final datasets at different training stages. For KITTI 2015 training dataset, we reported both the EPE and Fl-all errors. We also reported the EPE results on the testing data evaluated by the Sintel leaderboard and both the Fl-all and Fl-fg results reported by KITTI leaderboard. These results were compared with the most recent published state-of-the-art methods for optical flow estimation. In Table 4.4, we present these quantitative evaluation results on Sintel and KITTI test benchmark datasets from Sintel leaderboard and KITTI leaderboard respectively.

Furthermore, we report the EPE, Fl-all and Fl-fg results on different regions of the input frames on both Sintel and KITTI benchmark datasets. Table 4.5 shows the performance of our method in comparison with other methods on Sintel final test datasets near occluded regions

and regions with different velocities. Table 4.6 shows Fl-all, Fl-fg and Fl-bg percentage estimated flow outliers on both all regions and non-occluded regions only in the input frames.

Table 4.4: Performance of optical flow models on Sintel and KITTI 2015 benchmark datasets. 'V+M' refers to training on Virtual KITTI2 and Monkaa (analogous to Chairs+Things used by two frame methods). 'V+M+S/K (+H)' (analogous to 'C+T+S/K (+H)' used by two frame models) refers to using specific dataset during finetuning the model on Sintel using a mixture of datasets including Sintel, Virtual KITTI, Monkaa and HD1K. The models were finetuned on KITTI using the KITTI dataset only. Results with * refer to warm-start flow initialization as defined in RAFT [1]. "Ours (no attn)" refers to our SSTM model shown in Figure 3.2 and "Ours (attn)" refers to the SSTM++ design shown in Figure 3.7.

Training data	Method	Sintel (train)		KITTI-15 (train)		Sintel (test)		KITTI-15 (test)	
		Clean	Final	APE	Fl-all(%)	Clean	Final	Fl-fg (%)	Fl-all (%)
C+T	FlowNet2 [7]	2.02	3.54	10.08	30.0	-	-	-	-
	RAFT [1]	1.43	**2.71**	5.04	17.4	-	-	-	-
	GMA [23]	**1.3**	2.74	**4.69**	**17.1**	-	-	-	-
V+M	RAFT [1]	2.21	3.21	7.03	23.7	-	-	-	-
	Ours(no attn)	1.87	2.86	5.55	19	-	-	-	-
	Ours(attn)	2.1	2.97	5.46	17.6	-	-	-	-
C+T+S/K (+H)	FlowNet2 [7]	1.45	2.01	2.30	6.8	4.16	4.74	-	11.48
	RAFT [1]	0.76	1.22	0.63	1.5	1.61*	2.86*	6.87	5.10
	GMA [23]	**0.62**	1.06	**0.57**	**1.2**	**1.39***	2.47*	7.03	5.15
	CRAFT [45]	**0.6**	1.06	0.58	1.34	1.45	**2.42**	**4.58**	4.79
V+M+S/K (+H)	MFF [11]	-	-	-	-	3.43	4.57	7.25	7.17
	RAFT [1]	0.98	1.39	0.58	1.18	2.86	4.19	8.65	6.85
	Ours(no attn)	0.67	1.03	0.56	1.18	2.13 2.30*	3.08 3.32*	7.20	**5.02**
	Ours(attn)	**0.65**	**0.91**	**0.54**	**1.12**	**2.03** 2.65*	**2.94** 3.21*	**7.04**	5.04

Table 4.5: Sintel leaderboard results on Sintel-final test benchmark dataset. d_{m-n} represents EPE near occluded boundaries with a distance ranging m to n pixels. s_{m-n} represents EPE over regions with velocities between m to n pixels per frame as defined in Section 2.3. Results written in bold are the best results among the listed methods and the ones underlined are second to the best.

Method	d_{0-10}	d_{10-60}	d_{60-140}	s_{0-10}	s_{10-40}	s_{40++}
MFF [11]	4.664	2.017	1.222	0.893	2.902	26.810
RAFT-VM	3.890	1.624	1.192	0.724	2.470	25.543
RAFT[1]	3.112	1.133	<u>0.770</u>	0.634	<u>1.823</u>	**16.371**
Ours(no attn)	<u>2.992</u>	<u>1.129</u>	0.843	<u>0.553</u>	1.935	18.356
Ours(attn)	**2.917**	**1.076**	**0.648**	**0.511**	**1.660**	<u>18.022</u>

Table 4.6: KITTI2015 leaderboard results on KITTI2015 test benchmark dataset. 'non-occ' refers to non-occluded regions and 'all' refers to all pixels. Fl-bg%, Fl-fg% and Fl-all% refer to percentage outliers in the background, foreground and all regions respectively as defined in Section 2.3.

