Institute for Visualization and Interactive Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Learning the Loss in Optical Flow Estimation based on the End-Point-Error

Jiaqi Zhao

Course of Study:	Autonome Systeme M.Sc
Examiner:	Prof. Dr.-Ing. Andrés Bruhn
Supervisor:	M.Sc. Azin Jahedi
Commenced:	November 2, 2022
Completed:	May 2, 2023

Abstract

Currently, deep neural networks are penetrating every corner of computer vision tasks, including optical flow estimation. Although supervised methods of optical flow estimation have facilitated more accurate motion prediction, the lack of labeled training data has hindered this progress. As a result, unsupervised and semi-supervised methods have attracted enormous interest in optical flow estimation. Contrastive learning serves as one of the primary bases of some semi-supervised methods. Despite the advances of combining contrastive learning and flow estimation by using the loss between the positive and negative prediction as the semi-supervised loss term, there are neither no full investigations on whether such a loss can be estimated nor a clear indication of the extent of the penalties on the negative samples.

The aim of this work is to investigate whether such a flow error can be estimated in a supervised fashion given two consecutive images and the corresponding estimated optical flow only. We proposed three architectures of error estimation networks and performed experiments on them, in which the update of parameters is supervised by the end-point error to the ground truth flow error during training. The ground truth flow error is the difference between the ground truth flow and the estimated flow. The evaluation results indicate that with a proper combination of error and flow estimation networks, the flow error can be estimated to some extent, especially on the synthetic dataset FlyingChairs2. Furthermore, we fine-tune the RAFT flow estimation networks with the validation samples of FlyingChairs2 by means of the error-based semi-supervised methods and improve the accuracy by approximately 4.7% in AEPE on the FlyingChairs2 dataset.

In summary, we would conclude that the RAFT-like and GMA-like error estimation networks are able to predict flow errors. Moreover, the estimated flow error can be utilized as a potential direction to improve the optical flow estimation in a semi-supervised manner.

Contents

1 Introduction and Related Works

1.1 Introduction

A dense pixel-wise correspondence between two successive frames of an image sequence can be described as an optical flow, in which the first frame specifies where each pixel is in the second frame, and the second frame specifies where each pixel will be. An apparent "flow" of pixels between the two images is represented by the resulting vector field of relative pixel locations. Optical flow is useful for estimating motion, disparity, and semantic correspondence, thus the improvement in the optical flow estimation benefits the performance of various downstream tasks such as action recognition[SLG+19], visual odometry[WCWT17], object tracking[ZUB18], semantic segmentation[REYE19], motion segmentation[MES+21], SLAM[TD21], etc.

Deep learning has recently led to dramatic improvements in the performance of optical flow techniques. Inspired by the pioneering work of applying the end-to-end trainable Convolutional Neural Network (CNN) approach for optical flow estimation proposed by Dosovitskiy *et al.* [DFI+15], extensive studies focusing on supervised methods (e.g.FlowNet2[IMS+17], PWC-Net[SYLK18], RAFT[TD20], GMA[JCL+21], FlowFormer[HSZ+22], etc.) have established increasingly improved results.

Given the significant role deep neural networks have played in optical flow estimation, successfully training flow estimation networks in a supervised manner requires a large amount of labeled data. Nevertheless, such supervised approaches are hand-crafted by the limited labeled training samples. It is challenging to collect such datasets for optical flow estimation problems, especially those that involve real-world images because currently there is no sensor that can measure the per-pixel ground truth motions for the entire image. Manual labeling is also not applicable. Therefore studies over the past decades have provided important information on the effectiveness of unsupervised and semi-supervised methods in optical flow estimation. Recently, surveys such as that conducted by Jeong *et al.* [JLPK22] have even achieved further improvements in the accuracy of the estimation.

Several of these approaches with alternative learning paradigms are based primarily on contrastive learning introduced by Hadsell *et al.* [HCL06]. Take Bailer *et al.* [BVS17] and Zhang *et al.* [ZJB+22] for example: Bailer *et al.* [BVS17] proposed to learn a meaningful feature descriptor with a modified hinge embedding loss such that the L2 distance between the matching patches is small while the L2 distance between the non-corresponding is large. CLIP-Flow[ZJB+22] also embedded the contrastive learning concept to learn a contrastive flow loss, which is desired to hence the training with positive features and thus results in a better representation of the optical flow. To date, this topic has also been studied in our group, as two students, Jahedi[Jah18] and Schäufele[Sch21], have submitted their theses on contrastive-learning-based optical flow estimation. Jahedi's[Jah18] analysis of more descriptive "features" provided a strong insight into the contrastive

learning idea in a supervised fashion. Schäufele[Sch21] carried out a number of investigations into the learning of a loss term which incentives the matching samples and penalizes the non-matching ones generated by the teacher models.

However, previous studies of contrastive-learning-based optical flow estimation have not treated supervised learning of such contrastive loss in much detail. Furthermore, few studies have dealt with quantitative metrics of such a loss between various negative samples. Take the negative samples of the optical flow in Figure 1.1 for example. The red arrow represents the ground truth flow of pixel (i, j) while the purple and green arrows refer to the non-matching flow A and B, respectively. In the spirit of contrastive learning, the losses of negative samples A and B should be maximized. Although negative sample A is not the perfect match for the reference ground truth flow, it is not so deviated from the ground truth flow and is still capable of providing meaningful guidance to the ground truth flow. There is a significant difference between negative sample B and the ground truth, which may cause the flow estimation network to incorporate inaccurate features and mislead the flow estimation.

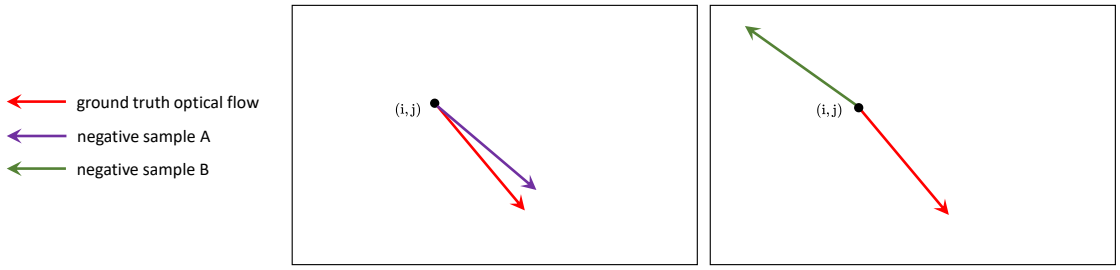


Figure 1.1: Illustration of two negative samples.

Therefore such a negative sample A should not be penalized by the loss term as much as the negative sample B in the right part of the figure. Thus a question arises: How can we quantitatively represent such a loss term and distinguish negative samples with different deviation extents? We decided to use the flow error, which is the difference between the ground truth flow and the negative sample, as the quantitative metric of the contrastive loss. Negative sample A has a significantly lower flow error than negative sample B, as is obvious without saying. Therefore, we assume that flow error can be used as a criterion to learn such a contrastive loss.

The aim of this work is to explore the feasibility of learning the flow error in a supervised manner. We propose various error estimation networks and they can produce the estimated flow error based on two consecutive input images and the corresponding predicted optical flow. The estimated optical flow is generated by some notable architectures, namely RAFT[TD20] and GMA[JCL+21]. Such learnable flow errors can be utilized to train the optical flow estimation networks in an unsupervised or semi-supervised fashion. Additionally, we conduct experiments relating to error-based semi-supervised optical flow estimation.

1.2 Related Works

To begin with, we provide a brief background on the contrastive learning method. Contrastive learning is first used as a powerful method of learning visual representations in a self-supervised manner. The basic concept of contrastive learning proposed by Hadsell *et al.* [HCL06] is: The representation of the positive (similar) samples should be mapped close together, while the negative (dissimilar) samples should be mapped away from the positive ones. In this way, it minimizes the distance between pairs of positive elements and maximizes the distance between pairs of negative elements. With the contrastive learning idea, the features or representations of the positive samples can be learned by contrasting the positive and negative samples.

With respect to optical flow estimation, we can observe that several attempts have been made to incorporate contrastive learning strategies into this process. Gadot and Wolf[GW16] proposed PatchMatch as the pioneer to adopt CNNs as feature extractors. In order to construct the training set, the method collects positive (matching) examples of corresponding patches with respect to ground truth flow. Following the contrastive learning concept, negative (non-matching) examples of non-matching patches obtained by shifting the image patches randomly around the ground truth flow direction are also collected. Intuitively, the flow estimation networks can extract more discriminative features from the contrastive of positive and negative samples. The network is trained with a modified DrLIM[HCL06] loss. It is possible to maximize the squared L2 distance between positive (matching) samples and minimize the distance between negative (non-matching) samples by employing such a training loss term.

In a similar vein, Bailer *et al.* [BVS17] established an approach to learn a meaningful feature descriptor with a modified hinge embedding loss. The hinge embedding loss is widely used for Siamese architectures to minimize the L2 feature distance between the matching patches and maximize the L2 feature distance between the non-matching patches above a certain margin m. To mitigate the architectural flaw of the hinge embedding loss, which will minimize the L2 distance between matching patches too aggressively, Bailer *et al.* proposed the variant of hinge embedding loss by adding a threshold t into the loss. This novel thresholded loss also inherits the idea of contrastive learning and improves the accuracy of the estimation flow.

In our group, Jahedi[Jah18] analyzed the effectiveness of learnable feature descriptors and embed them in the Coarse-to-Fine Patch Match (CPM)[HSL16] framework for optical flow estimation. The CNN-based learned descriptors are trained in a supervised manner following the contrastive learning idea, which incentives the matching patches and penalizes the non-matching ones.

Schäufele[Sch21], another student in our group, extended RAFT[TD20], which established SotA results at the time to the area of unsupervised learning. He carried out a number of investigations into the learning of a loss term in terms of the contrastive learning idea.

In 2022, Zhang *et al.* [ZJB+22] proposed CLIP-Flow to estimate the optical flow with an iterative pseudo-labeling framework in a semi-supervised method. An approach is proposed for improving representations for optical flow by explicitly training a network during the learning process, which is supervised by a semi-supervised contrastive flow loss in order to learn better features for optical flow.

1.3 Thesis Organization

The work has been organized in the following way: In Chapter 2, we provide an exhaustive explanation of optical flow and the optical flow datasets used in this thesis. Following these basic concepts, we also elaborate on the architectures of the well-celebrated flow estimation networks, RAFT[TD20] and GMA[JCL+21].

In Chapter 3, we first explain some crucial concepts in this work, including estimated flow error, ground truth error, and loss error. Then we propose three error estimation networks used in this work: namely the Single-iteration Conv, RAFT-like, and GMA-like error estimation networks. We provide an explanation of how these networks operate and move on to provide an elaborate introduction to our experiments on error estimation networks and report the results in Chapter 4.

In Chapter 5, we run semi-supervised optical flow estimation on the FlyingChairs2 dataset with the help of the error estimation networks. The conclusion of our thesis occurs in Chapter 6 where we summarize our work and provide an outlook on this thesis.

2 Foundations

A few notable flow estimation network architectures are discussed in this chapter, along with the basic principles of optical flow estimation. The concept of optical flow estimation is introduced in the first part of this chapter. In the following sections, we introduce the optical flow datasets that we used for the experiments in Chapter 4 and Chapter 5. There is a detailed explanation of the structures of RAFT[TD20] and GMA[JCL+21] in the final section of this chapter, as well as the optical flow estimator of PWC-Net[SYLK18].

2.1 Optical Flow

Estimating optical flow between two successive frames is a task that involves calculating pixel-wise displacement fields between two adjacent frames. Pixel motion is represented by such displacement fields, also called optical flow. Take two consecutive grey-scale images I_1 and I_2 shown in Figure 2.1 for example. Here, we want to estimate the flow of the pixel (x, y) (black point) in image I_1. It is assumed that the corresponding pixel in image I_1 fulfills the brightness constancy assumption. This means the intensities of the corresponding pixels are identical:

$$I_1(x, y) = I_2(x + u, y + v), \tag{2.1}$$

where $I_1(x, y)$ and $I_2(x + u, y + v)$ refer to the grey value of the pixel (x, y) in I_1 and the one of pixel $(x + u, y + v)$ in I_2. Thus we can confirm that pixel $(x + u, y + v)$ (red point) in the right image is the corresponding pixel of pixel (x, y). As such, the displacement between the corresponding pixels is (u, v) (blue vector). This displacement is the optical flow we want to estimate.

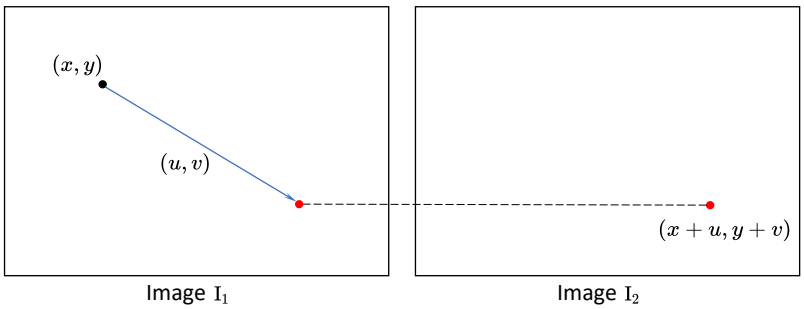


Figure 2.1: Illustration of optical flow of the pixel (x, y) between two consecutive frames.

2.2 Average End-Point Error (AEPE)

In the last section, the optical flow and the optical flow estimation problem are elaborated. So how can we evaluate the performance of an optical flow estimation network, or in other words, how is the quality of an estimated flow field evaluated? In this section, We introduce one of the widely used evaluation methodologies for optical flow estimation networks: Average End-Point Error (AEPE) Both methods focus on the flow error within metric spaces between a given ground truth flow field $F_{gt} \in \mathbb{R}^{2\times H\times W}$ and an estimated flow field $F \in \mathbb{R}^{2\times H\times W}$ generated by an optical flow estimation network. H and W represent the height and width of the flow field in pixels.

Before we explain the concept of Average End-Point Error (AEPE), let us first elaborate on the definition of End-Point Error (EE). Given the ground truth optical flow vector f_{gt} and an estimated optical flow vector f, the end-point error represents the Euclidean distance between the endpoints of two optical flow vectors. Intuitively, the total end-point error is averaged over the entire flow field so as to evaluate the performance of an optical flow estimation network with respect to the estimated optical flow field F and the corresponding ground truth flow field F_{gt}. The ground truth flow vector of the pixel (i, j) in the flow field is denoted as $f_{gt}(i, j) = (u_{gt}(i, j), v_{gt}(i, j))^\mathsf{T}$, while $f(i, j) = (u(i, j), v(i, j))^\mathsf{T}$ denotes the estimated optical flow vector of the same pixel. Consequently, the mathematical formulation of the average end-Point error yields as follows:

$$\begin{aligned} \mathrm{AEE}(F, F_{gt}) &= \frac{1}{H \times W} \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} \mathrm{EE}(f(i, j), f_{gt}(i, j)) \\ &= \frac{1}{H \times W} \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} \sqrt{(u(i, j) - u_{gt}(i, j))^2 + (v(i, j) - v_{gt}(i, j))^2} \end{aligned} \tag{2.2}$$

Public benchmarks also adopt other evaluation indicators. For instance, KITTI-2015 [MG15] benchmark evaluates the optical flow estimation network quantitatively by measuring the percentage of optical flow outliers over all ground truth pixels (F1-all). The F1-all (%) is a measure of the fraction of flow vectors with an end-point error exceeding more than three pixels or more than five percent of the ground truth flow. Mean Square Error(MSE) and Average Angular Error (AAE) are also common alternatives for flow estimation in particular applications.

In this thesis, we mainly utilize AEPE as the evaluation methodology for both error and flow estimation networks. The definition of the average end-point error of an estimated flow error field is introduced in Section 4.1.2.

2.3 Optical Flow Datasets

Considering the fact that deep neural networks have significantly influenced optical flow estimation, the most successful flow estimation networks all require a large amount of labeled data to directly train in a supervised fashion. In other words, we need to have many temporally consecutive image pairs with the corresponding flow map with per-pixel labels so as to train the networks based on standard supervised learning. However, the reality is cruel: Unlike other computer vision tasks like object recognition, segmentation, or stereo problems, collecting such datasets for optical flow estimation problems, especially those with real-world images, has been quite challenging due to the

lack of sensors at the time being that can directly measure the true per-pixel motions for the entire image. What's more, obtaining such datasets by manually labeling the samples is not applicable due to the inaccuracy of annotated motions and a huge amount of workload. This is why the real-world datasets for optical flow estimation problems are often constrained by quantities (e.g. the KITTI-2015 [MG15] dataset) or the constraint setup of samples (e.g. the VIPeR[RHK17] dataset).

To unfasten these handcuffs, researchers turn to another viable alternative: synthetic datasets, for example, the FlyingChairs[DFI+15] and the FlyingThings3D [MIH+16] datasets, which contain an abundance of synthesized images with ground truth flow information. Large-scale synthetic datasets are commonly used for pre-training optical flow estimation models (in which the training schedule FlyingChairs → FlyingThings3D is the most popular and effective one [IMS+17]), and then fine-tune them on the limited in-domain datasets, e.g. Sintel or KITTI-2015. Although this two-step training process surpasses the direct training on the limited target dataset, the networks inevitably suffer from domain discrepancy, and the performance is thus hindered. To narrow the domain gap, Huang *et al.* proposed Autoflow[HHH+22] which renders synthetic training samples to optimize the performance of the network on the target dataset. Autoflow's rendering system employs a layered approach, where each layer is controlled by a set of learnable hyperparameters to control its motion, shape, and appearance. Further studies focus on the dataset generation methods to enhance the quality of estimated optical flow on the target real-world samples. For example, RealFlow[HLL+22] is an Expectation-Maximization based framework that allows the direct production of large-scale optical flow datasets from arbitrary unlabeled realistic videos.

In this work, only the following five optical flow datasets are used for the training of error estimation networks in Chapter 4 and Chapter 5.

2.3.1 Sintel

The MPI-Sintel dataset[BWSB12], shorten as Sintel, is a synthetic dataset for optical flow evaluation that contains 1064 stereo image pairs and ground truth data for the disparity. Sintel samples are derived from an open-source animated 3D movie. Sintel is a more realistic dataset, including natural image degradations such as fog and motion blur. The entire dataset is divided into two parts, namely training split and test split. It comprises 1064 pixel-wise labeled training samples and 564 test samples, whose ground truth flow is not open to the public. Samples of Sintel are rendered with additional effect, so each sample has three versions: albedo pass, clean pass, and final pass. The albedo pass is the simplest version with no illumination effects; The clean pass is rendered with illumination effects while no post-processing effects are imposed on it; The most challenging version is the final pass, which is rendered with full effects such as motion blur and some atmospheric effects. In this work, we only use the clean and final pass of the Sintel training split for training and inference. Figure 2.2 shows some example samples of the Sintel dataset.

2.3.2 FlyingChairs and FlyingChairs2

On account of the limited samples of Sintel, Dosovitskiyz *et al.* [DFI+15] proposed a synthetic dataset by means of layering the static background image retrieved from Flickr (includes 964 images from categories "city", "mountain", and "landscape") with rendered CAD models of

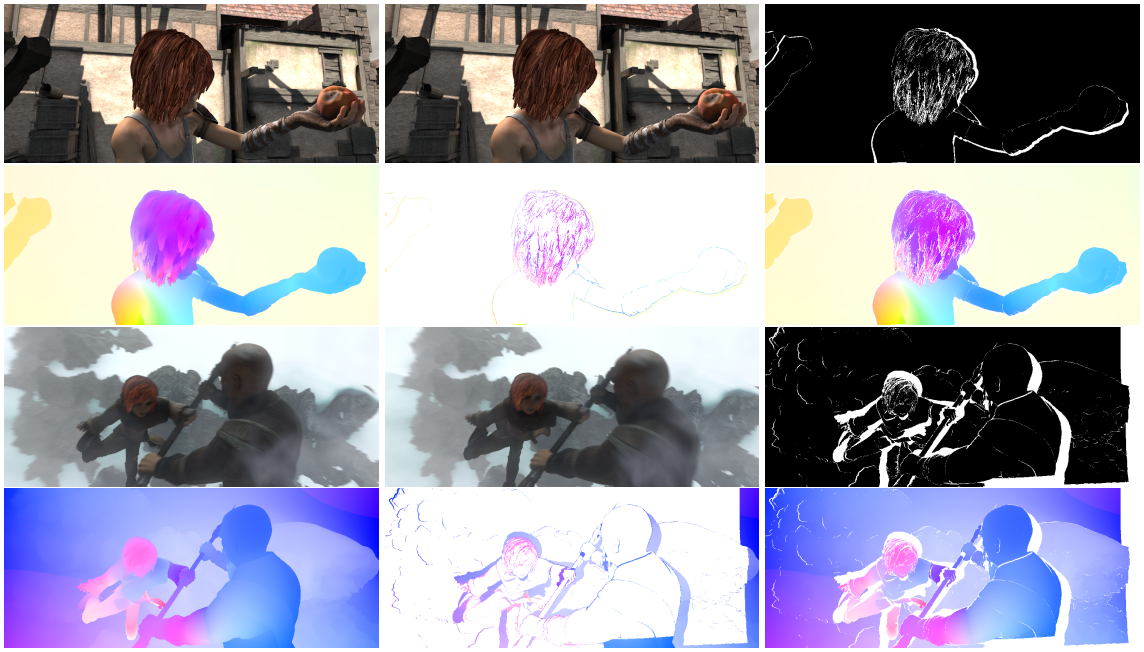


Figure 2.2: Examples of the Sintel dataset, occlusion map, and ground truth flow fields. Image source:[BWSB12]. The first two rows are an example of Sintel's clean pass. From left to right, on the upper row are the first and second input images as well as the occlusion map. On the lower row are the ground truth flow fields of all pixels, occluded and non-occluded pixels. The last two rows are an example of Sintel's final pass listed in the same order,

chairs[AME+14], whose parameterized affine motion imitates the motion of existing real-world samples. This random sampling of affine transformation for the background and the chairs generates the required motion. There exists a correlation between the 3D models' transformations and the background transformation. This can be depicted as the simultaneous movement of objects and the camera. Besides, all the other parameters regarding transformations including the initial position of chairs and the number of chairs that appeared in the image pairs are also randomly sampled in the fashion of a similar motion statistic of the Sintel dataset. Using this procedure, a dataset with 22,872 image pairs and ground truth flow fields was generated (22,232 training samples and 640 validation samples). To further investigate the motions in occluded and non-occluded regions separately, we adopt the FlyingChairs2 dataset[ISKB18] with additional provided occlusion maps in this work. Figure 2.3 shows an example sample of the FlyingChairs2 dataset. Despite the inevitable intrinsic differences with real-world samples, FlyingChairs and its varieties are utilized by researchers to pre-train their flow estimation networks before being fine-tuned on the target real-world datasets.