Method	Fl-bg % (non-occ)	Fl-fg % (non-occ)	Fl-all % (non-occ)	Fl-bg % (all)	Fl-fg % (all)	Fl-all % (all)
MFF [11]	4.52	4.25	4.47	7.15	7.25	7.17
RAFT-VM	4.30	5.29	4.48	6.49	8.65	6.85
RAFT[1]	<u>2.87</u>	<u>3.98</u>	**3.07**	4.74	**6.87**	5.10
GMA[23]	2.97	**3.80**	3.12	4.78	<u>7.03</u>	5.15
Ours(no attn)	2.93	4.02	3.13	**4.58**	7.20	**5.02**
Ours(attn)	**2.83**	4.16	<u>3.08</u>	<u>4.64</u>	7.04	<u>5.04</u>

4.5 Visual Results and Qualitative Comparisons

In this section, we present visual examples of estimated flows on both the KITTI2015 and Sintel test benchmark datasets. We also qualitatively compare our estimated flow results with other recent state-of-the-art methods. Figures 4.1 and 4.2 show estimated optical flows on KITTI2015 and Sintel test datasets respectively. In Figures 4.3 and 4.4, we qualitatively compared our estimated optical flow results on sample Sintel and KITTI test dataset respectively. We outlined regions of interest where our estimate shows significant improvement over the other methods within the region marked with a red boundary.

Figure 4.1: Sample frames from KITTI2015 test dataset and their corresponding optical flow estimates using our SSTM method.

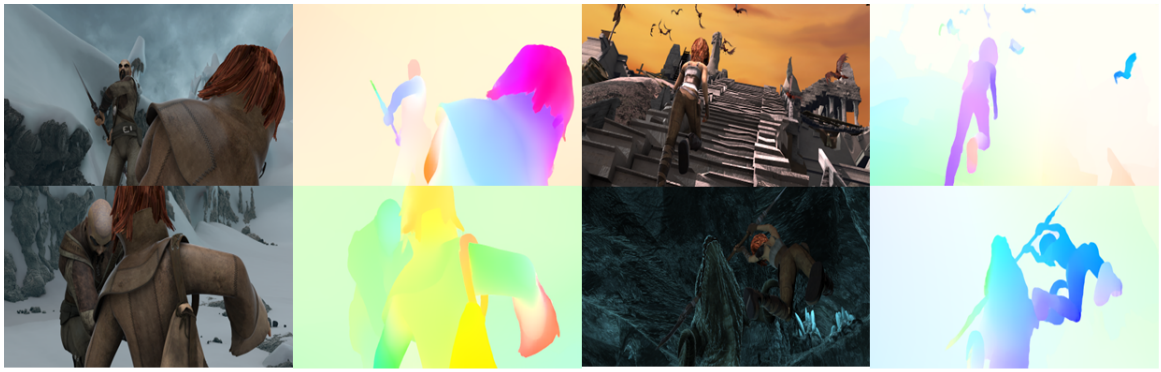


Figure 4.2: Results on sample frames from Sintel test dataset using our SSTM++ method.

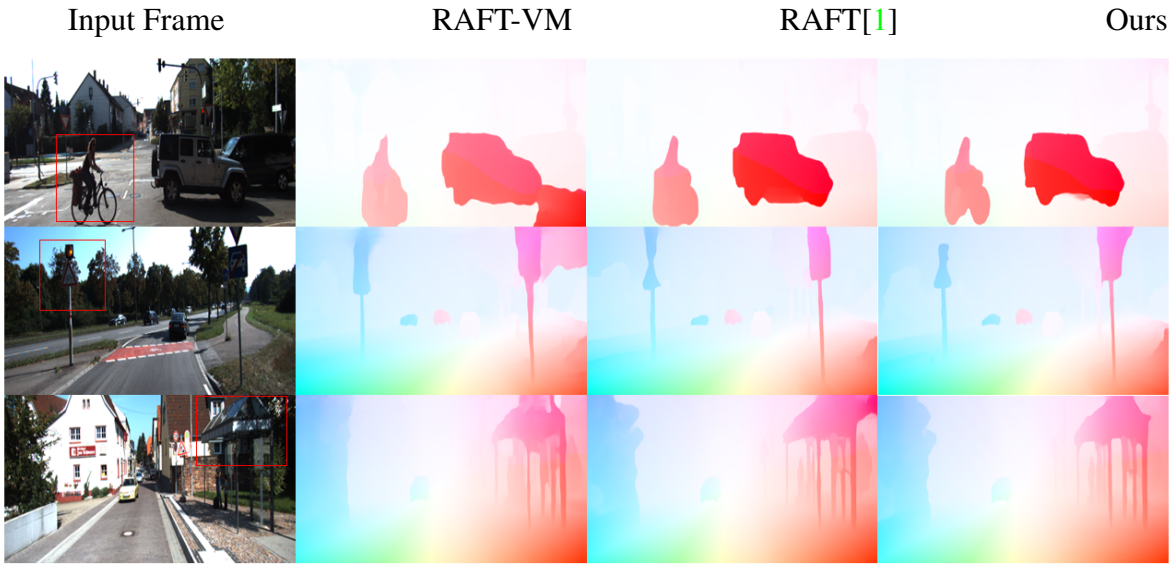


Figure 4.3: Visual comparison of optical flow estimates from various methods on sample KITTI2015 test dataset. Frame42, Frame63 and Frame81 (first column top to bottom respectively). The three results shown are from methods RAFT-VM, RAFT [1] and Ours (from left to right respectively). The regions bounded by red boxes in the input frames represent the regions where our method significantly outperformed the other two methods.

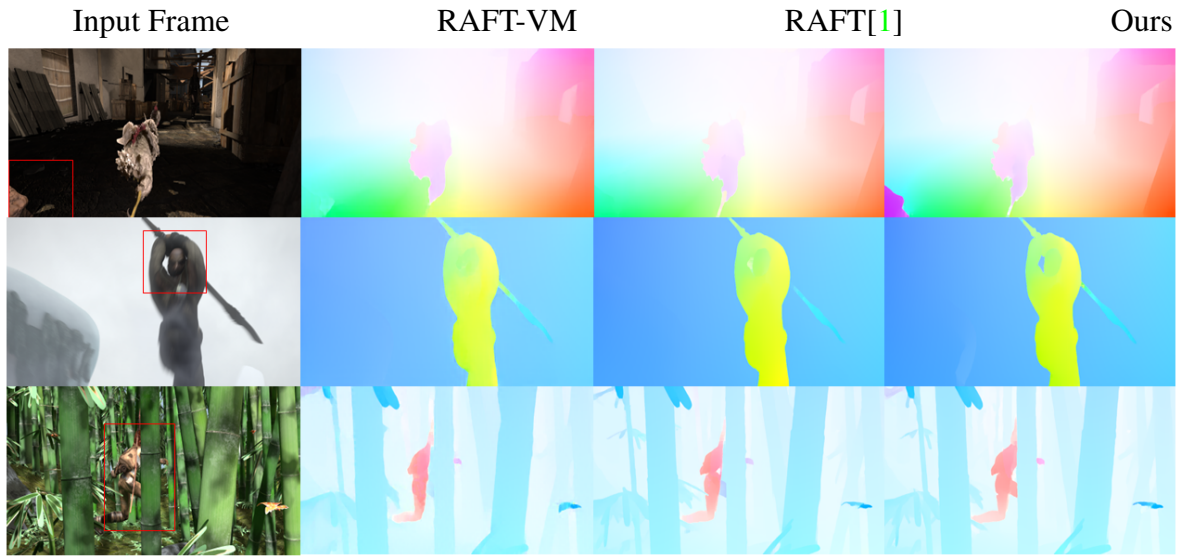


Figure 4.4: Visual comparison of optical flow estimates from various methods on sample Sintel test dataset. The three results shown are from methods RAFT-VM, RAFT [1] and Ours from left to right respectively on sample testing datasets from Sintel. The red box regions in the input frames show regions where our method significantly outperformed the other two methods.