2.3.3 FlyingThings3D

Inspired by FlyingChairs[DFI+15], FlyingThings3D[MIH+16] provides an abundance of samples that are generated by randomly moving foreground objects above a background image. Here the foreground objects are daily objects, whose detailed 3D models are collected from the ShapeNet database[SCH15]. The samples of FlyingThings3D are randomly rendering the frames with different

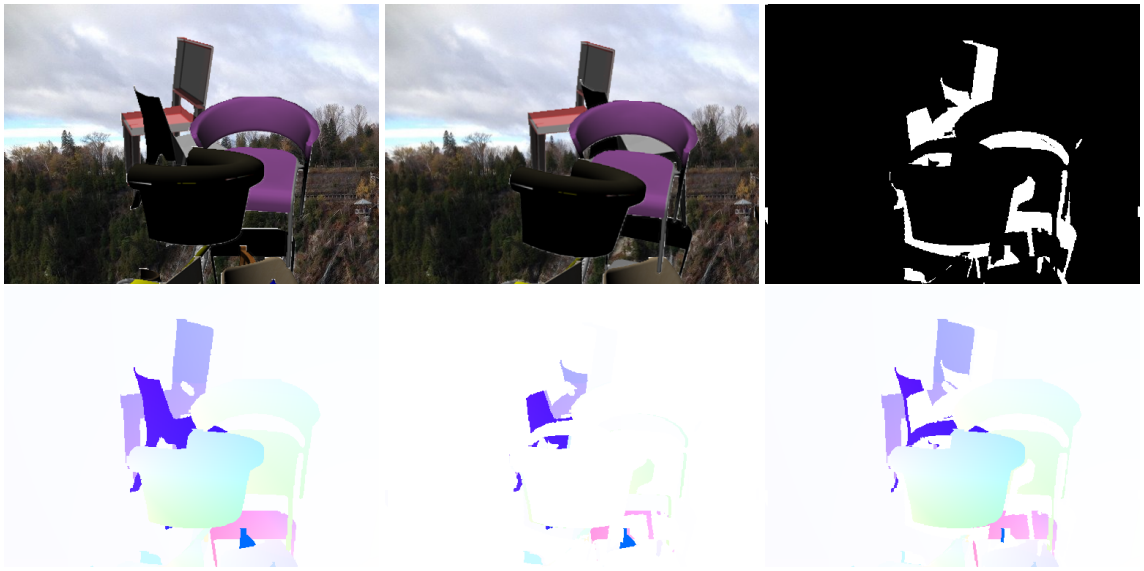


Figure 2.3: Examples of the FlyingChairs2 dataset, occlusion map, and ground truth flow fields. Image source:[ISKB18]. From left to right, on the upper row are the first and second input images as well as the occlusion map. On the lower row are the ground truth flow fields of all pixels, occluded and non-occluded pixels.

levels of realism. Similar to Sintel, each sample of FlyingThings3D has two versions. Lighting and shading effects can be observed in the clean pass images, while more noise and ambiguity are inserted into the final pass images with motion blur and defocus blur.

2.3.4 KITTI

KITTI-2012[GLU12] and KITTI-2015[MG15] are a collection of real-world optical flow samples with an application of autonomous driving. KITTI-2015 consists of dynamic scenes with moving cameras and moving objects while KITTI-2015 comprises static scenes with moving vehicles only. Thus, only the KITTI-2015 dataset is used in the following training and evaluation in this work, which can provide more realistic real-world samples. The term “KITTI“ in the following chapters refers to KITTI-2015. KITTI-2015 contains 200 training samples, along with test samples of the same amount, where the ground truth flow of the test samples is not revealed by Menze and Geiger[MG15]. The scenes of the datasets are captured by the high-resolution cameras on the vehicle and a 3D laser scanner. The laser scanner provides accurate depth and motion information for a subset of 3D points in the scene. The ground truth optical flow is collected through the projection of these 3D points in the image plane. As such, the pixel-wise motions can be computed between the consecutive images. However, the motion information of occluded and reflective points, or those with a large displacement can not be captured by the laser scanner. It only provides sparse data within a constrained margin of distance and height. Consequently, KITTI-2015 only provides a sparse ground truth flow, which comprises around 5% of the pixels in the images.

2.3.5 HD1K

HD1K[KNH+16] is also an optical flow dataset used in autonomous driving, which contains 1018 frames with a variety of lighting and weather scenarios and their corresponding ground truth data. Besides the occlusion mask is also provided. Similar to KITTI[MG15], the frames are collected by a stereo camera on top of the vehicle. It comprises 1018 frames with diverse lighting and weather scenarios, which involves the public traffic scenarios in urban and rural areas.

2.4 Architectures of the well-known Optical Flow Estimation Networks

A detailed explanation of the structure of Recurrent All-Pairs Field Transforms (RAFT) [TD20] and GMA[JCL+21] is presented in this section, together with a depiction of the flow estimator adopted by PWC-Net[SYLK18]. The architectures described below are also crucial to the error estimation networks discussed in the following chapter.

2.4.1 PWC-net

Constructing the feature pyramid with CNNs, PWC-Net[SYLK18] estimates optical flow in a coarse-to-fine way with six pyramid levels, with feature channels of 16,32,64,96,128, and 196. A cost volume is constructed with the feature map extracted from the source frame and the warped one of the target frame at each level. The optical flow estimator at the current level takes in the cost volume and the feature map of the first image, along with the upsampled estimated flow from the coarser level, and outputs the estimated flow at the current level. This estimator is a subsequent CNN module consisting of five convolutional layers with feature channels of 128, 128, 96, 64, and 32, respectively. Every convolutional layer of the optical flow estimator has two inputs: the output of its previous layer and the input of the layer that comes before it.

2.4.2 RAFT

RAFT [TD20] is an influential optical flow estimation framework and has demonstrated notable improvement at the time. Figure 2.4 presents a schematic overview of the RAFT working pipeline. First, the feature encoders which extract the feature maps from the input images, and the context encoder, which shares the same architecture as the feature ones, extracts the context feature only from the first frame; The matching block constructs an all-pairs correlation pyramid from the feature maps, which is denoted as 4D cost volume. Another core module of RAFT is the recurrent GRU-based iterative update operator, which can refine the estimated optical flow iteratively with the repeated look-up operation. The look-up cost can be retrieved from the correlation pyramid and updated in each GRU iteration by indexing the refined flow in the 4D cost volume. Then the motion encoder generates the motion feature of the current GRU iteration. Given the updated estimated flow, the motion feature, and the constant context feature map, the following GRU operator is capable of computing the flow residual. Therefore, the optical flow is refined. RAFT also proposed a convex upsampler to upsample the final refined flow at 1/8 resolution to the resolution of input frames.

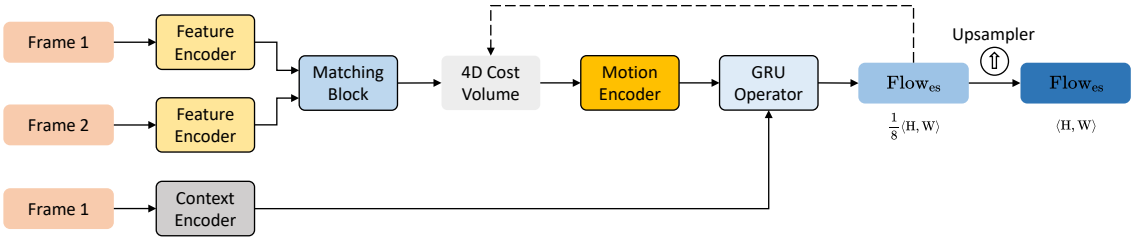


Figure 2.4: Schematic overview of the RAFT flow Estimation network pipeline. Image adapted from [TD20].

Feature and Context Encoders

In this section, we focus on the architecture of the feature and context encoders proposed by RAFT[TD20]. Both the images I_1 and I_2 will be applied with separate feature encoders which embrace their own learnable parameters. Only the first image I_1 will pass through the context encoder to produce the required context features for further flow error estimation.

Both encoders share the same architecture. Take the feature extraction network for example. It can be distilled down to three parts: (1) preprocessing block, (2) residual blocks, and (3) output block, whose structure is illustrated in Figure 2.5. First, the input image $I \in \mathbb{R}^{3\times H\times W}$ will be passed through a 7×7 convolutional layer, followed by a normalization layer and an output layer with ReLU activation function (denoted as "LayerNorm" in the grey background). This is the preprocessing block that outputs a feature map of $\mathbb{R}^{64\times H\times W}$. Then the feature map is further passed through 6 residual blocks with 3 expanding feature channels at 3 decreasing resolutions: 2 residual blocks (denoted as "Residual Block" in blue background) at $1/2$ resolution with 64 feature channels, followed by another 2 residual blocks (denoted as "Residual Block" in blue background) at $1/4$ resolution with 128 feature channels and finally the last 2 residual blocks (denoted as "Residual Block" in the red background) at $1/8$ resolution with 196 feature channels. The output block, which is a 3×3 convolutional layer, takes in the output feature map of the residual blocks and outputs a feature map at $1/8$ resolution with 256 feature channels. The difference between the feature and context encoders lies in the normalization layers: Instance normalization is employed by the feature encoder, while batch normalization is employed by the context encoder.

Matching Block, All-pairs Correlation Cost Volume and Look-up Cost

Teed and Deng[TD20] proposed an all-pairs correlation cost volume to model correlations for all possible displacements. The inputs of the matching block are the feature maps extracted by the feature encoders from the image pair. Features perform a dot product and thus the matching correlations across each level are constructed. These are called correlation volumes. By means of pooling and stride, all the correlation volumes are constructed together as a correlation pyramid, which is also referred to as 4D cost volume. This correlation pyramid represents the visual similarity of the image pairs and provides crucial information about the motion with various displacements.

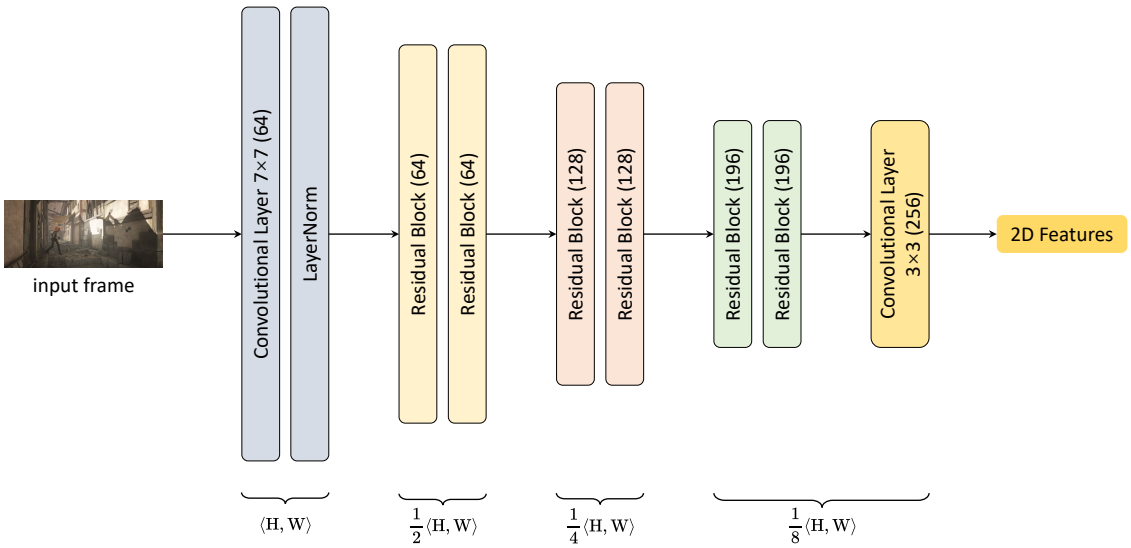


Figure 2.5: The Architecture of Feature and Context Encoders.

Furthermore, a look-up operator is introduced to retrieve the required feature map (look-up cost) from the correlation pyramid by indexing the displacement of each pixel in each level of the 4D cost volume. The correlation pyramid remains constant over the estimation while the look-up cost is updated in each GRU iteration.

Iterative Updates

The success of RAFT also largely lies in a large number of iterative refinements. With the Gated Recurrent Unit (GRU) based iterative update operator adopted by RAFT, a sequence of predicted flows is generated and refined over each GRU iteration. The inputs of the iterative update operator are the estimated flow from the last GRU iteration and the all-pairs correlation pyramid constructed from the feature maps at 1/8 resolution. The look-up cost can be retrieved from the correlation pyramid with the estimated flow and processed by the motion encoder which outputs the motion features. Given the concatenation of the look-up cost, the context features, and the estimated flow, the GRU operator can predict the flow residual of the current GRU iteration. Then the flow residual is added to the predicted flow from the last GRU iteration. Thus the flow is refined iteratively. Both flow and flow residual in the iterative update operator are at 1/8 resolution. The hidden state of the GRU operator is updated as well, which is passed through two convolutional layers and predicts a convex upsampling mask. Thus the final refined estimated flow can be upsampled to full resolution.

Convex Upsampling

The GRU-based iterative update operator outputs an estimated optical flow at a lower resolution. The estimated flow at low resolution is then upsampled to the original resolution of the input images with the proposed convex upsampling module. So how does the convex upsampling, or to be more

precise, the weighted combination work? The upsampling operation computes the target resolution flow of each pixel as the weighted average of the flow at the neighboring pixels in a 3×3 grid. In this context, neighboring pixels refer to pixels at $1/8$ resolution rather than pixels at the fine resolution. With the $H/8 \times W/8 \times (8 \times 8 \times 9)$ upsampling mask, which is also the output of the flow error estimation blocks, the weights of the neighboring pixels (nine in total) are performed with Softmax. Figure 2.6 illustrates the convex upsampling module. We also adopt this convex upsampling instead of the conventional adopted bilinear upsampling method in the error estimation networks proposed in Chapter 3 to enhance the accuracy of the estimated flow error at motion boundaries.

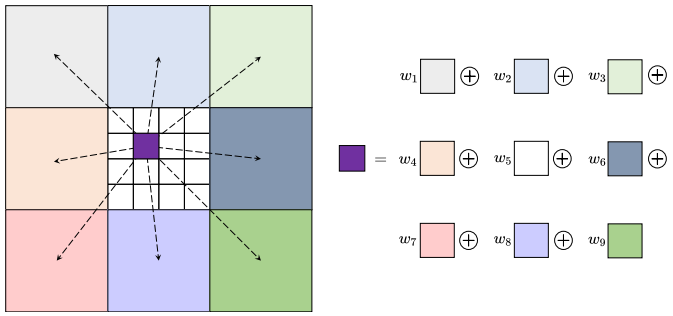


Figure 2.6: Illusiation of the convex upsampling module. Image source:[TD20].

2.4.3 GMA

The GMA[JCL+21] flow estimation network follows a similar framework to RAFT[TD20] except for a Global Aggregation Motion (GMA) module. The purpose of this GMA module is to combine the long-range image self-similarities with the corresponding motion feature with the help of the self-attention mechanism. Such a GMA module can boost the performance of the network especially in occluded regions. Figure 2.7 illustrates the working pipeline of GMA. Architectures of the other components in GMA[JCL+21] inherit those of RAFT[TD20]. The feature maps are extracted by the feature encoder from two images and a 4D cost volume is constructed based on the feature maps. In each GRU iteration, the GMA module takes in the updated motion feature and the context feature map and generates the aggregated motion feature. Then the concatenation features, including the motion feature and the aggregated motion feature as well as the context feature, pass through the GRU operator to compute the residual flow of the current GRU iteration. Thus the estimated optical flow is iteratively refined. Finally, the refined optical flow is upsampled to the target resolution via the convex upsampling mask.

Global Motion Aggregation (GMA) module

Taking inspiration from the Self-Attention mechanism proposed by Vaswani *et al.* [VSP+17], GMA designed a Global Motion Aggregation (GMA) module to model the long-range relations between the local features of the input frames. The details of the GMA module are presented in Figure 2.8. Same as RAFT[TD20], the motion feature is updated in each GRU iteration. Given the updated input motion feature matrix $\mathbf{y}$ and the constant context feature map $\mathbf{x}$ extracted by the context encoder, the query vector and key vector are derived from the context feature map and the value vector is derived from the motion feature. These vectors are all computed through linear projection, such that

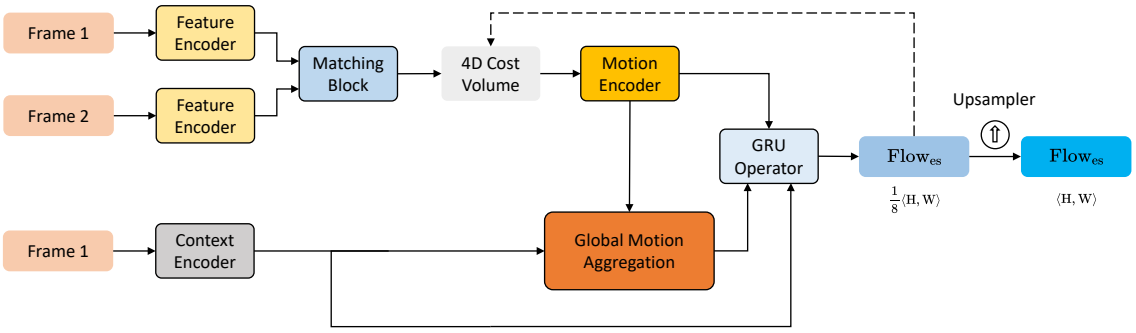


Figure 2.7: Schematic overview of the GMA flow Estimation network pipeline. Image adapted from [JCL+21].

$\mathbf{Q} = \mathbf{W}_{\text{qry}}\mathbf{x}$, $\mathbf{K} = \mathbf{W}_{\text{key}}\mathbf{x}$, $\mathbf{V} = \mathbf{W}_{\text{val}}\mathbf{y}$. The final output of the GMA module is the aggregated motion feature, which is an attention-weighted sum of the projected motion features. The aggregated motion feature can be formulated as follows:

$$\hat{\mathbf{y}} = \mathbf{y} + \alpha \text{Attention}\left(\mathbf{K}, \mathbf{Q}, \mathbf{V}\right), \tag{2.3}$$

where α is a learnable coefficient and $\text{Attention}\left(\mathbf{K}, \mathbf{Q}, \mathbf{V}\right)$ is the attention map:

$$\text{Attention}\left(\mathbf{K}, \mathbf{Q}, \mathbf{V}\right) = \mathcal{A}\left(\mathbf{K}, \mathbf{Q}\right)\mathbf{V}. \tag{2.4}$$

$\mathcal{A}\left(\mathbf{K}, \mathbf{Q}\right)$ is the scaled-dot product of query and key vector aiming to compute the self-similarities of the context feature, where is a dot-product attention function proposed by Vaswani *et al.* [VSP+17]

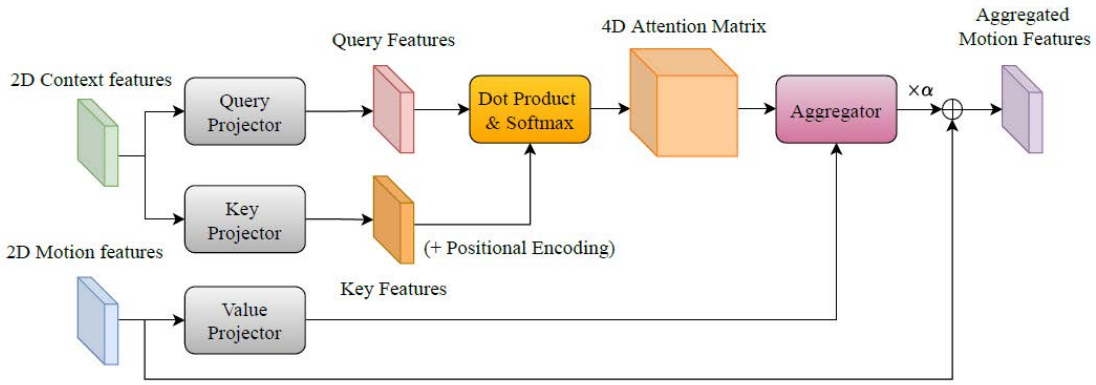


Figure 2.8: Schematic details of the Global Motion Aggregation (GMA) module. Image source: [JCL+21].

2.4.4 Chapter Conclusion

In this chapter, we introduce the fundamental knowledge of optical flow estimation and several widely used optical flow dataset. Additionally, a detailed explanation of the architectures of RAFT[TD20] and GMA[JCL+21] and the flow estimator of PWC-Net[SYLK18] is also provided. In the next chapter, three different architectures of error estimation networks are proposed in the spirit of the above-mentioned well-established flow estimation networks.

3 Error Estimation Networks

In our previous chapters, we have introduced the necessary foundations to this work as well as presented the related architectures of several well-known architectures. In this chapter, we propose three different error estimation network architectures based on the well-established architectures of RAFT[TD20] and GMA[JCL+21]. Two consecutive images are inputs for the error estimation network, with an additional estimated flow, which is estimated by a flow estimation network whose inputs are identical image pairs. Figure 3.1 demonstrates the basic idea of the proposed error estimation networks. Prior to presenting the architectures of the error estimation networks, it is necessary to explain the key concepts and definitions underlying this work: *ground truth flow error*, *estimated flow error*, and *loss error*. Then the architectures of the Single-iteration Conv, RAFT-like, and GMA-like error estimation networks are illuminated in the following sections.

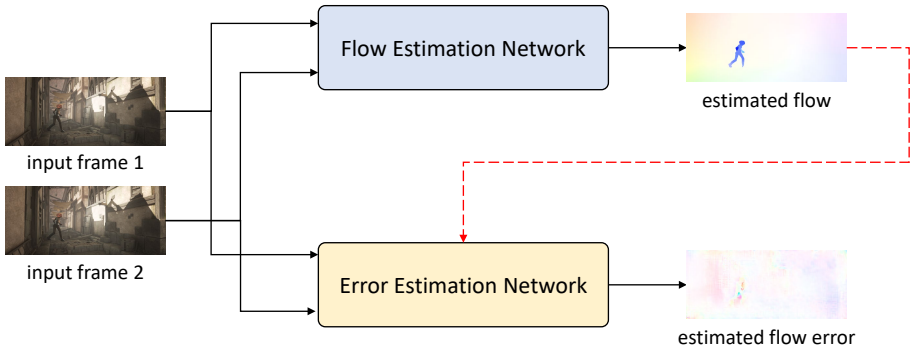


Figure 3.1: The basic idea of the error estimation network.

3.1 Error Ground truth, Estimated Error, and Loss Error

It is crucial for us to elucidate some new concepts and definitions which are crucial to our error estimation networks before we further discuss what error estimation networks are and how they work. These concepts and definitions include *ground truth flow error*, *estimated flow error*, and *loss error*. These definitions and notations are essential for us to explain the architecture and how we supervised the training of the error estimation networks.

Our definition of *flow error* follows the traditional definition, which is the difference between Flow_{es} and Flow_{gt}, as shown in Figure 3.2, or in the mathematical description, it is Flow Error = $\text{Flow}_{\text{gt}} - \text{Flow}_{\text{es}}$. The flow estimation network estimates the flow Flow_{es} given two consecutive images I_1 and I_2, while the accurate ground-truth flow Flow_{gt} is provided by the optical flow datasets themselves.