Chapter 5

Discussion

5.1 Results on Sintel

Our multi-frame methods, both SSTM and SSTM++, provided the best EPE results on Sintel clean and final datasets compared to other multi-frame methods as well as a recent two-frame state of the art method (RAFT-VM) when trained using the same training stages and datasets (V+M+S/K+H) as our SSTM and SSTM++ methods. Our SSTM method show up to 40% improvement on EPE results for Sintel clean and final datasets. More detailed quantitative evaluation results on Sintel final test benchmark dataset are shown in Table 4.5. Near the occluded regions that are 0 to 60 pixels away from occluded boundaries, both SSTM and SSTM++ methods outperformed all other listed methods by up to 6.6 % (on d_{0-10}) and 5.3 % (on d_{10-60}). These results strongly support our hypothesis that our multi-frame approach has a better generalization near occluded regions. Moreover, our methods achieved a superior performance with limited and less diverse training datasets (V+M) when compared with the original RAFT method [1] which was trained on a larger and more diversified datasets (C+T). Thus, we believe that, in future, with the availability of larger datasets for optical flow estimation with multi-frames, our model can be trained on such large datasets with diverse image characteristics to provide a more accurate optical flow estimates.

5.2 Results on KITTI

As shown in Table 4.6, our Fl-all result in non-occluded regions, Fl-all (non-occ), on the KITTI test dataset is comparable with other top performing recent state of the art methods (slightly worse). Both of our methods, SSTM and SSTM++, however, outperformed all listed methods based on the Fl-all (all) measure in both occluded and non-occluded regions. This implies that our multi-frame methods are performing better in occluded regions thus supporting our hypothesis that occluded regions can be better understood with multi-frame based optical flow estimation.

5.3 Effect of Warm-start

RAFT [1] introduced a warm-start procedure to initialize the flow estimates using previously estimated optical flows. For example, having the flow estimate F_1 between input frames I_1 and I_2, the method uses this flow to initialize the second flow to be estimated, F_2, between the next consecutive input frames in the sequence, I_2 and I_3. Such initialization resulted in a significant improvement of the final results. RAFT [1] reported approximately 20.5% and 11.2% improvements on EPE results of Sintel clean and final test benchmark datasets, respectively, just by using warm-start initialization.

We experimented the effect of such flow initialization in our multi-frame optical flow estimation method for Sintel test dataset. As our Multi-frame approach estimates a pair of flows in parallel at each inference instants (unlike two frame methods which estimate each flow sequentially), our flow initialization is done over two time step gaps. In other words, given two flow estimates F_1 and F_2 from the first input triplet images, I_1, I_2 and I_3, the next flow estimates F_3 and F_4 are initialized by F_1 and F_2 respectively. However, our results did not benefit much from such warm-start initialization as expected. In fact, in both reported methods (with and without attention), our method performance was lowered by up to 30% compared to the baseline where zero flow initialization was used. We believe that this is due to the two step temporal gap between the previously estimated flow and the initialized flow,

(between F_1 and F_2, F_2 and F_4). Given the large and non-rigid motions in the Sintel dataset, dynamics in the scene undergoes significant changes within two temporal steps. More likely, this resulted in a wrong flow initialization causing the method to use a less accurate initial estimate.

5.4 Effects of Training Datasets

One major challenge in developing a multi-frame based optical flow estimation method is that there are not many training datasets currently available which contain large number of scenes (or sequences) as well as with multiple frames per scene. The standard training benchmark datasets for two-frame based optical flow estimation are 3DFlyingThings [19] and Flying Chairs [6] datasets. These two datasets consist of 21,818 and 22,872 pairs of images from 2,247 and 809 number of distinct scenes respectively [19, 6]. On the other hand, the multi-frame datasets that we used for training our multi-frame optical flow estimation methods namely Virtual KITTI2 [17] and Monkaa [19], contain only 5 and 8 number of scenes respectively. With fewer distinct scences in the training datasets, the model had a fewer number of distinct moving objects and motion patterns available for learning to estimate optical flow. Even with this training data limitation, our models were able to provide a better performance without overfitting and loss of generalization (Table 4.4). Therefore, with a more diverse multi-frame training datasets, the performance of our models will likely improve significantly.

To quantitatively illustrate the effect of using these two different training datasets, we trained recent two-frame state of the art optical flow estimation method, RAFT [1] (originally trained on 3DFlyingThings and Chairs), on Virtual KITTI2 and Monkaa according to our training stages as discussed in Table 4.1 (we call this method RAFT-VM). All other hyper-parameters namely the batch size, loss function, learning rates and weight decay are kept the same as in the original RAFT method. As shown in Table 4.4, RAFT-VM method underperformed when compared to the original RAFT by approximately 40% and 32% on Sintel Clean and Final testing benchmark datasets respectively as reported from the Sintel

leaderboard server. This result signifies and highlights the possible limitations and differences when using these two datasets (i.e. when using V+M instead of C+T) for training.

Despite this limitation in available training datasets for multi-frame optical flow methods, our method achieves comparable results with RAFT on Sintel clean and final test benchmark datasets and outperforms both RAFT and GMA [1, 23] on KITTI test benchmark dataset. Moreover, our method outperforms RAFT-VM (RAFT trained on V+M+S/K(+H)) by up to by 40% and 42% on Sintel Clean and Final test benchmark datasets. From this observation, we believe that given larger multi-frame training datasets than the existing ones which comprised of large number of scenes with distinct objects and different motion patterns, our multi-frame method can achieve state of the art results.

5.5 Effect of Attention Mechanisms

Our multi-frame based optical flow estimation method introduced two final methods namely, SSTM and SSTM++. The main difference between these two methods is the use of space-time attention mechanism in SSTM++, which is not part of the SSTM design. Our initial hypothesis is that by introducing space-time attention networks to our SSTM method, we can have a better understanding of the global dependencies of the input sequences in space-time domain and have a more accurate and generalized optical flow estimation, especially around occluded regions. Our result on Sintel test benchmark dataset supports this hypothesis. Results in Table 4.4 show that SSTM++ improved the EPE result approximately by 5% on both Sintel final and clean test results. More detailed results in table 4.5 show that SSTM++ outperforms SSTM in all listed regions around occluded boundaries and velocities. However, we did not observe a similar result improvement in KITTI test dataset. In fact, SSTM performs slightly better than SSTM++ on KITTI. We believe that the shorter scene lengths in KITTI is primarily making it hard for SSTM++ to understand longer space-time dependency of each scene (i.e. difficulty in associating similar features that are separated by larger extents in space and/or time).

5.6 Inference Time and GPU Memory Usage

At each inference instants, our multi-frame based methods take three input images in parallel and estimate two optical flows. The first optical flow estimate is between first and second input images and the second optical flow estimate is between second and third input images. This inference scheme is faster than the two-frame methods where each pair of input frames are fed sequentially and a single optical flow is estimated at each inference instants. Results in Table 4.3 show that during inference, our methods are relatively faster than RAFT [1]. On the other hand, analyzing the GPU memory usage during training stages, we observed that our multi-frame methods consume relatively larger GPU memory for the same batch size. This is because a single input batch size in our case is equivalent to three images while in two-frame methods it's two images. Thus, for the same batch size during training, our methods utilize a higher GPU memory.

Another observation we made is on the GPU memory usage of 3D CNNs vs 2D CNNs architectures for context feature extraction. As described earlier, we used a 3D CNN context encoder in our SSTM method and a 2D CNN context feature encoder in our SSTM++ method. By using our 3D CNN based spatio-temporal context encoder as show in Figure 3.4, we were able to get comparable result with the 2D CNN based context encoder while reducing the learnable parameters by about half a million. However, the 3D CNN based method had a higher GPU memory usage during training.

Chapter 6

Conclusion

In this work, we introduced two multi-frame optical flow estimation methods that can better capture the temporal dependency of the input images in a sequence and can generalize any non-linear motion patterns. These salient features of our methods allow them to better understand flow estimates in occluded and out of boundary regions. Based on qualitative and quantitative assessment using standard benchmark optical flow datasets, our methods achieved higher performance than the state of the art multi-frame methods. Moreover, our methods outperformed recent two-frame state of the art methods (when trained using the same training datasets as our multi-frame methods) in occluded and out of boundary regions while achieving comparable results in other regions. We believe that, in future, with the availability of larger training datasets with multiple input sequences from various scenes, multi-frame based methods can achieve a more generalized and accurate optical flow estimations when compared to two-frame methods.

Bibliography

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

[2] Edward H Adelson and James R Bergen. Spatiotemporal energy models for the perception of motion. *Josa a*, 2(2):284–299, 1985.

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

[4] David J Heeger. Optical flow using spatiotemporal filters. *International journal of computer vision*, 1(4):279–302, 1988.

[5] Temujin Gautama and MA Van Hulle. A phase-based approach to the estimation of the optical flow field using spatial filtering. *IEEE transactions on neural networks*, 13(5):1127–1136, 2002.

[6] 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.

[7] 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.

[8] 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.

[9] Ali Salehi and Madhusudhanan Balasubramanian. Ddcnet: Deep dilated convolutional neural network for dense prediction. *arXiv preprint arXiv:2107.04715*, 2021.