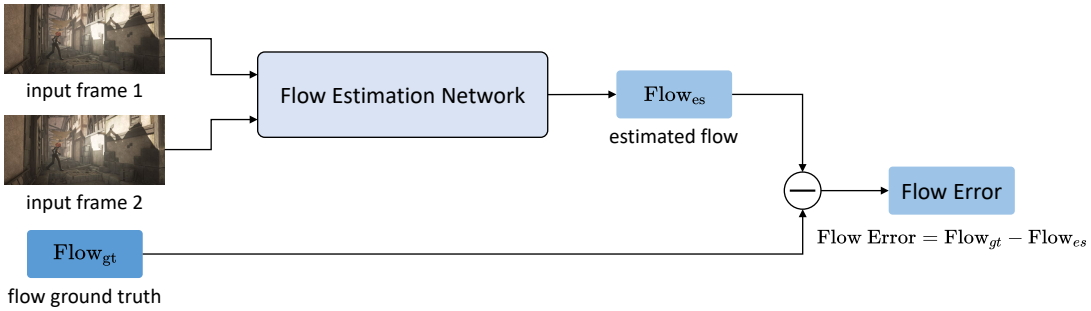


Figure 3.2: The definition of flow error

We would like to introduce a relatively new concept, *loss error*, in contrast to the traditionally defined *flow error*. We need to estimate the flow error Error_{es} via the proposed error estimation network, given two temporally consecutive image pairs I_1 and I_2 with an estimated flow Flow_{es}, which is produced by an arbitrary flow estimation network. The inputs of this flow estimation network are still the same image pairs I_1 and I_2 mentioned above. In order to supervise the training processes as well as evaluate the resulting tuned error estimation networks, we initiate the training methods of the optical flow estimation networks which are applied by RAFT [TD20], GMA [JCL+21] ,and many other celebrated flow estimation models. To put it another way, a error estimation network is trained by minimizing the pixel-wise Euclidean distance between the ground truth flow error and flow error estimation based on the entire flow error map with per-pixel labels. The *ground truth flow error*, denoted as Error_{gt}, refers to the difference between the estimated flow error and the ground truth flow error, as shown in Figure 3.3. Moreover, the loss of the estimated flow error, or loss error, is denoted as *loss error*, which is the difference between the ground-truth flow error and flow error estimation Error_{es}. A diagram illustrating the explanation of these concepts can be found in Figure 3.3.

3.2 Error Estimation Network Overview

Given two temporally consecutive images, I_1 and I_2, as well as an estimated flow Flow_{es} which is predicted by the selected flow estimation network given the same image pairs, the error estimation network is designed to generate a per-pixel flow error estimation Error_{es}.

In the following proposed error estimation networks, the feature encoders and context encoders inherit the flow estimation pipeline of RAFT [TD20]. Furthermore, the matching block, which is responsible for constructing the all-pairs correlation cost volume, also encapsulates the same idea as the one proposed by [TD20].

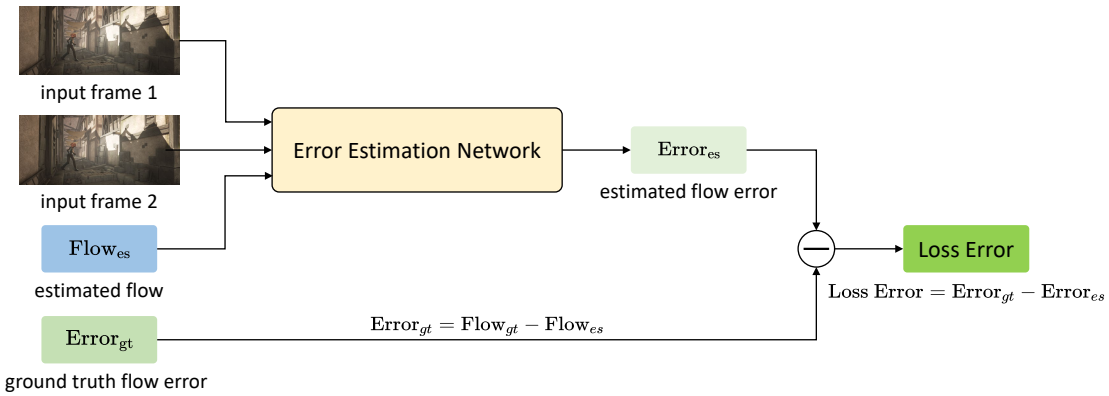


Figure 3.3: The definition of ground truth flow error and loss error.

All the flow error estimation blocks of the three error estimation networks output an estimated flow error at $1/8$ resolution. The estimated flow error at low resolution is then upsampled to the original resolution of the input images with the convex upsampling module. As such, we also adopt the convex upsampler to upsample the estimated flow error from the lower resolution to the target resolution. The convex upsampler here shares the same architecture as the one of RAFT [TD20].

3.2.1 Inputs of the Error Estimation Network

The inputs of error estimation networks are two consecutive RGB images $\mathrm{I}_1, \mathrm{I}_2 \in \mathbb{R}^{3\times H\times W}$, where H and W refer to the heights and widths of the corresponding images. Additionally, an estimated flow $\mathrm{Error}_{\mathrm{es}} \in \mathbb{R}^{2\times H/8\times W/8}$ is also obtained by the selected flow estimation network. We embrace the notable RAFT [TD20] and GMA [JCL+21] as the flow estimation network to produce the required estimated flow (Please refer to Section 4.2.2 for further explanation of the choice of these two notable flow estimation networks). On account of adopting the promising combination of feature encoders and visual similarity with the same architecture as RAFT and GMA, as well as the convex upsampling module, we utilize the estimated flow $\mathrm{Flow}_{\mathrm{es}}$ at 1/8 resolution instead of the upsampled estimated flow at full resolution as the input of error estimation networks. With the estimated flow $\mathrm{Flow}_{\mathrm{es}}$ at 1/8 resolution the error estimation network can spare a downsampling module to downsample the full-resolution estimated flow. This can avoid a bloated network and the potential information loss during the upsampling-downsampling operation. However, the estimated optical flow at full resolution is also needed to generate the ground truth flow error for the training of error estimation networks in Chapter 4. Further details about feature and context encoders as well as computing visual similarity can be found in Section 2.4.2 and Section 2.4.2.

3.3 Single-iteration Conv Error Estimation Network

First, we introduce an error estimation network with a relatively straightforward structure. Figure 3.4 depicts the design of the Single-iteration Conv error estimation network. The idea of the Single-iteration Conv error estimation network is to estimate the flow error without the iterative update operator and investigate whether the flow error can be estimated only with pure convolutional layers as the estimator part.

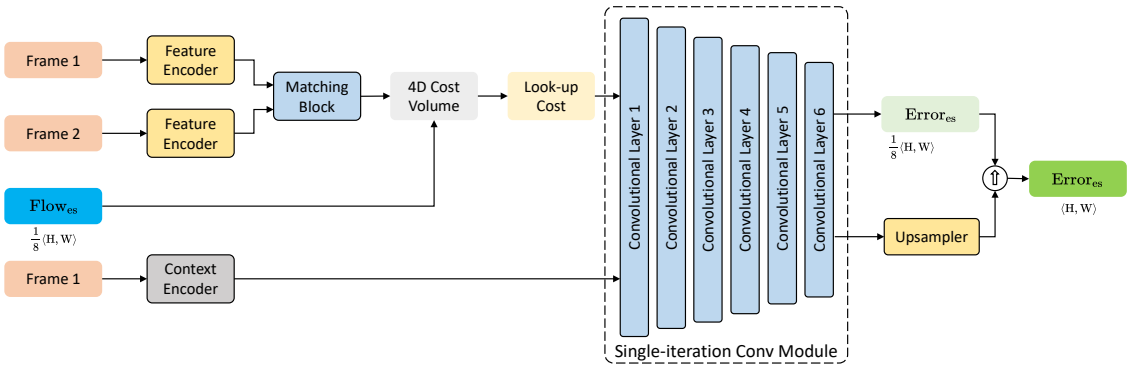


Figure 3.4: Architecture of the Single-iteration Conv Error Estimation Network.

The feature and context encoders are reserved, along with the correlation pyramid and the upsampler proposed by RAFT [TD20]. Inspired by the "Optical Flow Estimator" of PWC-Net [SYLK18] flow estimation network, the flow error estimator of the Single-iteration Conv error estimation network, denoted as "Single-iteration Conv Module" in Figure 3.4, takes two feature maps, namely the look-up cost and context features of the first image as inputs. However, as we adopt the feature and context encoder introduced above in Section 2.4.2, our flow error estimator discards the coarse-to-fine method of estimation with several pyramid levels, we only keep the pyramid level at $1/8$ resolution of the input images with 256 feature channels and estimate the flow error at this resolution and upsample the flow error to the original resolution. The look-up cost is retrieved from the 4D cost volume given the estimated flow Flow_{es} alone. The context features maps are generated by the context encoder. The Single-iteration Conv module consists of six convolutional layers with feature channels of respectively 256, 256, 128, 96, 64, and 32, and outputs the estimated flow error Error_{es} at $1/8$ resolution of the input images as well as the convex upsampling mask, denoted as "Upsampler" in Figure 3.4. Same as PWC-Net, these convolutional layers are also enhanced with DenseNet connections [HLVW17], which makes the inputs to every convolutional layer the output of and the input to its previous layer. Finally, the predicted flow error is upsampled to the desired resolution with the convex upsampling mask. Please refer to Section 3.2 for more information on the feature and context encoder, the matching block as well as the upsampler adopted by the Single-iteration Conv error estimation network.

3.4 RAFT-like Error Estimation Network

Inspired by the RAFT[TD20] flow estimation networks, the architecture of the RAFT-like error estimation network embeds an all-pairs correlation pyramid and a recurrent GRU-based iterative update operator that retrieves the look-up cost from the correlation pyramid and refines the estimated

flow error in each GRU iteration. The major differences between the RAFT-like error estimation network and the Single-iteration Conv one lie in the error encoder and the iterative update operator, whose architectures and operation pipelines are explained in the following subsections. The architecture of the RAFT-like error estimation network is presented in Figure 3.5.

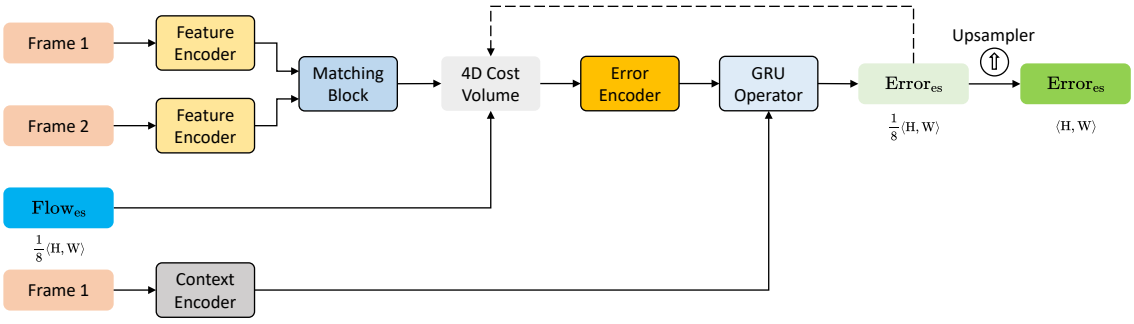


Figure 3.5: Schematic overview of the RAFT-like Error Estimation Network pipeline.

3.4.1 Error Encoder

The error encoder contains three convolutional blocks. Given the estimated flow error $\text{Error}_{\text{es}_{i-1}}$ of the current GRU iteration i (more details of the iterative update operator can be found in Section 3.4.2), the look-up cost will be retrieved from the 4D correlation volume by the estimated flow with offset, namely the sum of the estimated flow Flow_{es} and the estimated flow error $\text{Error}_{\text{es}_{i-1}}$ of last GRU iteration $i-1$. Then a convolutional block, which consists of two convolutional layers, is applied to the indexed correlation features, and the estimated flow error $\text{Error}_{\text{es}_{i-1}}$ itself is also processed by another convolutional block, which shares the same structure as the first convolutional block. Finally, the output of these two convolutional blocks is concatenated together and passed through an additional convolutional block, whose output is called error features. This feature map will be updated on every iteration. The operation principle of the error encoder is illustrated in Figure 3.6.

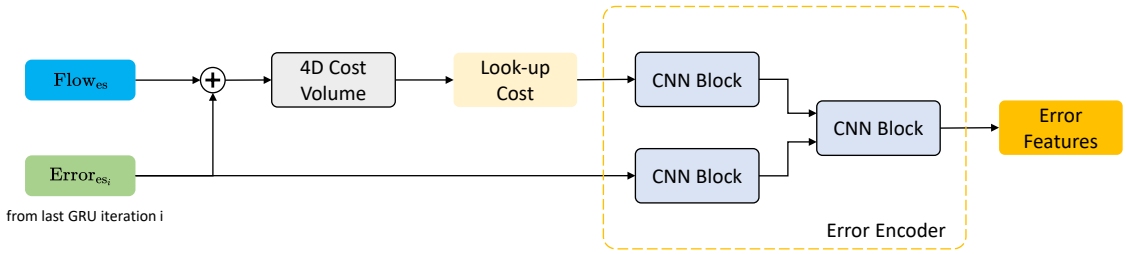


Figure 3.6: Illustration of the error encoder pipeline.

3.4.2 Iterative Update Operator

We also mimic the iterative update operator proposed by RAFT[TD20]. Our GRU-based iterative update operator is designed to compute a sequence of flow error estimates. Figure 3.7 illustrates how our iterative update operator helps the RAFT-like error estimation network predict the desired flow error. To distinguish the GRU operator iterations and training iterations, we call them *GRU iterations* and *iterations* respectively.

With each GRU iteration, the hidden state of the GRU operator is updated based on the iteration inputs of the static context information, the currently estimated flow field, and the result of the correlation look-up cost for the currently estimated flow error field. Let us take the GRU iteration i as an example to interpret the iterative estimation of the flow error. Given a flow estimation at $1/8$ resolution produced by a specified flow estimation network, denoted as Flow_{es}, and the estimated flow error of the last GRU iteration $\text{Error}_{\text{es}_{i-1}}$ at the same resolution, the correlated lookup feature map of the current GRU iteration, denoted as "Look-up Cost_i", can be retrieved from the constructed 4D cost volumes by the compensated flow, which equals the sum of Flow_{es} and $\text{Error}_{\text{es}_{i-1}}$. Taking in Look-up Cost_i and $\text{Error}_{\text{es}_{i-1}}$, the error encoder outputs "Error Features_i", which denotes the error feature map of the current GRU iteration. Then the outputted error feature map is concatenated with $\text{Error}_{\text{es}_{i-1}}$ as well as the directly injected context features. The GRU operator shares the same architecture as the one proposed by RAFT[TD20]. After taking in the concatenation of error and context feature maps as well as the estimated flow error at the last GRU iteration, the GRU operator will produce an updated hidden state of the GRU operator as well as the flow error residual at GRU iteration i, denoted as "$\triangle\text{Error}_i$". Then the flow error residual $\triangle\text{Error}_i$ is applied to the current estimated flow error $\text{Error}_{\text{es}_i}$. In other words, $\text{Error}_{\text{es}_i} = \triangle\text{Error}_i + \text{Error}_{\text{es}_{i-1}}$. The resulting flow error is then used as the initialization of the next GRU iteration. During training, the estimated flow error $\text{Error}_{\text{es}_i}$ is then upsampled to the estimated flow error at full resolution with the help of the convex upsampling mask mentioned in Section 2.4.2, which is the output of two convolutional layers taking the updated hidden state of the GRU operator of current GRU iteration as input. These estimated flow errors at full resolution of each GRU iteration are essential to the supervision of training (Please refer to Section 4.2.1 for more details of the loss function during training). When the error estimation network is in the inference mode, only the outputted flow error estimation $\text{Error}_{\text{es}_\text{N}}$ at the GRU iteration N, which is the sum over all flow error residual produced by the GRU operator over the N iterations, is the desired estimated flow error Error_{es} at $1/8$ resolution: $\text{Error}_{\text{es}} = \text{Error}_{\text{es}_0} + \sum_{i=1}^{\text{N}} \triangle\text{Error}_i = \sum_{i=1}^{\text{N}} \triangle\text{Error}_i$. Then the estimated flow error Error_{es} is upsampled to the full resolution with the convex upsampler of the last GRU iteration N. N is the total iterations of the iterative update operator, which will be assigned different numbers during training and inference. As initialization, we assume $\text{Error}_{\text{es}_0}$ to be 0, which means that at the first GRU iteration, the look-up cost is only retrieved by Flow_{es}.

3.5 GMA-like Error Estimation Network

We propose another architecture of the network, namely the GMA-like error estimation network, based on the notable flow estimation network GMA[JCL+21]. In comparison with the RAFT-like estimation network, we insert a Global Error Features Aggregation module in the network to globally aggregate the error features and input the concentrated features to the GRU-based iterative update operator instead of only the constant context features and the refined error features. Inspired by

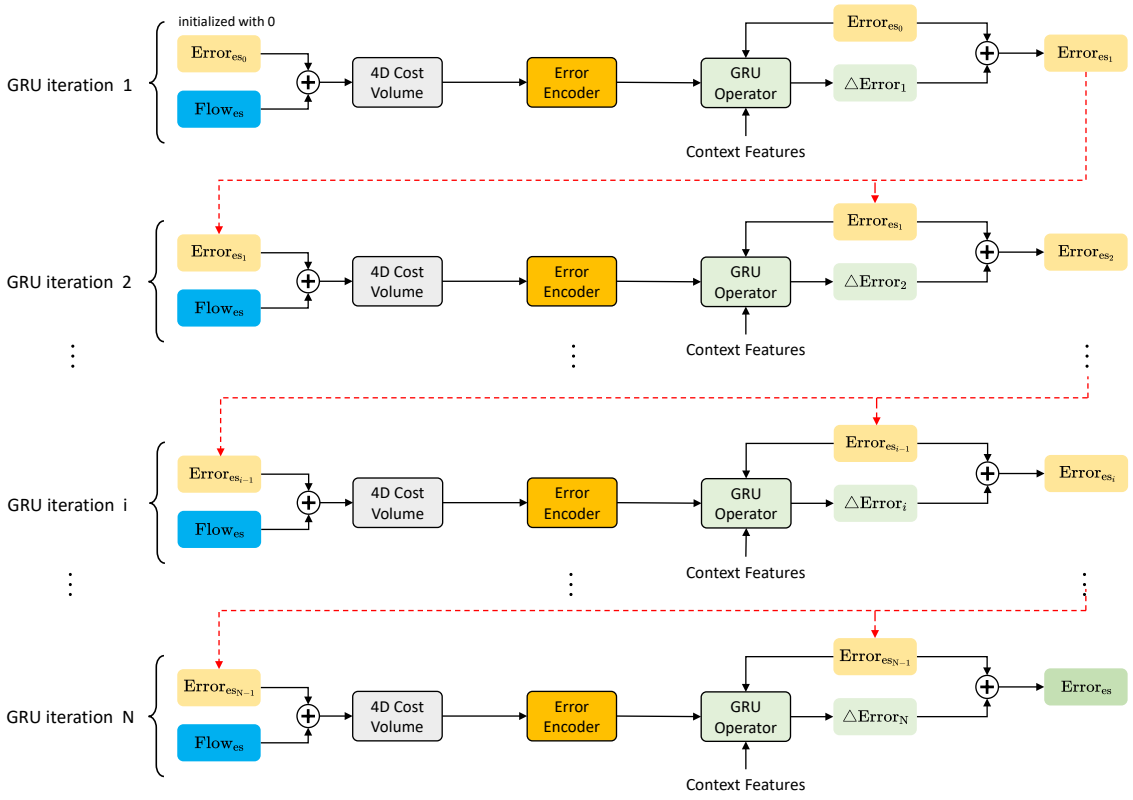


Figure 3.7: Illustration showing the iterative update operator. Image adapted from [TD20].

the aggregation mechanisms introduced by Jiang *et al.* , the purpose of proposing such an error estimation network is to improve the flow error predictions for occluded regions. Figure 3.8 presents the architecture of the GMA-like error estimation network.

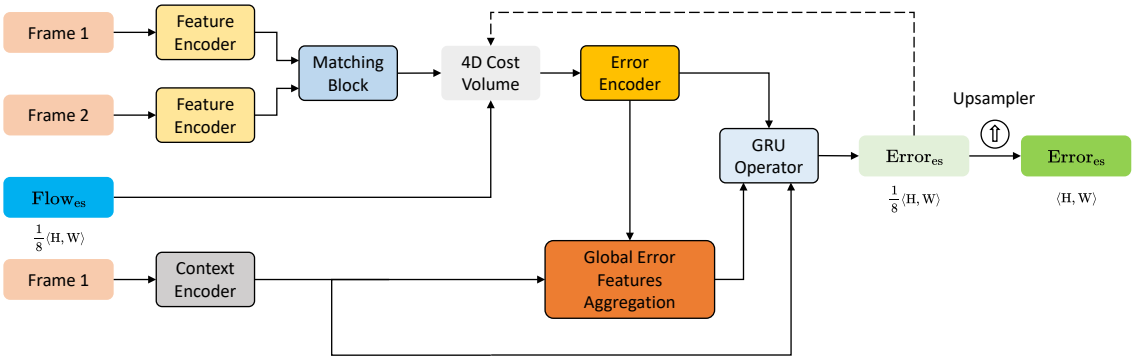


Figure 3.8: Schematic overview of the GMA-like Error Estimation Network pipeline.

3.5.1 Global Error Features Aggregation

Inspired by the self-attention mechanism which has become increasingly popular in computer vision in recent years, we initiate the Global Motion Aggregation (GMA) module [JCL+21] and propose our Global Error Features Aggregation module in the GMA-like error estimation network. The details of the Global Error Features Aggregation model are shown in Figure 3.9.

The self-attention mechanism proposed by Vaswani *et al.* [VSP+17] implies that the feature matrix is linearly projected to query, key, and value. This means that the difference between these output matrices only lies in the corresponding weight matrices $\mathbf{W}_{\text{qry}}$,$\mathbf{W}_{\text{key}}$, and $\mathbf{W}_{\text{val}}$. Nevertheless, our Global Error Features Aggregation model operates on two input matrices - i.e. the context feature matrix $\mathbf{x} \in \mathbb{R}^{N_\text{c} \times D_\text{c}}$ and the error feature matrix $\mathbf{y} \in \mathbb{R}^{N_\text{e} \times D_\text{e}}$, whose comprehensive introductions can be found in Section 2.4.2 and Section 3.4.1, respectively. Take N_c as an example: N_c represents the number of tokens of the context feature matrix, in other words, $N_\text{c} = H_\text{c} W_\text{c}$, where H_c and W_c represent the resolution the respective context feature map. Similarly, the number of tokens of the error feature matrix is $N_\text{e} = H_\text{e} W_\text{e}$, and as we have discussed in Section 2.4.2 and Section 3.4.1, the heights of the context feature map and the error feature map are $1/8H$, and the width $1/8W$, where H and W are the height and width of the input images, in a word, $N_e = N_c$. D_c and D_e invoke the channel dimension of the feature maps. The self-similarities of the input frame 1 are modeled by the query and key features [JCL+21], namely $\mathbf{Q}$ and $\mathbf{K}$. The value features $\mathbf{V}$ are the hidden representation of the error features.