[10] Damien Teney and Martial Hebert. Learning to extract motion from videos in convolutional neural networks. In *Asian Conference on Computer Vision*, pages 412–428. Springer, 2016.

[11] Zhile Ren, Orazio Gallo, Deqing Sun, Ming-Hsuan Yang, Erik B Sudderth, and Jan Kautz. A fusion approach for multi-frame optical flow estimation. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 2077–2086. IEEE, 2019.

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

[13] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International journal of computer vision*, 92(1):1–31, 2011.

[14] 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.

[15] Jonas Wulff, Daniel J Butler, Garrett B Stanley, and Michael J Black. Lessons and insights from creating a synthetic optical flow benchmark. In *European Conference on Computer Vision*, pages 168–177. Springer, 2012.

[16] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3061–3070, 2015.

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

[18] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4340–4349, 2016.

[19] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4040–4048, 2016.

[20] Daniel Kondermann, Rahul Nair, Katrin Honauer, Karsten Krispin, Jonas Andrulis, Alexander Brock, Burkhard Gussefeld, Mohsen Rahimimoghaddam, Sabine Hofmann, Claus Brenner, et al. The hci benchmark suite: Stereo and flow ground truth with uncertainties for urban autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 19–28, 2016.

[21] Toshio M Chin, William Clement Karl, and Alan S Willsky. Probabilistic and sequential computation of optical flow using temporal coherence. *IEEE Transactions on Image Processing*, 3(6):773–788, 1994.

[22] Ryan Kennedy and Camillo J Taylor. Optical flow with geometric occlusion estimation and fusion of multiple frames. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 364–377. Springer, 2015.

[23] 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.

[24] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European conference on computer vision (ECCV)*, pages 305–321, 2018.

[25] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[26] Boyuan Jiang, MengMeng Wang, Weihao Gan, Wei Wu, and Junjie Yan. Stm: Spatiotemporal and motion encoding for action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2000–2009, 2019.

[27] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.

[28] Jinrui Yang, Wei-Shi Zheng, Qize Yang, Ying-Cong Chen, and Qi Tian. Spatial-temporal graph convolutional network for video-based person re-identification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3289–3299, 2020.

[29] Jianing Li, Shiliang Zhang, and Tiejun Huang. Multi-scale 3d convolution network for video based person re-identification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8618–8625, 2019.

[30] Lin Sun, Kui Jia, Dit-Yan Yeung, and Bertram E Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4597–4605, 2015.

[31] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *proceedings of the IEEE International Conference on Computer Vision*, pages 5533–5541, 2017.

[32] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[33] 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.

[34] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[35] Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. Up-detr: Unsupervised pre-training for object detection with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1601–1610, 2021.

[36] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

[37] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8844–8854, 2022.

[38] Yihong Xu, Yutong Ban, Guillaume Delorme, Chuang Gan, Daniela Rus, and Xavier Alameda-Pineda. Transcenter: Transformers with dense queries for multiple-object tracking. *arXiv preprint arXiv:2103.15145*, 2021.

[39] Peize Sun, Jinkun Cao, Yi Jiang, Rufeng Zhang, Enze Xie, Zehuan Yuan, Changhu Wang, and Ping Luo. Transtrack: Multiple object tracking with transformer. *arXiv preprint arXiv:2012.15460*, 2020.

[40] Jieneng Chen, Yongyi Lu, Qihang Yu, Xiangde Luo, Ehsan Adeli, Yan Wang, Le Lu, Alan L Yuille, and Yuyin Zhou. Transunet: Transformers make strong encoders for medical image segmentation. *arXiv preprint arXiv:2102.04306*, 2021.

[41] Ali Hatamizadeh, Yucheng Tang, Vishwesh Nath, Dong Yang, Andriy Myronenko, Bennett Landman, Holger R Roth, and Daguang Xu. Unetr: Transformers for 3d medical image segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 574–584, 2022.

[42] Brendan Duke, Abdalla Ahmed, Christian Wolf, Parham Aarabi, and Graham W Taylor. Sstvos: Sparse spatiotemporal transformers for video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5912–5921, 2021.

[43] Zhaoyang Huang, Xiaoyu Shi, Chao Zhang, Qiang Wang, Ka Chun Cheung, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Flowformer: A transformer architecture for optical flow. *arXiv preprint arXiv:2203.16194*, 2022.

[44] Ao Luo, Fan Yang, Xin Li, and Shuaicheng Liu. Learning optical flow with kernel patch attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8906–8915, 2022.

[45] 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.