So how come these three essential features? As illustrated in Figure 3.9, using the weight matrices $\mathbf{W}_{\text{qry}} \in \mathbb{R}^{D_\text{in} \times D_\text{c}}$ and $\mathbf{W}_{\text{key}} \in \mathbb{R}^{D_\text{in} \times D_\text{c}}$, the query features $\mathbf{Q}$ and the key features $\mathbf{K}$ are the linear projections of the input context feature matrix, while value features $\mathbf{V}$ are linearly projected from another input of the Global Error Features Aggregation model, the error features with the help of the weight matrix $\mathbf{W}_{\text{val}} \in \mathbb{R}^{D_\text{m} \times D_\text{m}}$, such that $\mathbf{Q} = \mathbf{W}_{\text{qry}}\mathbf{x}$, $\mathbf{K} = \mathbf{W}_{\text{key}}\mathbf{x}$, $\mathbf{V} = \mathbf{W}_{\text{val}}\mathbf{y}$. The attention matrix is computed from the query features $\mathbf{Q}$ and the key features $\mathbf{K}$ via the scaled-dot product [VSP+17], which is given by

$$\mathcal{A}\left(\mathbf{Q}, \mathbf{K}\right) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\mathsf{T}}{\sqrt{D_\text{c}}}\right). \tag{3.1}$$

The output of the self-attention operation "$\bigotimes$" is defined as the weighted sum of N features in $\mathbf{V}$ with the weights corresponding to the attention map, which is an attention-weighted sum of projected error features:

$$\text{Attention}\left(\mathbf{K}, \mathbf{Q}, \mathbf{V}\right) = \mathcal{A}\left(\mathbf{K}, \mathbf{Q}\right)\mathbf{V}. \tag{3.2}$$

After aggregating the value features $\mathbf{V}$ with the above-mentioned attention matrix $\mathcal{A}\left(\mathbf{Q}, \mathbf{K}\right)$, in the Aggregrator* shown in Figure 3.9, the Global Error Features Aggregation module obtains its final output, the aggregated error features, via:

$$\mathbf{e} = \mathbf{y} + \alpha\text{Attention}\left(\mathbf{K}, \mathbf{Q}, \mathbf{V}\right) \tag{3.3}$$

The matrices $\mathbf{W}_{\text{qry}}$, $\mathbf{W}_{\text{key}}$, $\mathbf{W}_{\text{val}}$, and the coefficient α, which is initialized to zero are the learnable parameters in the Global Error Features Aggregation module.
Eventually, we concatenate all three feature maps, namely the error feature $\mathbf{y}$, the aggregated error features $\mathbf{e}$ as well as the context features $\mathbf{x}$ into the final concatenated features $[\mathbf{y}|\mathbf{e}|\mathbf{x}]$, which are the input of the following GRU operator, as shown in Figure 3.8, where these features are decoded and

residual flow error are computed. As interpolated by GMA [JCL+21], concatenation allows error vectors to be combined based on global context without any specification on how they should be combined.

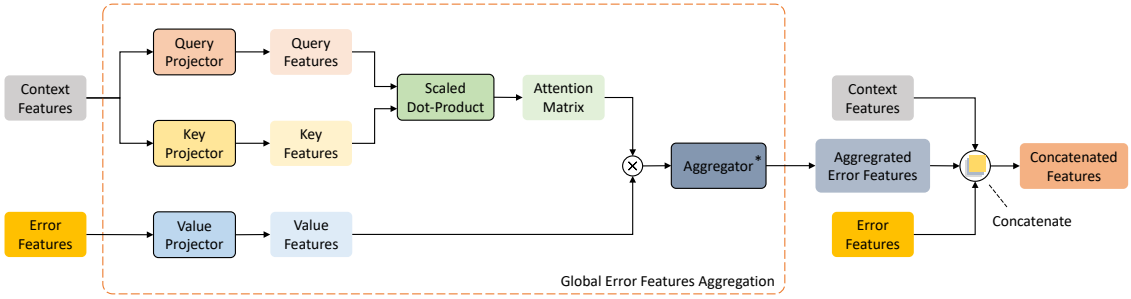


Figure 3.9: Illustration showing details of Global Error Features Aggregation module and the concatenated output features. Image adapted from [JCL+21].

3.6 Chapter Conclusion

The purpose of this chapter is to describe three distinct error estimation networks: the Single-iteration Conv, the RAFT-like error estimation network, and the GMA-like error estimation network. These networks are designed to predict the flow error by taking two consecutive images and the corresponding estimated optical flow. Experimental studies will be conducted using these error estimation networks in conjunction with different flow estimation networks and training settings. As such we can investigate whether adopting the end-point error to the ground truth flow can be used to learn such a flow error in a supervised manner by the proposed error estimation networks as expected in Chapter 4.

4 Experimental Results

In the previous chapter, we proposed and elaborated on the architectures of several error estimation networks. One of them is the Single-iteration Conv error estimation network, and the other two are networks with iterative update operators, namely RAFT-like and GMA-like error estimation networks. In this chapter, we conduct experiments on these error estimation networks in three phases following the schedule FlyingChairs2 → FlyingThings → Sintel-split with two different training settings.

We will introduce the details of the experimental setup and implementation details in Section 4.1 and Section 4.2, then the results of the error estimation networks on the corresponding datasets are explicated in the following sections, along with the ablation studies on some architectures and training settings, including shared encoders and/or the convex upsampler, look-up radius, and training iterations. In the end, we summarize our findings from this chapter in Section 4.7.

4.1 Experimental setup

4.1.1 Training schedule

This section presents the training and validation schedule and the primary evaluation metric. A key objective of this thesis is to investigate the learnability of flow error with end-point error (EPE) to ground truth flow error as a criterion. To this end, our error estimation networks are trained following a dataset schedule adopted by various well-established flow estimation networks, with minor modifications.

The error estimation networks are initially trained on the training split of the FlyingChairs2 dataset [ISKB18] (denoted as "C") and evaluated on the corresponding validation set. We utilize the FlyingChairs2 dataset, rather than the original FlyingChairs dataset [DFI+15], to further verify the performance of the error estimation networks in estimating flow error in both occluded and non-occluded areas of the flow error field. This is made possible by the additional provision of occlusion maps in the FlyingChairs2 dataset, which allow for the division of pixels in the flow field into occluded and non-occluded categories. Following the initial training phase, the error estimation networks are evaluated on the validation set of the FlyingThings3D dataset [MIH+16] and the training split of Sintel[BWSB12] before being further trained with its training set.

After training with FlyingChairs2 → FlyingThings3D (denoted as "T"), these networks will additionally be evaluated on the validation set of the FlyingThings3D dataset. The error estimation networks are also validated on the entire training set of Sintel[BWSB12] to verify their cross-data generalization capabilities.

For the third training phase, we adopt a modified Sintel training dataset (denoted as "S-split"), namely the Sintel-split dataset. Due to the lack of access to pixel-wise ground truth flow for the test set of the Sintel dataset, the original training set is divided into two sets: a new training set and a new validation set. This modification allows us to train the error estimation networks with samples from the relatively challenging Sintel dataset without concerns about overfitting.

The training set of the Sintel-split dataset includes the following scenes: `alley_1`, `ambush_2`, `ambush_5`, `ambush_7`, `bamboo_1`, `bandage_1`, `bandage_2`, `cave_2`, `market_5`, `mountain_1`, `shaman_2`, `shaman_3`, `sleeping_1`, `sleeping_2`, `temple_3`, while the other eight scenes, `alley_2`, `ambush_4`, `ambush_6`, `bamboo_2`, `cave_4`, `market_2`, `market_6`, `temple_2`, are divided into the validation set of the Sintel-split dataset. Specifically, the original 1041 training samples of the Sintel dataset are grouped into 706 training samples and 366 validation samples for the Sintel-split dataset. The error estimation networks are then trained with the combined "Sintel-split-mixed" dataset comprising Sintel-split, FlyingThings, KITTI, and HD1K. We use the clean and final passes of Sintel-split for training and analyze the performance of our error estimation networks on both passes. (Please refer to Section 2.3 for further detailed introductions to the FlyingChairs2, FlyingThings3D, Sintel, HD1K, and KITTI-2015 datasets).

An overview of the training and evaluation datasets adopted in different training phases is provided in Table 4.1. The notation "Sintel-split‡" represents the mixed training dataset consisting of 226737 training samples from Sintel-split, FlyingThings, KITTI, and HD1K, where 31% of these samples are composed of Sintel-split clean pass and another 31% of Sintel-split final pass.

Training Phase	**Training**	**Evaluation**
Phase 1: Chairs	FlyingChairs2	FlyingChairs2 FlyingThings3D
Phase 2: Things	FlyingThings3D	Sintel-train FlyingThings3D
Phase 3: Sintel-split	Sitel-split‡	Sintel-split

Table 4.1: The Overview of training and evaluation dataset of entire training phases.

4.1.2 Evaluation Metric

In order to evaluate the performance of the error estimation networks quantitatively, the main evaluation metric we use in this chapter is the Average End-Point Error (AEPE) of the estimated flow error field Error_{es} with respect to the corresponding ground truth flow error field Error_{gt}. Average pixel-wise loss error is defined similarly to flow error in Section 2.2. It follows intuitively that the end-point error of the flow error is averaged over the entire flow error field. This end-point error refers to the loss error previously introduced in Section 3.1. The ground truth flow error vector of the pixel (i, j) in the flow error field is denoted as $e_{\text{gt}}(i, j) = (u_{\text{gt}}(i, j), v_{\text{gt}}(i, j))^{\mathsf{T}}$, while

$e(i,j) = (u(i,j), v(i,j))^T$ denotes the estimated optical flow error vector of the pixel (i,j). The End-point Error (EE) between these two flow error vectors yields:

$$\mathrm{EE}(e(i,j), e_{gt}(i,j)) = \frac{1}{H \times W} \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} \sqrt{(u(i,j) - u_{gt}(i,j))^2 + (v(i,j) - v_{gt}(i,j))^2}. \tag{4.1}$$

The average end-Point error of the estimated flow error field is formulated as follows:

$$\mathrm{AEE}(\mathrm{Error}_{es}, \mathrm{Error}_{gt}) = \frac{1}{H \times W} \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} \mathrm{EE}(e(i,j), e_{gt}(i,j)). \tag{4.2}$$

4.2 Implementation Details

Using the mixed precision strategy, our error estimation networks are trained on a GeForce RTX 3090 GPU with a memory size of 24 GB on `PyTorch` library [PGC+17].

4.2.1 Training Loss

With the aim to investigate the effectiveness of using the end-point error to the ground truth flow, namely the ground truth flow error, to predict the flow error in the supervised fashion, the following loss functions are adopted for the purpose of supervising the update of the parameters during training:

the loss of the Single-iteration Conv error estimation networks during training is computed between the estimated flow error Error_{es} and the ground-truth flow error Error_{gt} with L1 distance:

$$\mathcal{L}_{\mathrm{loss}} = \left\| \mathrm{Error}_{gt} - \mathrm{Error}_{es} \right\|_1 . \tag{4.3}$$

On account of the iterative update operator, an alternative training loss is adopted for RAFT-like and GMA-like error estimation networks. We imitate how the renowned flow estimation network RAFT [TD20] supervises its training process. Intuitively, error estimation networks with these two architectures are supervised based on the L1 distance between the estimated flow error and the ground-truth flow error over every refinement. Given the ground-truth flow error Error_{gt}, and the estimated flow error Error_{es_i} of GRU iteration i, i = 1, ..., N, where N represents the total number of GRU iterations. The loss is defined as follows:

$$\mathcal{L}_{\mathrm{loss}} = \sum_{i=1}^{N} \gamma^{N-i} \left\| \mathrm{Error}_{gt} - \mathrm{Error}_{es_i} \right\|_1 , \tag{4.4}$$

where γ is the decay coefficient smaller than 1, whose function is to apply exponentially increasing weights to the L1 distance between the estimated flow error of each refinement and the ground-truth flow error. The finer the estimated flow error, the larger the contribution of the corresponding L1 distance to the loss function in Equation (4.4).

4.2.2 Flow Estimation Networks

Before we move forward to hyperparameter settings, it is essential to choose an appropriate flow estimation network first so that we can obtain the required estimated flow at the resolution of $1/8\ \langle H, W\rangle$ as one of the inputs to the error estimation networks. This estimated flow is the output of the iterative update operator of the corresponding flow estimation network before being upsampled to full resolution. As explained in Chapter 3, we adopt the estimated flow at 1/8 resolution instead of full resolution to avoid the potential information loss during the upsampling-downsampling operation. Moreover, the flow estimation network also helps to produce the ground truth flow error (Please refer to Section 3.1 for more details on the ground truth flow error.) to supervise the training of the error estimation networks. Here, it is important to note that the estimated flow used to produce the ground truth flow error is at full resolution. Because both the ground truth optical flow provided by the datasets and the later computed ground truth flow error which is used to supervise the parameter update are at full resolution. In conclusion, the estimated flow error at 1/8 resolution and full resolution are needed for the training of error estimation networks.

We select the following two flow estimation networks:

1. The flow estimation network with the architecture of RAFT [TD20], which is trained with FlyingChairs2 → FlyingThings3D following the training settings provided by RATF[TD20], is chosen to produce the required estimated flow. This flow estimation network is denoted as "$\text{RAFT}_{\text{C+T}}$".
2. Similarly, we also adopt another flow estimation network with the architecture of GMA [JCL+21], which is also trained with FlyingChairs2 and FlyingThings following the training settings adopted by GMA[JCL+21]. This flow estimation network is denoted as "$\text{GMA}_{\text{C+T}}$".

Since both flow estimation networks are not trained with samples from the Sintel or Sintel-split datasets, they are suitable for investigating the learnability of flow errors on various datasets. We can examine whether the flow error can be learned on the samples with complex motions if the flow estimation networks are not fine-tuned on the target dataset. Moreover, these flow estimation networks are trained with FlyingChairs2 → FlyingThings3D instead of FlyingChairs2 only, which means they both have a good generalization performance on the Sintel training set while the produced estimated flow is still not too abbreviated from the ground truth flow of FlyingChairs2 dataset.

RAFT flow estimation networks trained with FlyingChairs2 alone (denoted as "RAFT_{C}") are also used for the ablation study in Section 4.6 and the semi-supervised method of flow estimation in Chapter 5.

The upper part of Table 4.2 presents the performance of the above-mentioned flow estimation networks $\text{RAFT}_{\text{C+T}}$ and $\text{GMA}_{\text{C+T}}$ on the validation proportions of FlyingChairs2, Sintel, and Sintel-split. The bottle half of the table also reports the results of RAFT_{C} on the FlyingChairs2 validation dataset. To guarantee the consistency and comparability of the networks, we adopt the results of the flow estimation networks which are trained on the same GPU instead of the published ones.

Flow	Training Dataset	Chairs	Sintel		Sintel-split	
			clean	final	clean	final
$RAFT_{C+T}$	C+T	1.121	1.475	2.769	2.025	3.166
GMA_{C+T}		1.147	1.367	2.803	1.769	3.438
$RAFT_{C}$	C	0.867				

Table 4.2: Results of flow estimation networks appear in this work.

4.2.3 Experimental settings

As part of our investigation of whether the error estimation networks will work on different datasets and in different phases, we follow the two training settings adopted by RAFT [TD20] and GMA [JCL+21] We refer to these two training settings as *RAFT and GMA training settings*, respectively. Following these two training procedures, the AdamW [LH17] optimizer and the one-cycle policy [ST19] are adopted. During the first training phase, we use a batch size of 10. When the error estimation networks are trained with FlyingThings3D and Sintel-split, we use batch sizes of 6 for both phases. During training, the total number of GRU iterations N is always set to 12, while N varies with different datasets during validation. N=24 when the error estimation network is validated with FlyingChairs2 or FlyingThings3D, while N is set to 32 for inference on Sintel or Sintel-split. The decay coefficient γ of the loss function in Equation (4.4) is set to 0.8 for the first two training phases, while γ is set to 0.85 when the error estimation network is trained with the mixed Sintel-split dataset. All the data argumentations also follow the original code of RAFT[TD20] and GMA[JCL+21].

The differences between RAFT and GMA training settings are listed as follows. For RAFT and GMA training settings, the highest learning rate of the one-cycle policy is set to 0.0004 and 0.00025 during phase 1 respectively, then it is set to 0.000125 during the next two phases for both settings; When we adopt the RAFT training settings, We train the error estimation networks for 100,000 iterations in each phase, while the error estimation networks are trained for 120,000 iterations in each phase with the GMA training settings; The batch normalization is frozen during the second and third phases with the RAFT training settings, whereas no normalization is frozen during the whole training process if we adopt the GMA training settings.

4.2.4 Error Estimation Networks

In Chapter 3 we explain the pipeline of three different error estimation networks. In this chapter, experiments are performed on the combinations of the specific flow and error estimation networks. The error estimations networks with various architectures are also trained with different training settings.

We only train the Single-iteration Conv error estimation network with the corresponding training settings of the flow estimation network. Therefore, When the flow estimation network is $RAFT_{C+T}$, the Single-iteration Conv error estimation network is trained according to the RAFT training schedule; When the flow estimation network is GMA_{C+T}, the Single-iteration Conv error estimation network is trained with the GMA training schedule. A cross-check of the influence of RAFT and

GMA training settings on the error estimation networks was performed by training the other two error estimation networks, the RAFT-like network and the GMA-like network, using different training settings. This means that the RAFT-like error estimation networks are trained with the RAFT and the GMA training settings, and GMA-like ones are also the case.

Additionally, we also combine different flow estimation networks with different error estimation networks to verify whether such cross-cases can lead to better results. All the combinations of flow estimation networks, error estimation networks, and training settings are listed in Table 4.3. The superscript "*" refers to the error estimation network that is trained with the RAFT training schedule mentioned above; The superscript "†" refers to the error estimation network that is trained with the GMA training schedule. "Conv" represents the Single-iteration Conv error estimation network, while RAFT and GMA refer to the RAFT-like and the GMA-like ones.

Flow Estimation Network	Error Estimation Network
$\text{RAFT}_{\text{C+T}}$	$\text{RAFT}^\star$
	$\text{RAFT}^\dagger$
	$\text{GMA}^\star$
	$\text{GMA}^\dagger$
	$\text{Conv}^\star$
$\text{GMA}_{\text{C+T}}$	$\text{GMA}^\dagger$
	$\text{GMA}^\star$
	$\text{RAFT}^\dagger$
	$\text{RAFT}^\star$
	$\text{Conv}^\dagger$

Table 4.3: The combinations of various flow and error estimation networks following two training settings. The superscript "*" refers to the error estimation network that is trained with the RAFT training settings; The superscript "†" refers to the error estimation network that is trained with the GMA training settings.

4.3 Phase 1: FlyingChairs2

In this section, we present the results of the conducted experiments on the error estimation networks of the first phase: FlyingChairs2. The error estimation networks are trained with FlyingChairs2 for 100,000 iterations (RAFT training settings) or 120.000 iterations (GMA training settings). Then they are evaluated with FlyingChairs2 to investigate the learnability of the flow error on the FlyingChairs2 dataset. The effectiveness of the error estimation networks at estimating the flow error of occluded and non-occluded pixels is also separately verified and presented as loss error in the corresponding regions. To give a clear image of whether the loss error is decreasing during the training process, we also compare the loss error of the error estimation networks that are trained with the first 5,000 iterations and the networks that have finished the first phase of training, We do not adopt the loss error of the plain error estimation networks which are not trained with any samples because the parameters of these networks are randomly initialized and thus non-comparable.

4.3.1 Quantitative Results

Table 4.4 shows the evaluation results of the error estimation networks on the FlyingChairs2 dataset. The AEPE refers to the average end-point error of the flow error, namely the loss error. As shown in Table 4.4, the combination of the flow estimation network $\text{RAFT}_{\text{C+T}}$ and the error estimation network $\text{GMA}^{\dagger}$ achieves the most accurate results on the validation portion of FlyingChairs2 with a loss error of 1.021 among all the combinations. It is imperative to point out that both loss error in occluded and non-occluded areas decreases during training for all the combinations of flow and error estimation networks except the combinations with the Single-iteration Conv error estimation network. The improvement in the performance of these two combinations is insignificant in comparison to the other eight combinations. Positive improvements are indicated by a gray background in the "Relative Improvement" column. These results suggest that flow error can be estimated during the first phase with the proposed RAFT-like and GMA-like error estimation networks.

Flow	Error	Type	Training Progress		Rel. Improv.	Flow	Error	Type	Training Progress		Rel. Improv.
			5K	Phase 1					5K	Phase 1	
$\text{RAFT}_{\text{C+T}}$	$\text{RAFT}^{\star}$	All	1.136	1.049	+7.62%	$\text{GMA}_{\text{C+T}}$	$\text{GMA}^{\dagger}$	All	1.159	1.052	+9.19%
		Noc	0.894	0.826	+7.57%			Noc	0.913	0.827	+9.40%
		Occ	5.103	4.712	+7.66%			Occ	5.185	4.740	+8.58%
	$\text{RAFT}^{\dagger}$	All	1.139	1.030	+9.56%		$\text{GMA}^{\star}$	All	1.155	1.069	+7.44%
		Noc	0.897	0.811	+9.55%			Noc	0.909	0.840	+7.62%
		Occ	5.111	4.621	+9.58%			Occ	5.183	4.824	+6.92%
	$\text{GMA}^{\star}$	All	1.141	1.037	+9.11%		$\text{RAFT}^{\dagger}$	All	1.161	1.050	+9.58%
		Noc	0.899	0.816	+9.22%			Noc	0.915	0.825	+9.83%
		Occ	5.113	4.664	+8.80%			Occ	5.190	4.731	+8.84%
	$\text{GMA}^{\dagger}$	All	1.143	**1.021**	+10.63%		$\text{RAFT}^{\star}$	All	1.144	1.077	+5.86%
		Noc	0.901	**0.805**	+10.61%			Noc	0.897	0.844	+5.99%
		Occ	5.110	**4.564**	+10.69%			Occ	5.178	4.892	+5.52%
	Conv*	All	1.126	1.125	+0.10%		Conv†	All	1.147	1.146	+0.12%
		Noc	0.883	0.882	+0.16%			Noc	0.901	0.899	+0.14%
		Occ	5.099	5.103	-0.07%			Occ	5.183	5.182	+0.04%

Table 4.4: Comparison of loss error of various combinations of flow and error estimation networks in phase 1: FlyingChairs2. The superscript "$\star$" refers to the error estimation network that is trained with the RAFT training settings mentioned above; The superscript "$\dagger$" refers to the error estimation network that is trained with the GMA training settings. The best performance is denoted with bold font.

4.3.2 Qualitative Results

Figure 4.1 provides example visual results on the FlyingChairs2 validation dataset, which demonstrate that flow error can be estimated with the error estimation networks. The first row is the ground truth flow error of all pixels, non-occluded pixels, and occluded pixels, while the last row is the estimated flow error in the corresponding regions. The ground truth flow error and the estimated flow error are generated by the combination of $\text{RAFT}_{\text{C+T}}$ and $\text{GMA}^{\dagger}$. One limitation of this qualitative result

however is the blurred and misleading estimation of the fine structure around the boundaries of the chairs. This may result from the limitation of the feature encoders and the resulting correlation pyramid architecture we adopt in the networks.

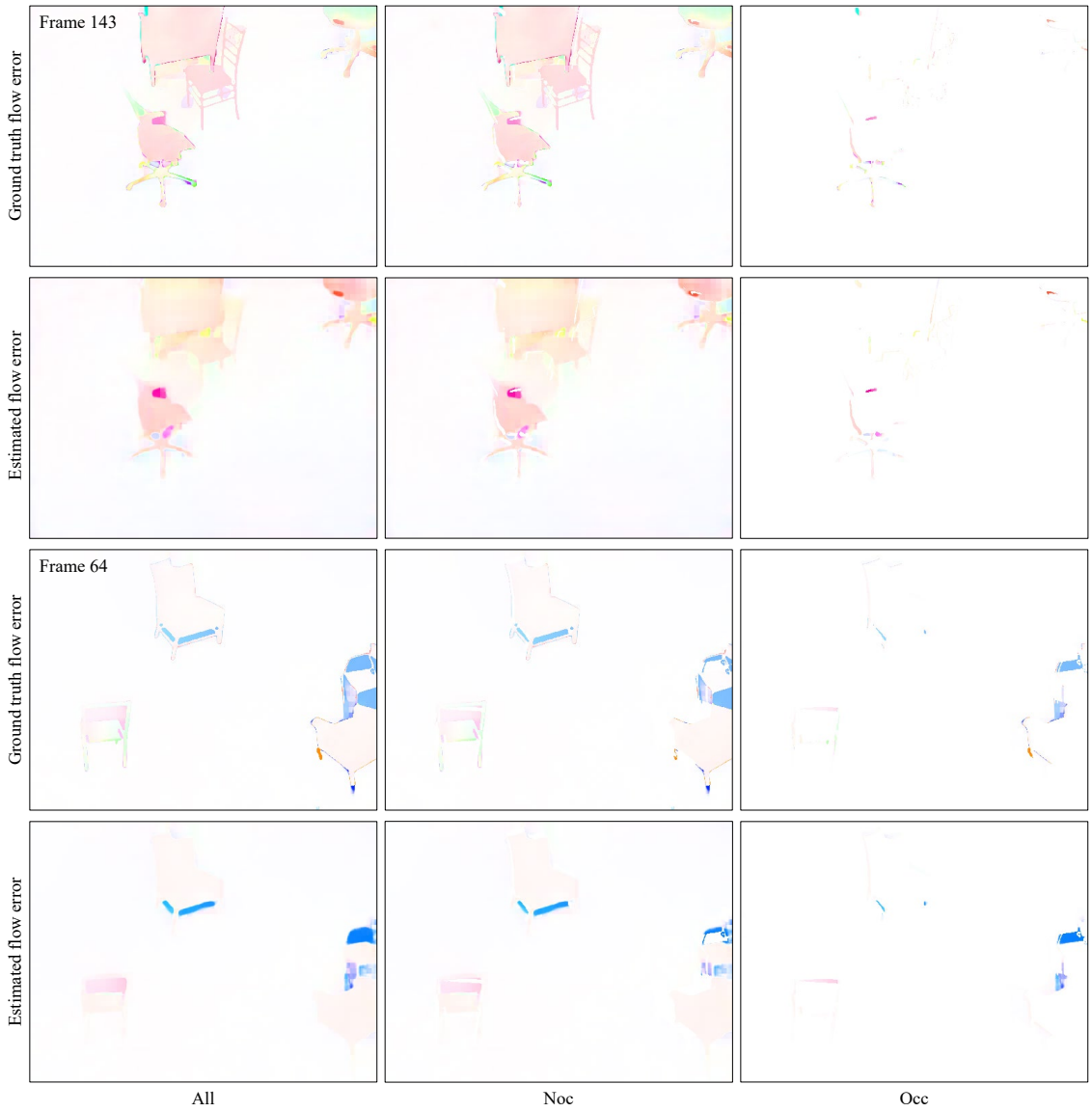


Figure 4.1: Visualized results of estimated flow error on FlyingChairs2 dataset. The ground truth flow error and the estimated flow error are generated by the combination of $\text{RAFT}_{\text{C+T}}$ and $\text{GMA}^{\dagger}$.

4.4 Phase 2: FlyingThings

In the second phase of training, the error estimation networks are trained with FlyingThings3D for another 100,000 or 120,000 iterations. A comparison of the accuracy of flow error on the validation dataset of FlyingThings3D is conducted in order to determine the learnability of flow error. Additionally, the error estimation networks are evaluated on the entire Sintel benchmark training set in order to determine their generalization performance.

4.4.1 Quantitative Results

Table 4.5 compares the accuracy of the estimated flow errors on the validation set of FlyingThings3D benchmarks. The quality of the various combinations of flow and error estimation networks is listed in the table. In the analyses of data, we can come to the conclusion that the flow error can be predicted on both clean and final passes of FlyingThings3D. On the clean pass of FlyingThings3D, the combination of GMA_{C+T} and $GMA^{\dagger}$ achieve the best result with AEPE = 14.002, while on the final pass, the best performance is acquired by the combination of GMA_{C+T} and $RAFT^{\dagger}$ with AEPE = 13.155.

Data Type Things	Flow	Error	Training Progress		Rel. Improve.	Flow	Error	Training Dataset		Rel. Improve.
			C	C+T				C	C+T	
clean	$RAFT_{C+T}$	$RAFT^{\star}$	14.781	14.146	+4.30%	GMA_{C+T}	$GMA^{\dagger}$	16.213	**14.002**	+13.64%
		$RAFT^{\dagger}$	14.753	14.134	+4.19%		$GMA^{\star}$	15.462	14.016	+9.35%
		$GMA^{\star}$	16.145	14.108	+12.62%		$RAFT^{\dagger}$	14.891	14.011	+5.91%
		$GMA^{\dagger}$	16.219	14.076	+13.21%		$RAFT^{\star}$	14.813	14.017	+5.38%
		$Conv^{\star}$	14.169	14.169	0.00%		$Conv^{\dagger}$	14.022	14.021	+0.01%
final	$RAFT_{C+T}$	$RAFT^{\star}$	14.004	13.624	+2.71%	GMA_{C+T}	$GMA^{\dagger}$	14.495	13.178	+9.09%
		$RAFT^{\dagger}$	14.099	13.581	+3.67%		$GMA^{\star}$	14.035	13.157	+6.26%
		$GMA^{\star}$	14.621	13.567	+7.21%		$RAFT^{\dagger}$	13.902	**13.155**	+5.37%
		$GMA^{\dagger}$	14.668	13.566	+7.51%		$RAFT^{\star}$	13.696	13.159	+3.92%
		$Conv^{\star}$	13.643	13.642	+0.01%		$Conv^{\dagger}$	13.160	13.159	+0.01%

Table 4.5: Comparison of loss error of various combinations of flow and error estimation networks on the FlyingThings3D validation dataset in phase 2:FlyingThings3D. The superscript "*" refers to the error estimation network that is trained with the RAFT training settings mentioned above; The superscript "†" refers to the error estimation network that is trained with the GMA training settings. The best performance is denoted with bold font.

In Table 4.6, we report the AEPE of flow error over all pixels (All), over occluded pixels (Occ), and over pixels in non-occluded regions (Noc). The results suggest that the flow error can also be predicted on training samples of Sintel. This is on account of the fact that the performances of the error estimation networks improve after the second phase compared to the ones only trained with FlyingChairs2.

The only exception is still the Single-iteration Conv error estimation networks, where no improvement in estimated flow error appeared during the second phase. A closer inspection of the table shows that the loss error of the Single-iteration Conv error estimation networks is the most accurate among all error estimation networks. This strange finding, however, reveals that the end-point error to the ground truth flow error is not the most suitable criterion for inference of such a flow error to

some extent, or at least cannot be used as the only criterion. Although the Single-iteration Conv error estimation networks achieve the best AEPE of flow error, the visualized results show that the networks do not learn much useful in comparison to other error estimation networks. More details of the visualized results are illustrated in Section 4.4.2.

Except for the Single-iteration Conv error estimation networks, the best result on the clean pass of the Sintel training dataset is acquired by the combination of $\text{GMA}_{\text{C+T}}$ and $\text{GMA}^{\star}$, whereas the combination of $\text{RAFT}_{\text{C+T}}$ and $\text{GMA}^{\star}$ achieves the best performance on the final pass. We should note that the quality of the estimated flow also influences the results given that the flow estimation network $\text{GMA}_{\text{C+T}}$ has a better estimation on the clean pass with an AEPE (of flow) of 1.367 and $\text{RAFT}_{\text{C+T}}$ is superior to $\text{GMA}_{\text{C+T}}$ on the final pass with an AEPE (of flow) of 2.769.

Data Type Sintel(train)	Flow	Error	Type	Training Progress		Rel. Improve.	Flow	Error	Type	Training Progress		Rel. Improve.
				C	C+T					C	C+T	
clean	$\text{RAFT}_{\text{C+T}}$	$\text{RAFT}^{\star}$	All	1.750	1.479	+15.50%	$\text{GMA}_{\text{C+T}}$	$\text{GMA}^{\dagger}$	All	1.934	1.367	+29.34%
			Noc	0.885	0.662	+25.19%			Noc	1.028	0.620	+39.66%
			Occ	12.723	11.839	+6.96%			Occ	13.434	10.839	+19.31%
		$\text{RAFT}^{\dagger}$	All	1.713	1.480	+13.60%		$\text{GMA}^{\star}$	All	2.030	**1.364**	+32.80%
			Noc	0.864	0.665	+23.06%			Noc	1.073	**0.618**	+42.38%
			Occ	12.484	11.823	+5.30%			Occ	14.186	10.838	+23.60%
		$\text{GMA}^{\star}$	All	2.412	1.473	+38.94%		$\text{RAFT}^{\dagger}$	All	1.651	1.366	+17.23%
			Noc	1.378	0.665	+51.73%			Noc	0.841	0.620	+26.27%
			Occ	15.537	11.722	+24.55%			Occ	11.925	10.835	+9.14%
		$\text{GMA}^{\dagger}$	All	2.139	1.480	+30.80%		$\text{RAFT}^{\star}$	All	1.478	1.366	+7.58%
			Noc	1.186	0.673	+43.21%			Noc	0.720	0.620	+13.86%
			Occ	14.233	11.717	+17.68%			Occ	14.233	11.717	+17.68%
		$\text{Conv}^{\star}$	All	1.474	1.473	+0.03%		$\text{Conv}^{\dagger}$	All	1.366	1.366	0.00%
			Noc	0.655	0.655	+0.08%			Noc	0.619	0.619	-0.01%
			Occ	11.860	11.860	0.00%			Occ	10.806	**10.806**	0.00%
final	$\text{RAFT}_{\text{C+T}}$	$\text{RAFT}^{\star}$	All	2.941	2.761	+6.15%	$\text{GMA}_{\text{C+T}}$	$\text{GMA}^{\dagger}$	All	3.216	2.802	+12.88%
			Noc	1.799	1.610	+10.49%			Noc	2.809	1.682	+19.50%
			Occ	17.442	17.361	+0.47%			Occ	13.434	10.839	+19.31%
		$\text{RAFT}^{\dagger}$	All	3.022	2.730	+9.68%		$\text{GMA}^{\star}$	All	3.456	2.799	+19.02%
			Noc	1.867	1.587	+15.00%			Noc	2.220	1.680	+24.33%
			Occ	17.683	17.233	+2.54%			Occ	17.514	17.011	+2.87%
		$\text{GMA}^{\star}$	All	3.077	2.751	+10.25%		$\text{RAFT}^{\dagger}$	All	3.534	2.802	+20.69%
			Noc	1.909	1.594	+16.48%			Noc	2.270	1.683	+25.85%
			Occ	17.898	17.436	+2.58%			Occ	19.569	17.006	+13.10%
		$\text{GMA}^{\dagger}$	All	3.032	**2.721**	+10.27%		$\text{RAFT}^{\star}$	All	3.029	2.802	+7.52%
			Noc	1.889	**1.581**	+16.35%			Noc	1.910	1.682	+11.94%
			Occ	17.534	**17.192**	+1.95%			Occ	17.237	17.013	+1.30%
		$\text{Conv}^{\star}$	All	2.740	2.739	+0.01%		$\text{Conv}^{\dagger}$	All	2.802	2.802	0.00%
			Noc	1.588	1.588	+0.02%			Noc	1.683	1.683	0.00%
			Occ	17.352	17.352	0.00%			Occ	17.009	17.009	0.00%

Table 4.6: Comparison of loss error of various combinations of flow and error estimation networks on the Sintel training dataset in phase 2: FlyingThings3D. The superscript "*" refers to the error estimation network that is trained with the RAFT training settings mentioned above; The superscript "†" refers to the error estimation network that is trained with the GMA training settings. The best performance is denoted with bold font.

4.4.2 Qualitative Results

Figure 4.2 provides qualitative results on the clean and final pass of the training split of Sintel, which demonstrates that flow error can be estimated with our error estimation network and the accuracy of the estimated flow error improves compared to the ones only trained with Chairs2. For the results of clean pass, the ground truth flow error and the estimated flow error are generated by the combination of $\text{GMA}_{\text{C+T}}$ and $\text{GMA}^{\star}$, while they are generated by the combination $\text{RAFT}_{\text{C+T}}$ and $\text{GMA}^{\dagger}$ for the final pass.

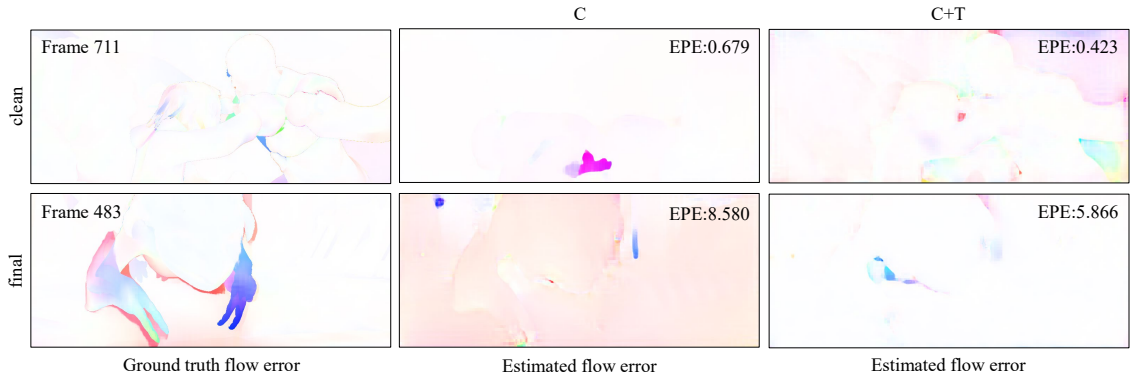


Figure 4.2: Visualized results of estimated flow error on Sintel training dataset. The estimated flow error on the clean pass of Sintel is generated by the combination $\text{GMA}_{\text{C+T}}$ and $\text{GMA}^{\star}$, while the one on the final pass is generated by the combination $\text{RAFT}_{\text{C+T}}$ and $\text{GMA}^{\dagger}$.

4.5 Phase 3: Sintel-split

We have trained our error estimation networks using the FlyingChairs → FlyingThings schedule. Lastly, the error estimation networks are trained on the Sintel-split dataset during the third phase,. These networks are then evaluated on the validation split of Sintel-split.

4.5.1 Quantitative Results

The results are presented in Table 4.7 for a variety of combinations of flow and error estimation networks on the Sintel-split validation set. As shown in the table, the combination of $\text{GMA}_{\text{C+T}}$ and $\text{GMA}^{\dagger}$ outperforms other components on the clean pass of the Sintel-split validation set. On the final pass, the combination of $\text{RAFT}_{\text{C+T}}$ and $\text{GMA}^{\star}$ surpasses all the other combinations with an AEPE of 3.048. Interestingly, not all the combinations of flow and error estimation networks are able to estimate the flow error during this phase, especially on the relatively “difficult“ final pass. As with the previous two phases, the Single-iteration Convolution error estimation networks failed to predict the flow error once again. As a consequence of misleading predictions in non-occluded areas, the performance of the other ineffective error estimation networks has been degraded. Except for the combinations of $\text{GMA}_{\text{C+T}}$ and $\text{RAFT}^{\dagger}$, all the other error networks with the iterative operator can slightly enhance the accuracy of the estimated flow error in the occluded areas after fine-tuning the networks with Sintel-split. These findings suggest that the final performance of the

error estimation networks is a balance between the improved estimation in occluded areas and the consequent misguided prediction of non-occluded pixels. Additionally, these experiments indicate the importance of the Global Error Feature Aggregation module as both networks that achieve the best performance on the clean and final passes are GMA-like error estimation networks.

The mediocre performance of the error estimation networks may be caused by the limitation of the correlation pyramid which is constructed from the feature encoder we adopt. The dataset we used in this phase could also contribute to these findings. It cannot be guaranteed that the training set of Sintel-split contains all the characteristics of the flow in the validation set due to the fact that the training samples of Sintel are divided into training and validation splits by us.

Data Type Sintel-split	Flow	Error	Type	Training Progress		Rel. Improve.	Flow	Error	Type	Training Progress		Rel. Improve.
				C+T	C+T+S-split					C+T	C+T+S-split	
clean	$RAFT_{C+T}$	RAFT*	All	2.025	1.791	+11.56%	GMA_{C+T}	GMA†	All	1.761	**1.597**	+9.27%
			Noc	1.047	0.943	+9.89%			Noc	0.920	0.861	+6.44%
			Occ	12.229	10.634	+13.05%			Occ	10.522	**9.275**	+11.85%
		RAFT†	All	2.067	1.771	+14.33%		GMA*	All	1.761	1.606	+8.78%
			Noc	1.065	0.950	+10.87%			Noc	0.920	**0.855**	+7.04%
			Occ	12.510	10.344	+17.39%			Occ	10.531	9.440	+10.36%
		GMA*	All	2.053	1.726	+15.92%		RAFT†	All	1.759	1.827	-3.89%
			Noc	1.074	0.902	+16.04%			Noc	0.920	0.991	-7.82%
			Occ	12.264	10.326	+15.80%			Occ	10.509	10.541	-0.30%
		GMA†	All	2.048	1.727	+15.64%		RAFT*	All	1.759	1.755	+0.26%
			Noc	1.065	0.931	+12.57%			Noc	0.920	0.957	-4.02%
			Occ	12.295	10.030	+18.42%			Occ	10.517	10.079	+4.16%
		Conv*	All	2.031	2.031	+0.03%		Conv†	All	1.761	1.761	+0.01%
			Noc	1.047	1.048	-0.01%			Noc	0.920	0.920	0.00%
			Occ	12.286	12.285	+0.01%			Occ	10.526	10.525	+0.01%
final	$RAFT_{C+T}$	RAFT*	All	3.166	3.178	-0.38%	GMA_{C+T}	GMA†	All	3.256	3.166	+2.78%
			Noc	2.017	2.061	-2.18%			Noc	2.074	2.106	+6.44%
			Occ	15.151	14.818	+2.13%			Occ	15.579	14.216	+8.75%
		RAFT†	All	3.133	3.177	-1.41%		GMA*	All	3.255	3.313	-1.79%
			Noc	2.005	2.103	-4.86%			Noc	2.073	2.186	-5.42%
			Occ	14.894	14.383	+3.43%			Occ	15.571	15.064	+3.26%
		GMA*	All	3.126	**3.048**	+2.51%		RAFT†	All	3.255	3.364	-3.35%
			Noc	1.997	**2.018**	-1.03%			Noc	2.072	2.149	-7.08%
			Occ	14.900	13.790	+7.45%			Occ	15.587	15.305	+1.81%
		GMA†	All	3.144	3.064	+2.55%		RAFT*	All	3.257	3.212	+1.40%
			Noc	2.008	2.044	-1.81%			Noc	2.075	2.141	-3.20%
			Occ	14.995	**13.700**	+8.63%			Occ	15.587	14.480	+7.10%
		Conv*	All	3.136	3.136	+0.01%		Conv†	All	3.258	3.258	0.00%
			Noc	2.005	2.005	0.00%			Noc	2.075	2.075	0.00%
			Occ	14.928	14.927	+0.01%			Occ	15.590	15.590	+0.01%

Table 4.7: Comparison of loss error of various combinations of flow and error estimation networks on the Sintel-split validation dataset in phase 3: Sintel-split. The superscript "*" refers to the error estimation network that is trained with the RAFT training settings mentioned above; The superscript "†" refers to the error estimation network that is trained with the GMA training settings. The best performance is denoted with bold font.

4.5.2 Qualitative Results

Qualitatively, Figure 4.3 demonstrates that it is possible to estimate the flow error in some samples of the Sintel-split validation set. The fine structures around the motion boundaries of the flow error are still wicked issues for error estimation networks. We also present some failed estimations of flow error during phase 3 in Figure 4.4. The major issue is observed in the non-occluded areas of the flow error field. As a result of the misleading estimation in these areas, the predicted flow error field is worsen after training phase 3.

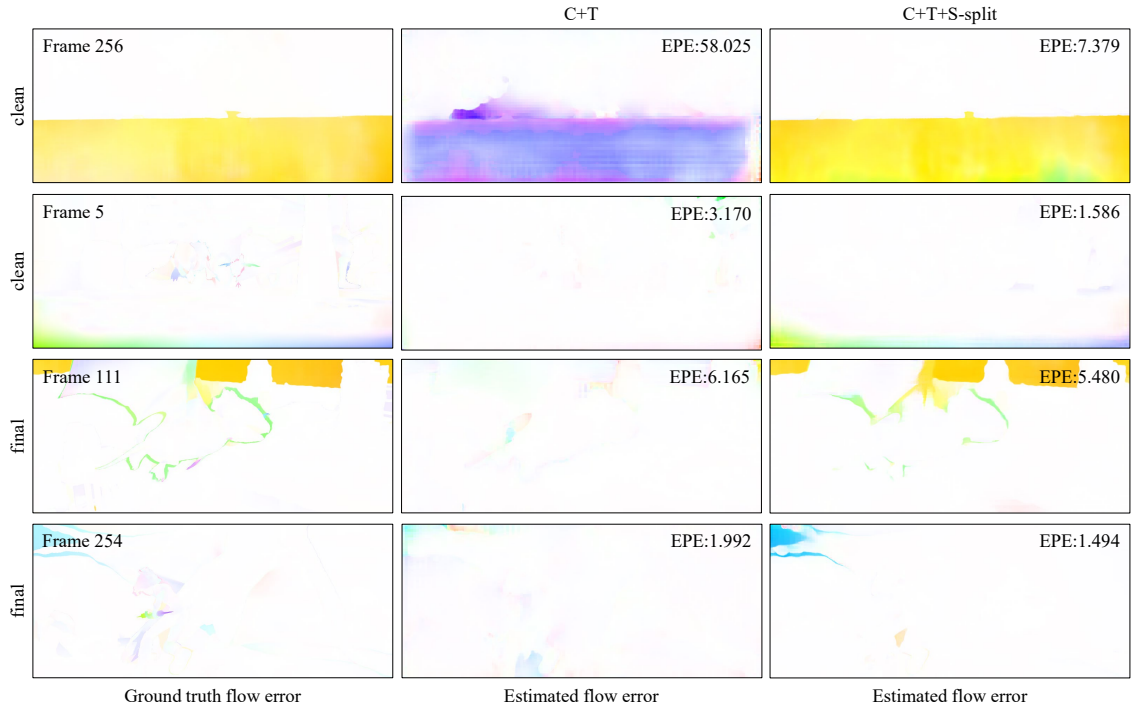


Figure 4.3: Visualized results of estimated flow error on Sintel-split evaluation dataset. The estimated flow error on the clean pass of Sintel-split is generated by the combination $\text{GMA}_{\text{C+T}} + \text{GMA}^{\dagger}$, while the one on the final pass is generated by the combination $\text{RAFT}_{\text{C+T}} + \text{GMA}^{\star}$.

To qualitatively compare the difference between the estimated flow error generated by the Single-iteration Conv error estimation networks and the one with iterative update operators, we visualized the results in Figure 4.5. The visual results in Figure 4.5 demonstrate that the End-point Error of the estimated flow error to the ground truth may not be the only reasonable criterion for inference. If we look closely at the results, we can find that in phase 2, the estimated flow errors produced by $\text{RAFT}_{\text{C+T}} + \text{Conv}^{\dagger}$ are look-alike and do not express the correct direction of estimations despite the relatively acceptable End-point Error. The most surprising aspect of these results is that the estimated flow errors produced by $\text{GMA}_{\text{C+T}} + \text{Conv}^{\dagger}$ still demonstrate some akin structures to the ground truth flow error in phase 1 and phase 3, though the entire flow error field is masked with the wrong direction of estimations. This may be the culprit behind the quantitative results

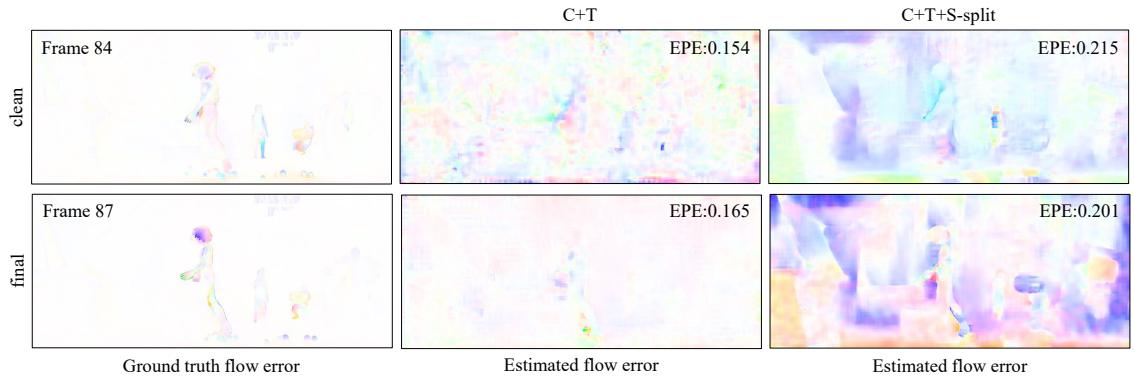


Figure 4.4: Visual examples of failed estimated flow error on Sintel-split evaluation dataset. The estimated flow error on the clean pass of Sintel-split is generated by the combination $\text{GMA}_{\text{C+T}}$ + $\text{GMA}^{\dagger}$, while the one on the final pass is generated by the combination $\text{RAFT}_{\text{C+T}}$ + $\text{GMA}^{\star}$.

of $\text{GMA}_{\text{C+T}}$ + $\text{Conv}^{\dagger}$ and $\text{GMA}_{\text{C+T}}$ + $\text{Conv}^{\dagger}$ which run on the spot in phase 1 and phase 3. In contrast, the visual results of the estimated flow error produced by $\text{GMA}_{\text{C+T}}$ + $\text{GMA}^{\dagger}$ show a positive direction to the ground truth flow error in all three phases.

4.6 Ablations

We perform exhaustive ablation experiments to show the relative importance of the encoders and the convex upsampler in the error estimation network design. Furthermore, ablated experiments on the training iterations and the look-up radius for constructing the correlation cost volume are also conducted. These two ablation studies aim to search for potential improvements in the performance of the proposed error estimation network. These networks are crucial to the studies in Chapter 5. In the following, we elaborate on each of the above-mentioned experiments in more detail.

4.6.1 Sharing Parameters of Encoder and/or Umsampler

Among the error estimation networks and the flow estimation networks listed in Table 4.3, there are two combinations of flow and error estimation networks that are quite unique: the flow estimation network $\text{RAFT}_{\text{C+T}}$ + the error estimation network $\text{RAFT}^{\star}$ and the flow estimation network $\text{GMA}_{\text{C+T}}$ + the error estimation network $\text{GMA}^{\dagger}$. A common characteristic exists between these two combinations. Take the combination $\text{RAFT}_{\text{C+T}}$ + $\text{RAFT}^{\star}$ for example: In addition to the same architecture of feature and context encoders as well as the upsampler, both networks are trained following the identical training procedure with the exact same experimental settings. For the combination $\text{GMA}_{\text{C+T}}$ + $\text{GMA}^{\dagger}$ is also the same case. There arises a question in our minds: whether the error estimation network should share the same parameters of the encoders and/or the upsampling mask as the flow one to promote accuracy as well as the model efficiency?

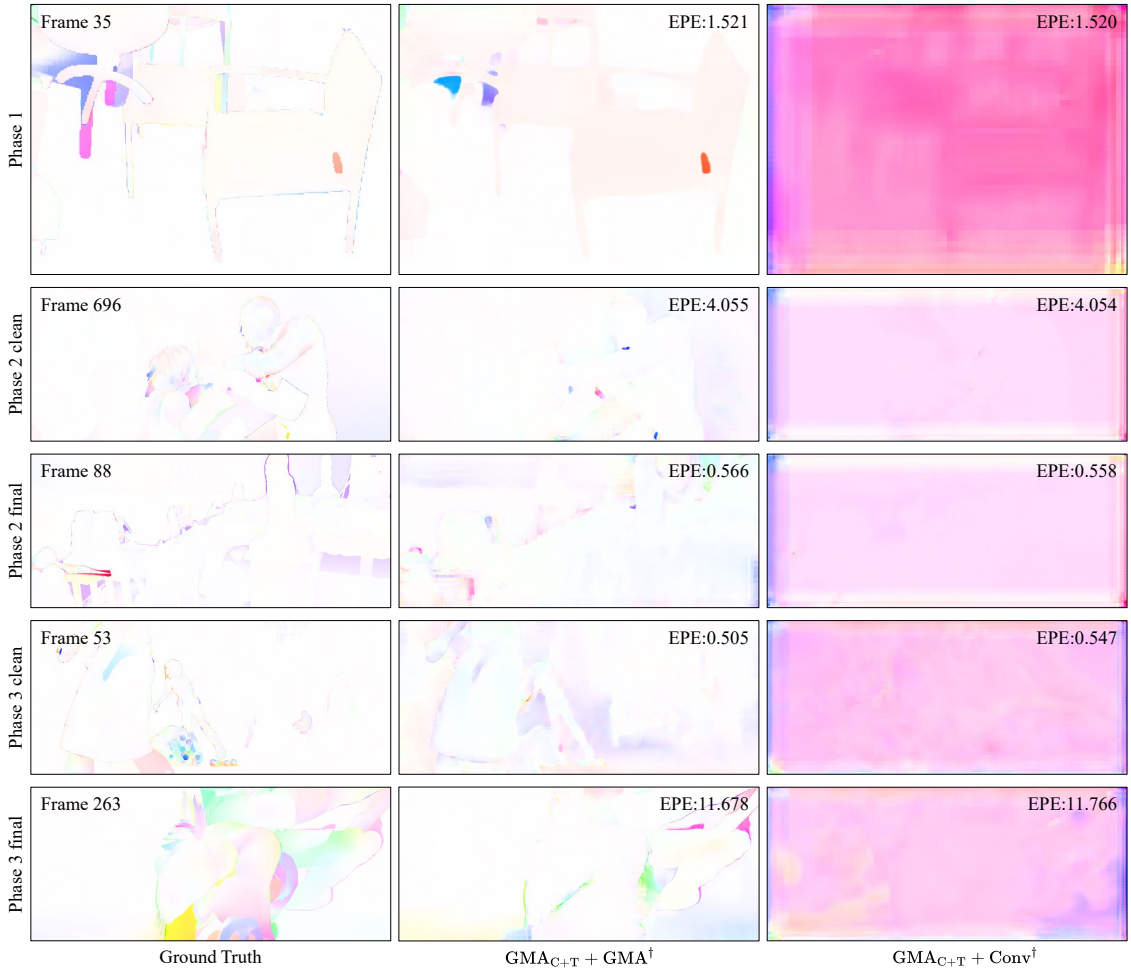


Figure 4.5: Visual comparison of estimated flow error generated by the Single-iteration Conv and the GMA-like error estimation networks with respect to the ground truth flow error during all training phases. The estimated flow on the middle column is generated by GMA$_{C+T}$ + GMA†; The estimated flow on the right column is generated by GMA$_{C+T}$ + Conv†.

To verify our hypothesis, we conducted the following experiments: Taking the combination RAFT$_{C+T}$ + RAFT* as the baseline experiment, we conducted three additional experiments, namely the flow estimation network RAFT$_{C+T}$ and the RAFT-like error estimation network that shares the same parameters of the encoders (two feature encoders and the context encoder) and/or the upsampler as their counterparts in the flow estimation network. The RAFT*-fc error estimation network embeds the RAFT-like architecture and shares the parameters of feature and context encoders of the flow estimation network RAFT$_{C+T}$. The error estimation network RAFT*-fc differs from the baseline network RAFT* in that the parameters of the feature and context encoders are frozen throughout the training process. Simply put, "-fc" represents that the error estimation network shares the same upsampler as the corresponding flow estimation network RAFT$_{C+T}$. When the error estimation network shares the same feature and context encoders as the flow estimation network, the constructed all-pairs correlation pyramids of the flow and error estimation networks

are also identical. Thus the look-up operation retrieves the look-up cost from the same 4D cost volume. Similarly, "-u" refers to the error estimation network that adopts the same upsampler's parameter of the upsampler as the flow estimation network $\text{RAFT}_{\text{C+T}}$; While "-fcu" indicates that the error estimation network shares not only the same encoders but also the identical upsampling mask to $\text{RAFT}_{\text{C+T}}$. All these error estimation networks are trained following the same training schedule, namely FlyingChairs2 (C) → FlyingThings3D(T) → Sintel-split(S-split), and with the identical RAFT training settings so they are marked with a superscript "*".

The experiments on the combinations of the flow estimation network $\text{GMA}_{\text{C+T}}$ and the varieties of the error estimation network $\text{GMA}^\dagger$ are also denominated in the same manner. This means that the error estimation networks $\text{GMA}^\dagger$-fc, $\text{GMA}^\dagger$-u and $\text{GMA}^\dagger$-fcu are additionally trained and evaluated following the GMA training procedure and settings.

Table 4.8 implicates the importance of adopting learnable encoders and upsamplers instead of sharing the same ones of the flow estimation networks in phase 1. For inference on FlyingChairs2, the error estimation network $\text{RAFT}^\star$ and $\text{GMA}^\dagger$ combined with the corresponding flow estimation networks eclipse all their varieties respectively in the accuracy of the estimated flow error in both the non-occluded and occluded regions.

Flow	Error	Type	Training Progress		Rel. Improv.	Flow	Error	Type	Training Progress		Rel. Improv.
			5K	Phase 1					5K	Phase 1	
$\text{RAFT}_{\text{C+T}}$	$\text{RAFT}^\star$	All	1.136	<u>1.049</u>	+7.59%	$\text{GMA}_{\text{C+T}}$	$\text{GMA}^\dagger$	All	1.159	<u>1.052</u>	+9.19%
		Noc	0.894	<u>0.826</u>	+7.57%			Noc	0.913	<u>0.827</u>	+9.40%
		Occ	5.103	<u>4.712</u>	+7.66%			Occ	5.185	<u>4.740</u>	+8.58%
	$\text{RAFT}^\star$-fc	All	1.145	1.066	+6.83%		$\text{GMA}^\dagger$-fc	All	1.148	1.072	+6.61%
		Noc	0.902	0.837	+7.27%			Noc	0.902	0.842	+6.65%
		Occ	5.118	4.834	+5.56%			Occ	5.182	4.844	+6.51%
	$\text{RAFT}^\star$-u	All	1.137	1.055	+7.18%		$\text{GMA}^\dagger$-u	All	1.165	1.059	+9.07%
		Noc	0.895	0.829	+7.34%			Noc	0.919	0.832	+9.41%
		Occ	5.105	4.761	+6.74%			Occ	5.193	4.774	+8.07%
	$\text{RAFT}^\star$-fcu	All	1.156	1.071	+7.32%		$\text{GMA}^\dagger$-fcu	All	1.160	1.074	+7.37%
		Noc	0.913	0.840	+7.99%			Noc	0.913	0.844	+7.60%
		Occ	5.133	4.858	+5.36%			Occ	5.195	4.847	+6.70%

Table 4.8: Loss error of $\text{RAFT}_{\text{C+T}}$ + $\text{RAFT}^\star$ and $\text{GMA}_{\text{C+T}}$ + $\text{GMA}^\dagger$ with their varieties of shared encoders and/or upsamplers on FlyingChairs2 in phase1. The best result of each group is underlined.

For phase 2, the comparison of the flow error predicted by the combinations $\text{RAFT}_{\text{C+T}}$ + $\text{RAFT}^\star$ and $\text{GMA}_{\text{C+T}}$ + $\text{GMA}^\dagger$ with their corresponding varieties are summarised in Table 4.9 and Table 4.10 respectively. In this regard, it is worth noting that error estimation networks with their own encoders and upsamplers establish the most accurate results on FlyingThings3D.

What is striking about the figures in Table 4.10 is that error estimation networks with learnable encoders and upsamplers are no longer the most accurate ones among their varieties. When it comes to the flow estimation network $\text{RAFT}_{\text{C+T}}$, $\text{RAFT}^\star$-fc outperforms all its varieties in AEPE on the clean pass of the Sintel training split; $\text{RAFT}^\star$-u acquires the best results on the final pass. However, in the right part of the table, the error estimation network $\text{GMA}^\dagger$ is on par with its varieties on both passes of the Sintel-train.

Data Type Things	Flow	Error	Training Progress C	Training Progress C+T	Rel. Improve.	Flow	Error	Training Dataset C	Training Dataset C+T	Rel. Improve.
clean	$\text{RAFT}_{\text{C+T}}$	RAFT*	14.781	<u>14.146</u>	+4.30%	$\text{GMA}_{\text{C+T}}$	GMA†	14.004	<u>14.002</u>	+13.64%
		RAFT*-fc	14.390	14.153	+1.65%		GMA†-fc	14.656	14.012	+4.40%
		RAFT*-u	14.864	14.152	+4.79%		GMA†-u	16.427	14.006	+14.74%
		RAFT*-fcu	14.361	14.154	+1.44%		GMA†-fcu	14.442	14.010	+2.99%
final	$\text{RAFT}_{\text{C+T}}$	RAFT*	14.004	<u>13.624</u>	+2.71%	$\text{GMA}_{\text{C+T}}$	GMA†	14.495	<u>13.178</u>	+9.09%
		RAFT*-fc	13.887	13.631	+1.84%		GMA†-fc	14.411	13.186	+8.50%
		RAFT*-u	14.118	13.628	+3.47%		GMA†-u	14.618	13.180	+9.84%
		RAFT*-fcu	13.839	13.631	+1.50%		GMA†-fcu	14.102	13.180	+6.50%

Table 4.9: Loss error of $\text{RAFT}_{\text{C+T}}$ + RAFT* and $\text{GMA}_{\text{C+T}}$ + GMA† with their varieties of shared encoders and/or upsamplers on FlyingThings3D in phase2. The best result of each group is underlined.

Data Type Sintel	Flow	Error	Type	Training Progress C	Training Progress C+T	Rel. Improve.	Flow	Error	Type	Training Progress C	Training Progress C+T	Rel. Improve.
clean	$\text{RAFT}_{\text{C+T}}$	RAFT*	All	1.750	1.479	+15.50%	$\text{GMA}_{\text{C+T}}$	GMA†	All	1.934	1.367	+29.34%
			Noc	0.885	0.662	+25.19%			Noc	1.028	0.620	+39.66%
			Occ	12.723	11.839	+6.95%			Occ	13.434	<u>10.839</u>	+19.31%
		RAFT*-fc	All	1.776	<u>1.471</u>	+17.16%		GMA†-fc	All	1.657	<u>1.366</u>	+17.57%
			Noc	0.914	<u>0.658</u>	+28.02%			Noc	0.879	0.620	+29.55%
			Occ	12.707	<u>11.786</u>	+7.25%			Occ	11.535	10.845	+5.98%
		RAFT*-u	All	1.687	1.472	+12.71%		GMA†-u	All	2.067	1.367	+33.84%
			Noc	0.823	0.659	+0.01%			Noc	1.107	0.620	+43.97%
			Occ	14.928	14.927	+0.01%			Occ	14.246	10.848	+23.86%
		RAFT*-fcu	All	1.687	1.472	+12.71%		GMA†-fcu	All	1.676	1.366	+18.49%
			Noc	0.823	0.659	+19.91%			Noc	0.864	<u>0.619</u>	+28.33%
			Occ	12.655	11.798	+6.771%			Occ	11.982	10.844	+9.50%
final	$\text{RAFT}_{\text{C+T}}$	RAFT*	All	2.941	2.761	+6.15%	$\text{GMA}_{\text{C+T}}$	GMA†	All	3.216	2.802	+12.88%
			Noc	1.799	1.610	+10.49%			Noc	2.089	<u>1.682</u>	+19.50%
			Occ	17.442	17.361	+0.47%			Occ	17.514	17.011	+2.87%
		RAFT*-fc	All	3.251	2.745	+15.55%		GMA†-fc	All	3.609	2.804	+22.31%
			Noc	2.011	1.595	+20.67%			Noc	2.408	1.687	+29.96%
			Occ	18.995	17.346	+8.68%			Occ	18.855	<u>16.987</u>	+9.90%
		RAFT*-u	All	2.853	<u>2.737</u>	+4.09%		GMA†-u	All	3.201	<u>2.801</u>	+12.48%
			Noc	1.699	<u>1.589</u>	+6.48%			Noc	2.043	1.683	+17.63%
			Occ	17.506	<u>17.304</u>	+1.16%			Occ	17.900	17.000	+5.03%
		RAFT*-fcu	All	2.985	2.739	+8.24%		GMA†-fcu	All	3.396	2.803	+17.46%
			Noc	1.810	1.590	+12.15%			Noc	2.200	1.685	+23.42%
			Occ	17.894	17.318	+3.22%			Occ	18.577	16.995	+8.51%

Table 4.10: Loss error of $\text{RAFT}_{\text{C+T}}$ + RAFT* and $\text{GMA}_{\text{C+T}}$ + GMA† with their varieties of shared encoders and/or upsamplers on Sintel training dataset in phase2. The best result of each group is underlined.

A striking feature of Table 4.11 is that error estimation networks with learnable encoders and upsamplers are again the most accurate of their varieties in phase 3. On the clean pass of Sintel-split, error estimation networks with their own encoders and upsampling masks achieve better results than their varieties. The shared upsampling mask can only slightly improve the results, but the gap is insignificant. For inference on the final pass, the only exception is the combination $\text{RAFT}_{\text{C+T}}$ + RAFT*-u, which outperforms all its varieties. When considering the combination $\text{GMA}_{\text{C+T}}$ + GMA†, the learnable encoders and upsampling masks can still promote the estimation on the final pass.

Data Type Sintel-split	Flow	Error	Type	Training Progress		Rel. Improve.	Flow	Error	Type	Training Progress		Rel. Improve.
				C+T	C+T+S-split					C+T	C+T+S-split	
clean	$\text{RAFT}_{\text{C+T}}$	RAFT*	All	2.025	1.791	+11.56%	$\text{GMA}_{\text{C+T}}$	GMA†	All	1.761	1.590	+9.71%
			Noc	1.047	0.943	+9.89%			Noc	0.920	0.863	+6.22%
			Occ	12.229	10.634	+13.634%			Occ	10.522	9.166	+12.89%
		RAFT*-fc	All	2.025	1.820	+10.10%		GMA†-fc	All	1.762	1.754	+0.46%
			Noc	1.046	0.955	+8.64%			Noc	0.920	0.934	+1.45%
			Occ	12.232	10.838	+11.40%			Occ	10.537	10.306	+2.20%
		RAFT*-u	All	2.029	1.806	+11.00%		GMA†-u	All	1.766	1.609	+8.87%
			Noc	1.049	0.937	+10.69%			Noc	0.923	0.858	+7.12%
			Occ	12.241	10.861	+11.27%			Occ	10.552	9.447	+10.47%
		RAFT*-fcu	All	2.027	2.066	-1.96%		GMA†-fcu	All	1.766	1.609	+8.89%
			Noc	1.047	1.111	-6.16%			Noc	0.923	0.958	+7.12%
			Occ	12.242	12.023	+1.79%			Occ	10.552	9.447	+10.47%
final	$\text{RAFT}_{\text{C+T}}$	RAFT*	All	3.166	3.178	-0.38%	$\text{GMA}_{\text{C+T}}$	GMA†	All	3.256	3.166	+2.78%
			Noc	2.017	2.061	-2.18%			Noc	2.071	2.223	-1.52%
			Occ	15.141	14.818	2.13%			Occ	15.579	14.216	+8.75%
		RAFT*-fc	All	3.189	3.239	-1.56%		GMA†-fc	All	3.255	3.344	-2.72%
			Noc	2.034	2.142	-5.28%			Noc	2.074	2.223	-7.20%
			Occ	15.228	14.678	+3.61%			Occ	15.568	15.024	+3.50%
		RAFT*-u	All	3.137	3.133	+0.10%		GMA†-u	All	3.257	3.232	+1.05%
			Noc	2.007	2.056	-2.44%			Noc	2.075	2.138	-3.03%
			Occ	14.918	14.371	+3.66%			Occ	15.580	14.531	+6.73%
		RAFT*-fcu	All	3.166	3.236	-2.20%		GMA†-fcu	All	3.434	3.352	+2.37%
			Noc	2.018	2.128	-5.43%			Noc	2.186	2.227	-1.85%
			Occ	15.135	14.790	+2.28%			Occ	16.443	15.092	+8.21%

Table 4.11: Loss error of $\text{RAFT}_{\text{C+T}}$ + RAFT* and $\text{GMA}_{\text{C+T}}$ + GMA† with their varieties of shared encoders and/or upsamplers on Sintel-split in phase 3. The best result of each group is underlined.

It turns out to be uncertain whether the error estimation networks should adopt the same encoders and/or upsamplers as the corresponding flow estimation networks. Thus we perform additional experiments on the combination RAFT_{C} + RAFT* and its varieties. Here the flow estimation networks are only trained with samples from FlyingChairs2 and the RAFT-like error estimation networks are also trained with FlyingChairs2 alone. The most obvious finding to emerge from the analysis in Table 4.14 is that shared feature and context encoders can enhance the accuracy of the estimated flow error. Adopting an identical upsampling mask to the flow estimation network could only slightly improve the result. These findings may be due to the fact that both flow and error estimation networks are fine-tuned on the same target datasets. Thus it is reasonable for the error estimation network to share the same encoders as the corresponding flow one.

However, another finding that stands out from Table 4.8 is that the error estimation should learn its own feature and context encoders as well as the upsampling masks when the flow and error estimation networks are not tuned with the same target dataset. It is also imperative to point out that the predicted flow error from the combinations in this ablation study is not as accurate as the one

generated by the combinations of cross cases. Take the flow estimation network RAFT_C and the error estimation networks with various architectures and training settings as examples. Despite the relatively better results achieved by $\text{RAFT}^\star$-fc with an AEPE of 0.867, the combination RAFT_C + $\text{RAFT}^\star$ achieved the most accurate prediction with an AEPE of 0.861 in Table 4.8, let alone $\text{RAFT}^{\star *}$ which is trained with extended training steps. The explanation of the extended training steps can be found in the following section.

4.6.2 Potential Improvement Methods

The two following ablated experiments are performed for further investigation, aiming to improve the performance of error estimation networks. The two potential improvement directions are: training the networks with more epochs and adopting a larger look-up radius for the look-up operation.

Look-up Radius

In light of the fact that the proposed error estimation networks provide less than satisfactory results in the third phase, one may ask: Is it because the networks do not look broad enough in the correlation pyramid? When performing a lookup, the look-up radius specifies the retrieval grid used in the search for the 4D cost volume. We conduct experiments on error estimation networks with increasing look-up radii in order to confirm our suspicions.

We only perform experiments on one combination of flow and error estimation network and the increasing look-up radius of 5, 6, and 7 because of limited time and hardware. We choose the combination of the flow estimation network $\text{RAFT}_\text{C+T}$ + $\text{GMA}^\dagger$ error estimation network. The evaluation results of this combination imply that the flow error can be learned on FlyngChairs2 in training phase 1, and the same case on FlyingThings3D and Sintel in training phase 2, while the accuracy slightly increases in training phase 3 when it is trained on Sintel-split. It is set to 4 for the look-up radius of the baseline error estimation network. Table 4.12 presents the results of these four error estimation networks. “Flow“ represents the flow estimation network, while “Error“ refers to the error estimation network. “-4“ refers to the error estimation network with a look-up radius of 4, “-5“, “-6“, and “-7“ following the same naming rule. All the hyperparameters and the architecture of the networks, except the look-up radius, are identical in these error estimation networks trained according to the GMA training settings and procedure (please refer to Section 4.2 for more details). Specifically, all our error estimation networks are trained with FlyingChairs2 (C) → FlyingThings3D(T) → Sintel-split(S-split), and evaluated on FlyingChairs2, Sintel training set, and Sintel-split respectively. “C“, “C+T“, and “C+T+S-split“ refers to the first, second, and third training phases.

In the analyses of data from Table 4.12, it is apparent that the error estimation networks $\text{GMA}^\dagger$ with a broader look-up range compare unfavorably with the baseline networks $\text{GMA}^\dagger$ with a look-up radius of 4 in all three training phases. Increasing the look-up radius of the error estimation network $\text{GMA}^\dagger$ does not significantly increase the accuracy of the estimated flow error as expected, while the AEPE for flow error is deteriorating. During the first phase of Chairs2, the AEPE (of flow error) of the baseline networks have no rivals, while the networks are evaluated on Sintel and Sintel-split during phase2 and phase3, the other three networks $\text{GMA}^\dagger$-5, $\text{GMA}^\dagger$-6, and $\text{GMA}^\dagger$-7 are still

Flow	Error	Training Dataset	Chairs	Sintel		Sintel-split	
				clean	final	clean	final
$RAFT_{C+T}$	$GMA^{\dagger}$-4	C	<u>1.021</u>				
	$GMA^{\dagger}$-5		1.032				
	$GMA^{\dagger}$-6		1.050				
	$GMA^{\dagger}$-7		1.050				
	$GMA^{\dagger}$-4	C+T		<u>1.480</u>	<u>2.721</u>		
	$GMA^{\dagger}$-5			1.487	2.726		
	$GMA^{\dagger}$-6			2.175	3.561		
	$GMA^{\dagger}$-7			1.482	2.756		
	$GMA^{\dagger}$-4	C+T+S-split				<u>1.727</u>	<u>3.064</u>
	$GMA^{\dagger}$-5					1.738	3.143
	$GMA^{\dagger}$-6					2.001	3.430
	$GMA^{\dagger}$-7					1.942	3.155

Table 4.12: Comparison of error estimation networks $GMA^{\dagger}$ with different look-up radii. "-4" refers to the error estimation network with a look-up radius of 4, "-5", "-6", and "-7" following the same naming rule. The superscript † refers to the GMA training setting and procedure introduced in Section 4.2. The best performance is underlined.

eclipsed by the baseline network $GMA^{\dagger}$-4 both on clean and final pass inference. Overall, these results indicate that we should adopt a look-up radius of 4 for error estimation networks, at least for the combination of $RAFT_{C+T}$ + $GMA^{\dagger}$.

Training Iterations

In regard to the second question, we would like to ask: Are the error estimation networks trained with enough epochs using the target datasets? To verify the suspicion above, we experiment with more epochs in the training of the error estimation networks, or more precisely, by doubling the training steps of each phase. Let us take the GMA training settings as an example. All the error estimation networks are trained with FlyingChairs2 (C) → FlyingThings(T) → Sintel-split(S-split) for 120,000 steps in each phase, and evaluated on FlyingChairs2, Sintel training set, and Sintel-split respectively. And the extended GMA training schedule is still following FlyingChairs2 (C) → FlyingThings(T) → Sintel-split(S-split) with 240,000 steps in each phase.

We compare the results of the combination of flow estimation network $RAFT_{C+T}$ with error estimation network $GMA^{\dagger}$ with the two training iterations mentioned above in Table 4.13. "Flow" represents the flow estimation network, while "Error" refers to the error estimation network. These error estimation networks are trained following the identical training schedule proposed in this chapter, and all the hyperparameters are the same as the GMA training settings except for the training steps. The superscript "†" represents the original GMA training schedule with training steps of 120,000 in each phase, while the superscript "$^{\dagger *}$" refers to the GMA training schedule with training steps of 240,000 in each phase, double as many as the original training schedule. "C", "C+T", and "C+T+S-split" refer to the first, second, and third phases of training.

Flow	Error	Training Dataset	Chairs	Sintel		Sintel-split	
				clean	final	clean	final
$RAFT_{C+T}$	$GMA^{\dagger}$	C	1.021				
	$GMA^{\dagger*}$		<u>0.963</u>				
	$GMA^{\dagger}$	C+T		1.480	2.721		
	$GMA^{\dagger*}$			<u>1.442</u>	<u>2.704</u>		
	$GMA^{\dagger}$	C+T+S-split				<u>1.727</u>	3.064
	$GMA^{\dagger*}$					1.792	<u>2.924</u>

Table 4.13: Comparison of the combinations of $RAFT_{C+T}$ + $GMA^{\dagger}$ with different training iterations. The superscript "†" represents the original GMA training schedule with training steps of 120,000 in each phase, while the superscript "†*" refers to the GMA training schedule with training steps of 240,000 in each phase, double as many as the original training schedule. The best performance is underlined.

As can be seen in Table 4.13, a positive correlation is found between the accuracy of the estimated flow error and training iterations. When the iterations in each phase are doubled, the AEPE (of flow error) increases by 5.6% in phase 1, and at the end of the second phase, the error estimation network $GMA^{\dagger*}$ achieved an improved AEPE (of flow error) of 1.442 on the clean pass of Sintel training split, while the AEPE (of flow error) on the final pass has already surpassed the best performance of all the error estimation networks performed in Section 4.4. For inference on Sintel-split, improved accuracy of estimated flow error can also be observed on final passes. The loss error of $GMA^{\dagger*}$ has improved by 4.6/Nevertheless, the network $GMA^{\dagger*}$ performs deterioratingly on the clean pass of Sintel-split than $GMA^{\dagger}$.

On account of the findings mentioned above, we also perform experiments on the combination of the flow estimation network $RAFT_C$ with various error estimation networks trained with extended iterations. Given the experiments in Section 4.6.2, the look-up radius of error estimation networks is still set to 4. These experiments are crucial to the survey in Chapter 5. The surveys on the Single-iteration one are discarded because of the ineffectiveness demonstrated in the aforementioned surveys. All the error estimation networks are only trained with FlyingChairs2 following the RAFT and GMA training settings, which are denoted with superscripts "*" and "†" respectively. For comparison, experiments with the extended version of these two training settings are also conducted. We denote them as "**" and "†*", which refer to the doubled training iterations of the settings. To be more precise, for the extended RAFT training settings, the networks are trained with 200,000 iterations instead of 100,000, while for the extended GMA training setting with 240,000.

The results of these experiments are compared in Table 4.14. What stands out in the table is the improved accuracy of the estimated flow error on Chairs2 of all the combinations compared to the results presented in Table 4.4. What emerges from the results reported here is that the quality of the estimated flow utilized in the error estimation networks also plays an important role in the accuracy of the estimated flow error. The flow estimation network we adopt here is $RAFT_C$ with an AEPE (of flow) of 0.867, which is tuned with the target dataset Chairs2 with an AEPE (of flow) of 0.867, dwarfing the adopted flow estimation networks $RAFT_{C+T}$in Table 4.4. The most striking observation to emerge from the data comparison is the quality of the estimated flow generated by the combination of $RAFT_C$ + $GMA^{\dagger*}$, which achieves an AEPE (of flow error) of 0.858.

Another interesting finding that stands out in the table is the performance of the error estimation network RAFT* and its variants. Although the shared feature and context encoders can improve prediction accuracy if we adopt the original RAFT training settings, the final results of these networks are on the same level if they are trained with double iterations. The accuracy of these error estimation networks is not as good as the one with cross cases which are also trained with extended iterations. This may indicate the limitations of the RAFT-like error estimation networks trained with RAFT training settings. In contrast, a more accurate estimation can be observed (AEPE=0.859) when the RAFT-like error estimation network is trained with extended GMA training settings.

Flow	Error	Training Dataset	Chairs		
			Occ	Noc	All
RAFT_C	RAFT*	C	3.994	0.686	0.876
	RAFT**		3.969	0.678	0.867
	RAFT*-fc	C	3.988	0.676	0.867
	RAFT**-fc		3.985	0.675	0.866
	RAFT*-u	C	3.995	0.686	0.876
	RAFT**-u		3.987	0.676	0.866
	RAFT*-fcu	C	3.988	0.676	0.867
	RAFT**-fcu		3.987	0.677	0.867
	RAFT†	C	3.995	0.686	0.876
	RAFT†*		3.950	0.670	0.859
	GMA*	C	3.996	0.688	0.878
	GMA**		3.944	0.672	0.860
	GMA†	C	3.950	0.673	0.861
	GMA†*		**3.915**	**0.671**	**0.858**

Table 4.14: Comparison of the combinations of RAFT_C with various error estimation networks trained with different iteration settings. The best performance is denoted with bold font.

4.7 Discussion

In this chapter, we train the proposed error estimation networks following the FlyingChairs2 → FlyingThings → Sintel-split schedule and evaluate the performance on the corresponding datasets. Together these inference results provide important insights into doable flow error estimation. In general, both error estimation networks with iterative update operators can perform their function in the first two phases. This is done by taking in only the image pairs and the estimated optical flow.

For phase 1 the most effective combination is $\text{RAFT}_\text{C+T}$ (flow) and GMA† (error). As for phase 2, the combination $\text{GMA}_\text{C+T}$ + GMA* and $\text{RAFT}_\text{C+T}$ + GMA* achieve performance for inference on the clean and final passes of Sintel respectively. However, not all combinations work as expected in phase 3. The combination of $\text{GMA}_\text{C+T}$ + GMA† and $\text{RAFT}_\text{C+T}$ + GMA* surpasses all the other combinations on the clean and final pass of Sintel-split respectively, whereas only a slight

improvement in the loss error can be observed. With the ablation studies, we can conclude that error estimation networks with shared encoders and/or upsamplers are a poor substitute for the ones embedding their own encoders and/or upsamplers when the error and flow estimation networks are not fine-tuned on the same datasets. The look-up radius of error estimation networks should be set to 4. Moreover, more training iterations instead of the originally adopted 100,000 or 120,000 iterations are beneficial to the quality of estimated flow errors.

One limitation of these methods however is that the above-mentioned ablation studies are not performed on all combinations of flow and error estimation networks due to limited computation ability and time space. Another limitation lies in the inadequate ablation studies on the training settings and the error estimation network architectures. A future direction for ablation studies could include a deeper correlation level or an adjusted loss function that can separate the flow error in the occluded and non-occluded regions.

The next part of the survey is concerned with the error-based semi-supervised optical flow estimation on FlyingChairs2 in Chapter 5.

5 Error-Based Semi-supervised Flow Estimation

In Chapter 4 we present the results of the proposed error estimation networks combined with the adopted flow estimation networks. Despite their preliminary characters, these findings clearly indicate that flow error can be estimated with the error estimation networks, among which the RAFT- and GMA-like ones achieve prominent performance, especially on the synthetic dataset FlyingChairs2. In light of this, it is logical to consider whether error estimation networks can be employed to boost the performance of the RAFT[TD20] flow estimation.

We utilize the estimated flow error to fine-tune the RAFT flow estimation network in the fashion of semi-supervised learning. The basic idea of our semi-supervised training schedule for the RAFT flow estimation network is as follows: We first pre-train the flow estimation network with the training samples of FlyingChairs2 only, then fine-tune the network with the validation samples, whose ground truth optical flow is replaced by the corresponding "estimated flow with offset". The estimated flow with offset is the sum of the estimated optical flow generated by the pre-trained RAFT and the estimated flow error produced by the selected error estimation network (Please refer to Section 5.1.2 for more details).

In the first part of this chapter, we introduce the experimental setup for the flow estimation as well as the implementation details. Two methods of applying the fine-tuning phase are elaborated on in this part. Next, the quantitative and qualitative results of the RAFT flow estimation networks, after fine-tuning on FlyingChairs2 validation samples, are demonstrated. Finally, a summary of the semi-supervised learning of optical flow estimation proposed in this chapter is provided. As with the experiments on the error estimation networks in Chapter 4, all the optical flow estimation networks presented in this chapter are also trained on the same GeForce RTX 3090 GPU with a memory size of 24 GB on `PyTorch` library [PGC+17] using the mixed precision strategy.

5.1 Experimental Setup and Implementation Details

5.1.1 Experimental setup

We choose the RAFT flow estimation network which has been trained with FlyingChairs2 (following the RAFT flow estimation training schedule provided by [TD20]) as the baseline experiment. Two different fine-tuning methods are proposed in this chapter:

Method 1

After completing the initial pre-train phase, an additional fine-tuning phase is implemented using validation samples from FlyingChairs2. During this phase, the flow estimation network is trained with validation samples labeled with the corresponding estimated flow with offset, as described in Section 5.1.2, or with mixed samples of validation and training samples from FlyingChairs2. Figure 5.1a illustrates how the learning rate varies during the pre-train and fine-tuning phases with method 1.

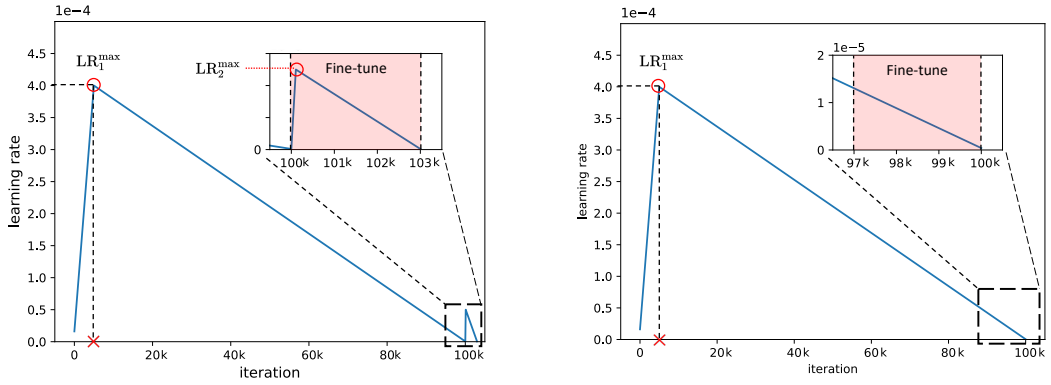
Two questions arise from this process: Firstly, how many additional iterations should be performed during the fine-tuning phase? For the baseline network, the flow estimation network is trained with 22,232 training samples for 100,000 iterations using a batch size of 10. This results in an epoch of 45. On the ground that the validation set of FlyingChairs2 consists of 640 samples, we propose setting the additional iterations to 3000. This will enable us to run the validation samples for 45 epochs using a batch size of 10.

Secondly, since the training is conducted under the same settings as RAFT, what should be the maximum learning rate (denoted as "LR_2^{max}") for the One-cycle Policy [ST19] during the fine-tuning phase? To answer this question, it is necessary to interpret how the One-cycle Policy functions during training. This policy involves training a model with a learning rate that increases linearly before decreasing linearly over one epoch, allowing deep learning models to converge faster and achieve better accuracy. Consequently, the initial learning rate increases to the set maximum learning rate LR_2^{max} and then from that maximum learning rate to some minimum learning rate in the fine-tuning phase. In the pre-train phase, the maximum learning rate LR_1^{max} is set to 0.0004, which means the learning rate is decreased to 0.000013 when there are still 3,000 iterations from the end of the first phase. Therefore, we set LR_2^{max} to 0.000013 in order to avoid the potential accuracy degradation attributable to the bump in the upcoming fine-tuning stage. Experiments with $LR_2^{max} = 0.0004$ (which is equal to LR_1^{max}) are also conducted.

Method 2

Instead of fine-tuning the flow estimation network during an additional phase after the first phase of training, we choose the last 3000 iterations of the original 100,000 iterations as the fine-tuning phase of our second method. For the first 997,000 iterations, the flow estimation network is trained following the training schedule with all the settings identical to the original code of RAFT[TD20], which is called the "pre-train phase", while the networks are trained with the fine-tuned dataset during the last 3,000 iterations without any modifications to the training setup. Method 2 does not require a re-start of the training progress with the learning rate first increasing to a set number and then declining. This may lead to a better quality of the estimated optical flow owing to the smooth learning rate schedule. Figure 5.1b illustrates how the learning rate varies with method 2.

The evaluation metric used in the following experiments is the Average End-point Error (AEPE), which is the mean pixel-wise flow error. The details of the average end-point error of the estimated flow field with respect to the corresponding ground truth flow field are elaborated in Section 2.2.



(a) Method 1 **(b)** Method 2

Figure 5.1: The learning rate schedules of the pre-train and fine-tune phases.

5.1.2 Implementation Details

All experiments have been conducted in the same setting as the official code of RAFT[TD20]. The differences are explained explicitly in the following paragraphs.

Training Loss

On account of using RAFT [TD20] as the flow estimation network, it is reasonable to adopt the weighted multi-iteration L1 loss function provided by RAFT[TD20] given the ground truth flow f_{gt} and the output estimated flow f_i of each GRU iteration.

$$\mathcal{L}_{\text{loss}} = \sum_{i=1}^{N} \gamma^{N-i} \left\| f_{gt} - f_i \right\|_1 , \tag{5.1}$$

where γ is the decay coefficient and N is the total GRU iteraions.

Estimated Flow with Offset

We have introduced in Section 2.3 that the FlyingChairs2 training set provides 22,232 image pairs as well as the corresponding ground truth optical flow. In this chapter, we denote the ground truth optical flow field $F_{gt}(I_{t,1}, I_{t,2}) \in \mathbb{R}^{2\times H\times W}$ between the two consecutive images $I_{t,1}, I_{t,2} \in \mathbb{R}^{3\times H\times W}$ from the FlyingChairs2 training set, where t denote training samples and W, H are the image width and height. Although we also have the ground truth optical flow of the validation samples of FlyingChairs2 on hand, we discard these correct labels during the fine-tuning phase to avoid overfitting. Therefore people would ask: is it possible to construct a flow that is believed to be close to the correct answer to the estimated flow, namely the ground truth optical flow? And this agrees with one of the motivations of this thesis. In Chapter 4, we draw the conclusion that the flow error

can be learned to some extent, especially in phase 1 where the error estimation networks are trained with the FlyingChairs2 dataset. In our opinion, it is possible to determine the right path to the ground truth optical flow through the use of the estimated flow error.

Given an image pair $I_{v,1}, I_{v,2} \in \mathbb{R}^{3\times H\times W}$ from the FlyingChairs2 validation set and a combination of a flow estimation network and an error estimation network, the flow estimation network can generate an estimated optical flow field $\text{Flow}_{es} \in \mathbb{R}^{2\times H\times W}$ and the error estimation network can produce an estimated flow error field $\text{Error}_{es} \in \mathbb{R}^{2\times H\times W}$ by taking in the image pair and the estimated optical flow. v denotes the image from validation samples. We define the sum of the estimated optical flow and the estimated flow error as the **Estimated Flow with Offset**, which is denoted as $\tilde{F}(I_{v,1}, I_{v,2}) \in \mathbb{R}^{2\times H\times W}$, $\tilde{F}(I_{v,1}, I_{v,2}) = \text{Flow}_{es} + \text{Error}_{es}$. Thus we can construct a "labeled" sample with two consecutive input images $I_{v,1}, I_{v,2}$ and the estimated flow with offset $\tilde{F}(I_{v,1}, I_{v,2})$, which performs the same function as the ground truth optical flow of the training samples in the supervision of training.

Figure 5.2 presents an example of the computation of the error estimation with offset. In respect to the pixel (i, j) in the flow field, the estimated flow (blue vector) generated by the flow estimation networks is denoted with $\text{flow}_{es} \in \mathbb{R}^2$, $\text{flow}_{es} = (u_{gt}, v_{gt})^T$. The predicted flow error (green vector)is denoted with $\text{error}_{es} \in \mathbb{R}^2$, $\text{flow}_{es} = (m_{es}, n_{es})^T$. As such, the estimated flow with offset denoted with the dashed black vector is computed as $(u_{es} + m_{es}, v_{es} + n_{es})^T$.

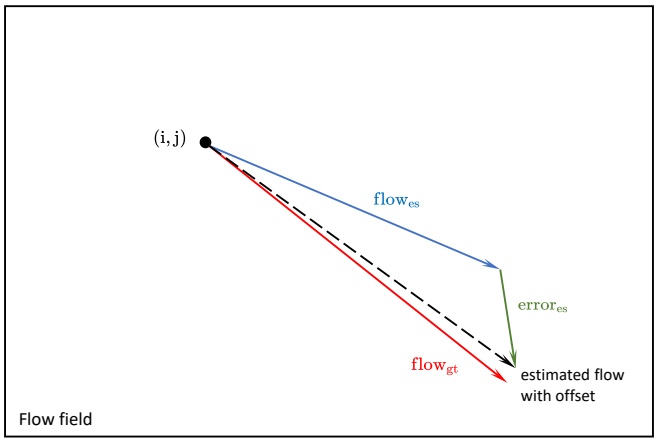
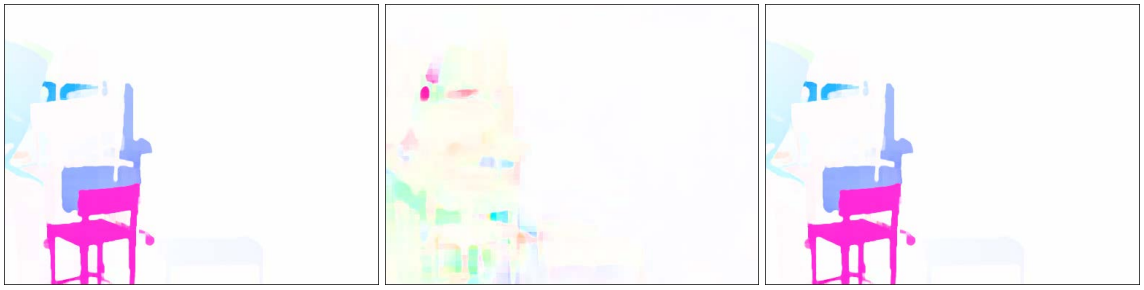


Figure 5.2: Illustration of computing the estimated flow with offset.

Figure 5.3 gives an example of the visualized estimated flow with offset in FlyingChairs2 validation samples. The flow estimation network RAFT_C generates the estimated optical flow in Figure 5.3a from the given image pair (Frame 530-1 and 530-2) and the estimated flow error in Figure 5.3b is produced by the error estimation network GMA* that takes the same estimated optical flow and the identical image pair as input.

Flow Estimation Network

We choose the RAFT_C instead of RAFT_{C+T} adopted in Chapter 4 since RAFT_C is trained only with the training set of the FlyingChairs dataset. Thus we can avoid or mitigate the underperformance of the flow estimation results from domain discrepancy. We also use RAFT_C as the baseline network



(a) Estiamted flow **(b)** Estimated flow error **(c)** Estimated flow with offset

Figure 5.3: The estimated flow, the estimated flow error, and the resulting estimated flow with offset (Frame 530).

in the following evaluations. This flow estimation network is trained with 100,000 steps with a batch size of 10 on the FlyingChairs2 training set, following the original training settings provided by RAFT[TD20].

Error Estimation Network

It is also essential to choose an appropriate error estimation network to provide a reliably estimated flow error so that the final estimated flow with offset is not too abbreviated from the ground truth flow. We have already run experiments on different combinations of error estimation networks with RAFT_C flow estimation network in Chapter 4. Table 4.14 shows the evaluation results of these error estimation networks on the FlyingChairs2 validation set. "AEPE" in this table refers to the average end-point error of the estimated flow error. It is apparent from this table that the combination of RAFT_C + $\text{GMA}^{\dagger *}$ achieves the most favorable results with an AEPE of 0.858 (denoted with bold font). Consequently, we choose this combination for our semi-supervised method of optical flow estimation to generate the required estimated flow with offset.

Training and Validation Dataset of the Fine-tuning Phase

Training Dataset of the Fine-tuning Phase: With the aim of error-based semi-supervised optical flow estimation in this chapter, we verify the application of the error estimation networks on semi-supervised learning of optical flow by training our RAFT flow estimation network with these validation samples. For the training set of FlyingChairs2, we still use the input image pairs and the ground truth flow provided by the dataset. As we explained before, we want to fine-tune the RAFT[TD20] flow estimation network also with the validation samples of the FlyingChairs2 dataset, which are named *validation-semi samples*. A validation-semi sample consists of two consecutive input images and the corresponding estimated flow with offset, which performs a similar function of ground truth flow during training to mitigate the overfitting issue. The entire validation-semi dataset refers to all 640 validation-semi samples. Furthermore, we also investigate the influence of different subsets of the validation-semi samples on the final performance of the fine-tuned flow estimation networks. Thus we also adopt three subsets of the entire validation-semi dataset, whose samples are selected based on the corresponding estimated flow error. For the validation-semi samples whose estimated flow error is among the top 160 best samples, they are named after 25best validation-semi

samples (denoted as "25best Val_C"). Similarly, those flow errors are among the top half, and the top 480 samples are called 50best and 75best validation-semi samples (denoted as "50best Val_C" and "75best Val_C") respectively.

The network was also fine-tuned using both samples from the training set and samples taken from the entire/subsets of validation semi-samples to investigate how they influence the accuracy of the estimated optical flow. In summary, the training samples of the fine-tuning phase are the entire/subsets of the validation-semi samples or a combination of the samples from the FlyingChairs2 training set and the entire/subsets of the validation-semi samples. Mathematically, the training samples adopted for the fine-tuning phase can be formulated as follows:

$$\text{Taining Dataset}_{\text{Finetune}} = \mathbf{x} \times \text{Train}_\text{C} + \mathbf{y} \times \text{Val}_\text{C}, \tag{5.2}$$

where x and y are two factors to adjust the different combinations of the training samples Train_C and entire/subset of validation-semi samples Val_C (Val_C can be 25best, 50best, 75best or entire validation-semi samples). Factor x is set to zero if we fine-tune the network only with validation-semi samples, while x is set to 1 if the combined fine-tune samples are adopted.

Validation Set of the Fine-tuning Phase: For the different training samples of the fine-tuning phase adopted in the experiments, we evaluate the flow estimation network also with different validation sets. All the networks are evaluated on the entire validation set of FlyingChairs2 (denoted as "Chairs-val"). In addition, the networks are also evaluated on the subsets of the validation set according to the training samples used in the fine-tuning phase. For example, when we use the 25best validation-semi samples as the training samples for fine-tuning, we adopt the 25best validation samples with the ground truth optical flow as the additional validation set. This validation set is called the 25best-subset of Chairs-val, where "Chairs-val" refers to the Chairs2 validation set. To be more precise, the quantitative analysis of the corresponding flow estimation networks will be evaluated on the entire validation set and the 25best validation set. Table 5.1 lists the training and validation datasets used in the entire training process.

Training Phase	Training Set	Validation Set
Pre-train	Train_C	Chairs-val
Fine-tune	25best Val_C (with Train_C)	Chairs-val 25best-subset of Chairs-val
	50best Val_C (with Train_C)	Chairs-val 50best-subset of Chairs-val
	75best Val_C (with Train_C)	Chairs-val 75best-subset of Chairs-val
	Val_C (with Train_C)	Chairs-val

Table 5.1: The overview of training and evaluation dataset of the entire training process. "Train_C" refers to the entire training samples of Chairs2; "Val_C" refers the entire validation samples of Chairs2; "Chairs-val-semi" refers to the training samples adopted in fine-tune phase

5.2 Evaluation

In this section, we present the evaluation results of the fine-tuned RAFT flow estimation networks in a semi-supervised fashion. For method 1, we choose the network which is only tuned with FlyingChairs2 training samples for 100,00 iterations (denoted with "Baseline^1") and the network which is trained with additional 3,000 iterations with FlyingChairs2 training samples alone following the setup of method 1 (denoted with "Baseline^2"), while only Baseline^1 is chosen as the baseline network for method 2 since no additional iterations are needed. Both baseline networks are evaluated with the FlyingChairs2 validation set as well as all of the subsets (25best, 50best, and 75best).

5.2.1 Quantitative Results

We initially test the effectiveness of fine-tuning the flow estimation network with method 1, which involves inserting an additional fine-tune phase after the original 100,000 iterations of the pre-train phase. Table 5.2 and Table 5.3 compare the summary statistics for the RAFT flow estimation network that are fine-tuned with validation-semi samples only or the mixed samples (training samples Train_C and validation-semi samples Val_C) in various proportions. The networks are evaluated with the entire validation samples and the corresponding subsets. The LR_2^{max} for the fine-tuning phase is set to 0.000013 and 0.0004 respectively.

What stands out in the tables is that: For FlyingChairs2, the semi-supervised training of the RAFT flow estimation networks with the aid of error estimation networks can improve the performance of RAFT trained in a supervised fashion alone. A closer inspection of the tables mentioned above indicates several interesting findings:

1. Comparing the results of Baseline^1 and Baseline^2, we can find that even if we fine-tune the network with training samples of Chairs2 alone for another 3,000 iterations, the network can still achieve favorable performance when LR_2^{max} is set to 0.000013. In contrast, if LR_2^{max} is relatively larger, for example, 0.0004, an degraded performance can be expected.

2. The network favors a lower LR_2^{max} during the fine-tuning phase. A bump in the learning rate during the additional 3,000 iterations may lead to worse estimations of optical flow. Moreover, if we compare the results in Table 5.2 and Table 5.3 with the same training samples and the same proportion, the results with the lower LR_2^{max} (in Table 5.2) always dwarf the corresponding results with the higher LR_2^{max}. As a result, we can come to the conclusion that the higher the LR_2^{max}, the higher the bump, and thus the inferior the results.

3. Fine-tuning the RAFT flow estimation network with the entire or the subsets of the validation-semi samples alone is not a good choice. Even when the LR_2^{max} is set to 0.000013, the improvement of the results is not in the same league as those which are fine-tuned with mixed samples. The network fine-tuned with the combined samples ($1 \times \text{Train}_\text{C} + 10 \times \text{Val}_\text{C}$) achieves an AEPE of 0.825. Those fine-tuned with combined samples in different proportions and with different subsets of validation-semi samples also surpass those only fine-tuned with validation-semi samples (The best of which only acquire an AEPE of 0.858).

4. In addition, the estimated flow can also be enhanced by using the entire validation-semi samples instead of its subsets. This may result from inadequate samples of the validation sets. This is a balance between the accuracy of the estimated flow with offset and the

quantity of the validation-semi samples. Although we can "supervise " the fine-tuning with these relatively accurate estimated flows with offset, which means they are more suitable to perform the function of the "ground truth optical flow", the quantity of these samples may still constrain the performance of the fine-tuned networks. Those samples in the subsets may not be sufficiently representative of all the properties of the motions that appear in the whole validation samples.

Another perspective on the inadequate validation-semi samples is that: the samples in the subsets of validation-semi samples are only chosen according to the loss error of the corresponding estimated flow error. Those samples with a lower error may be the relatively "easy" samples in the validation set. If we only fine-tune the networks with the subsets of the validation-semi samples, the networks do not perform well since they are only fine-tuned with those "easy" samples and may be tackled by those unseen "difficult" samples.

The influence of the proportion of the mixed fine-tuning samples is still unclear. Taking a look at Table 5.2, we can find that when the network is fine-tuned with mixed samples, it can achieve an AEPE of 0.825 if the fine-tuning samples consist of $1 \times \text{Train}_\text{C} + 1 \times 25\text{best-Val}_\text{C}$. If we turn to the mixed samples with 50best validation-semi samples, the best combination is $1 \times \text{Train}_\text{C} + 20 \times 50\text{best-Val}_\text{C}$ with an AEPE of 0.827. The most satisfactory performance for the mixed 75best validation-semi samples occurs when the combination is $1 \times \text{Train}_\text{C} + 15 \times 75\text{best-Val}_\text{C}$.

Table 5.4 demonstrates the results of the fine-tuned flow estimation networks with method 2. Compared with experiments shown in Table 5.4 and Table 5.2, what stands out in the tables is the well-matched performance of the networks fine-tuned with method 2 and with method 1. This finding confirms again the importance of applying a LR_2^{max} of 0.000013 to avoid a learning rate bump in method 1. Compared with the baseline experiments where the networks are trained with a training split of Chairs2 for the entire 100,000 iterations, the results of fine-tuned network surpass those of the baseline experiment even with the validation-semi samples alone. However, as can be seen from the table, adopting the mixed samples as the fine-tuning training samples is still superior to adopting the validation-semi samples alone. Taking the entire validation-semi samples for example, the fine-tuned network achieves an AEPE of 0.857 when taking only the validation-semi samples, while the networks fine-tuned with $1 \times \text{Train}_\text{C} + 5 \times \text{Val}_\text{C}$ can acquire a better accuracy of estimated flow with an AEPE of 0.826, which is an improvement of 4.7% with respect to the baseline experiment Baseline[1].

5.2.2 Qualitative Results

In Figure 5.4 we compare the results of the flow estimation networks qualitatively with some examples from the validation split of FlyingChairs2. For method 1, we select the fine-tuned RAFT with the samples of $1 \times \text{Train}_\text{C} + 10 \times \text{Val}_\text{C}$, and the LR_2^{max} is set to 0.000013. The visual results of the optical flow predicted by this network are marked with "Method 1". As for method 2, the fine-tuned RAFT with the samples of $1 \times \text{Train}_\text{C} + 5 \times \text{Val}_\text{C}$, which achieves superior performance among its competitors, is chosen as the representative flow estimation network. The visual results of the optical flow predicted by this network are marked "Method 2". In addition, we also present the consecutive input image pairs, the ground truth flow in occluded, non-occluded, and entire regions as well as the visual results of Baseline[1] (denoted as "RAFT_C") for comparison.

Training Dataset	factors		25best			50best			75best			full		
	x	y	Occ	Noc	All	Occ	Noc	All	Occ	Noc	All	Occ	Noc	All
Method 1: LR_2^{max} = **0.000013**														
Baseline[1]	-	-	0.978	0.165	0.181	1.237	0.265	0.295	1.811	0.386	0.446	4.004	0.676	0.867
Baseline[2]	1	0	1.003	0.173	0.190	1.250	0.269	0.300	1.809	0.388	0.448	3.965	0.672	0.862
25best-mix	0	1	1.008	0.172	0.190							4.061	0.678	0.872
	1	1	0.989	0.173	0.189							3.808	0.643	**0.825**
	1	5	0.979	0.173	0.189							3.884	0.646	0.832
	1	10	0.993	0.171	0.188							3.924	0.650	0.838
	1	15	0.984	0.172	0.189							3.833	0.646	0.829
	1	20	0.991	0.172	0.189							3.952	0.651	0.841
	1	25	0.977	0.171	0.188							3.858	0.644	0.829
	1	30	0.997	0.172	0.189							3.909	0.649	0.836
	1	35	0.985	0.172	0.189							3.935	0.649	0.838
	1	40	0.976	0.172	0.188							3.796	0.649	0.830
	1	45	0.982	0.171	0.188							3.937	0.650	0.839
	1	50	0.976	0.171	0.187							3.882	0.648	0.834
50best-mix	0	1				1.286	0.275	0.307				4.007	0.671	0.863
	1	1				1.228	0.266	0.296				3.888	0.650	0.837
	1	5				1.228	0.265	0.295				3.880	0.646	0.832
	1	10				1.217	0.265	0.295				3.864	0.648	0.833
	1	15				1.244	0.266	0.296				3.861	0.650	0.834
	1	20				1.244	0.275	0.305				3.751	0.649	**0.827**
	1	25				1.330	0.272	0.302				3.907	0.649	0.836
	1	30				1.228	0.266	0.296				3.843	0.648	0.831
	1	35				1.233	0.267	0.297				3.848	0.645	0.829
	1	40				1.231	0.267	0.297				3.867	0.650	0.835
	1	45				1.229	0.267	0.297				3.895	0.651	0.838
	1	50				1.216	0.267	0.297				3.934	0.651	0.839
75best-mix	0	1							1.822	0.401	0.461	3.928	0.675	0.862
	1	1							1.773	0.384	0.442	3.928	0.648	0.837
	1	5							1.770	0.384	0.443	3.843	0.644	0.828
	1	10							1.761	0.384	0.442	3.857	0.650	0.834
	1	15							1.776	0.386	0.444	3.837	0.645	**0.829**
	1	20							1.764	0.385	0.444	3.920	0.651	0.839
	1	25							1.778	0.385	0.444	3.857	0.647	0.831
	1	30							1.771	0.390	0.448	3.902	0.652	0.839
	1	35							1.775	0.388	0.446	3.861	0.650	0.835
	1	40							1.785	0.388	0.447	3.874	0.652	0.837
	1	45							1.781	0.392	0.450	3.849	0.652	0.836
	1	50							1.779	0.389	0.448	3.847	0.651	0.835
full-mix	0	1										3.874	0.675	0.858
	1	1										3.857	0.648	0.833
	1	5										3.862	0.647	0.829
	1	10										3.697	0.649	**0.825**
	1	15										3.807	0.649	0.831
	1	20										3.784	0.649	0.829
	1	25										3.816	0.654	0.835
	1	30										3.777	0.652	0.831
	1	35										3.783	0.652	0.832
	1	40										3.787	0.656	0.836
	1	45										3.787	0.654	0.834
	1	50										3.814	0.655	0.837

Table 5.2: Comparison of the fine-tuned RAFT flow estimation network with method 1. The LR_2^{max} during fine-tuning phase is set to 0.000013. "Factors" refers to the proportion of the training samples and validation-semi samples and the corresponding "x" and "y" are formulated in the same way as Equation (5.2).

Method 1: LR_2^{max} = **0.0004**													
Training Dataset	**factors**		**25best**			**50best**			**75best**			**full**		
	x	**y**	**Occ**	**Noc**	**All**	**Occ**	**Noc**	**All**	**Occ**	**Noc**	**All**	**Occ**	**Noc**	**All**
Baseline[1]	-	-	0.978	0.165	0.181	1.237	0.265	0.295	1.811	0.386	0.446	4.004	0.676	0.867
Baseline[2]	1	0	1.033	0.178	0.195	1.300	0.280	0.312	1.882	0.405	0.467	3.960	0.691	0.879
25best-mix	0	1	1.054	0.180	0.198							4.292	0.6783	0.984
	1	1	1.017	0.180	0.197							4.050	0.675	0.869
	1	5	1.013	0.181	0.198							4.007	0.676	0.868
	1	10	0.999	0.174	0.191							3.959	0.667	**0.856**
	1	15	1.022	0.179	0.196							3.850	0.675	0.858
	1	20	0.995	0.176	0.193							4.049	0.679	0.873
	1	25	0.996	0.178	0.195							3.893	0.672	0.857
	1	30	1.013	0.177	0.194							3.886	0.683	0.867
	1	35	1.002	0.175	0.192							4.028	0.676	0.868
	1	40	1.006	0.178	0.195							3.928	0.694	0.880
	1	45	1.026	0.179	0.196							3.957	0.693	0.881
	1	50	1.019	0.173	0.190							3.982	0.675	0.865
50best-mix	0	1				1.307	0.282	0.314				4.528	0.734	0.952
	1	1				1.268	0.276	0.307				4.107	0.674	0.871
	1	5				1.248	0.275	0.305				3.909	0.667	**0.853**
	1	10				1.256	0.276	0.307				3.971	0.680	0.870
	1	15				1.243	0.282	0.312				3.933	0.679	0.866
	1	20				1.261	0.276	0.307				4.031	0.677	0.870
	1	25				1.256	0.275	0.305				3.910	0.677	0.863
	1	30				1.267	0.276	0.306				3.962	0.683	0.872
	1	35				1.269	0.280	0.311				3.813	0.673	0.854
	1	40				1.250	0.276	0.306				3.890	0.684	0.869
	1	45				1.263	0.274	0.304				3.896	0.675	0.860
	1	50				1.253	0.276	0.306				3.956	0.680	0.868
75best-mix	0	1							1.867	0.412	0.473	4.225	0.717	0.919
	1	1							1.821	0.408	0.467	3.922	0.680	0.866
	1	5							1.835	0.399	0.460	3.760	0.671	**0.849**
	1	10							1.826	0.403	0.463	3.925	0.676	0.862
	1	15							1.839	0.402	0.462	3.904	0.679	0.864
	1	20							1.862	0.401	0.463	3.950	0.676	0.864
	1	25							1.824	0.402	0.462	3.959	0.677	0.866
	1	30							1.830	0.405	0.465	3.988	0.691	0.881
	1	35							1.829	0.401	0.462	3.905	0.680	0.866
	1	40							1.853	0.403	0.464	4.099	0.691	0.887
	1	45							1.860	0.404	0.466	4.088	0.682	0.878
	1	50							1.875	0.404	0.466	4.033	0.687	0.879
full-mix	0	1										4.052	0.706	0.899
	1	1										4.032	0.682	0.874
	1	5										3.982	0.680	0.870
	1	10										3.970	0.675	0.864
	1	15										4.096	0.683	0.879
	1	20										3.885	0.672	**0.857**
	1	25										3.873	0.682	0.865
	1	30										3.932	0.685	0.871
	1	35										4.055	0.686	0.880
	1	40										3.926	0.684	0.870
	1	45										3.955	0.684	0.873
	1	50										3.989	0.694	0.884

Table 5.3: Comparison of the fine-tuned RAFT flow estimation network with method 1. The LR_2^{max} during fine-tuning phase is set to 0.0004. "Factors" refers to the proportion of the training samples and validation-semi samples and the corresponding "x" and "y" are formulated in the same way as Equation (5.2).

Method 2														
Training Dataset	**factors**		**25best**			**50best**			**75best**			**full**		
	x	**y**	**Occ**	**Noc**	**All**	**Occ**	**Noc**	**All**	**Occ**	**Noc**	**All**	**Occ**	**Noc**	**All**
Baseline[1]	-	-	0.978	0.165	0.181	1.237	0.265	0.295	1.811	0.386	0.446	4.004	0.676	0.867
25best-mix	0	1	1.006	0.172	0.189							4.022	0.675	0.867
	1	1	0.985	0.173	0.189							3.827	0.648	0.831
	1	5	0.981	0.172	0.189							3.935	0.645	0.836
	1	10	0.991	0.172	0.189							3.930	0.649	0.838
	1	15	0.987	0.172	0.189							3.928	0.647	0.835
	1	20	0.980	0.172	0.189							3.823	0.645	**0.828**
	1	25	0.990	0.172	0.188							3.883	0.644	0.830
	1	30	0.980	0.171	0.188							3.920	0.649	0.837
	1	35	0.982	0.172	0.189							3.864	0.646	0.831
	1	40	0.977	0.172	0.188							3.799	0.649	0.830
	1	45	0.985	0.172	0.189							3.940	0.651	0.840
	1	50	0.979	0.172	0.188							3.938	0.648	0.837
50best-mix	0	1				1.269	0.277	0.308				4.001	0.673	0.865
	1	1				1.232	0.266	0.296				3.889	0.648	0.834
	1	5				1.238	0.266	0.296				3.970	0.651	0.842
	1	10				1.228	0.266	0.296				3.789	0.646	**0.827**
	1	15				1.226	0.265	0.295				3.921	0.651	0.839
	1	20				1.227	0.271	0.301				3.851	0.648	0.832
	1	25				1.234	0.266	0.296				3.893	0.648	0.835
	1	30				1.232	0.267	0.297				3.866	0.646	0.831
	1	35				1.234	0.267	0.297				3.864	0.645	0.830
	1	40				1.232	0.267	0.297				3.903	0.649	0.836
	1	45				1.230	0.272	0.302				3.895	0.650	0.837
	1	50				1.234	0.267	0.297				3.883	0.649	0.835
75best-mix	0	1							1.823	0.404	0.464	3.888	0.673	0.858
	1	1							1.775	0.385	0.444	3.871	0.647	0.832
	1	5							1.788	0.388	0.447	3.897	0.650	0.837
	1	10							1.772	0.384	0.443	3.921	0.650	0.838
	1	15							1.776	0.387	0.445	3.924	0.651	0.839
	1	20							1.767	0.385	0.443	3.865	0.647	**0.832**
	1	25							1.787	0.387	0.446	3.862	0.650	0.836
	1	30							1.779	0.387	0.446	3.891	0.650	0.836
	1	35							1.775	0.391	0.449	3.837	0.651	0.834
	1	40							1.788	0.388	0.447	3.844	0.650	0.834
	1	45							1.776	0.390	0.449	3.894	0.653	0.840
	1	50							1.786	0.388	0.447	3.870	0.652	0.837
full-mix	0	1										3.874	0.673	0.857
	1	1										3.893	0.647	0.834
	1	5										3.777	0.646	**0.826**
	1	10										3.811	0.652	0.834
	1	15										3.864	0.651	0.835
	1	20										3.813	0.649	0.831
	1	25										3.841	0.651	0.834
	1	30										3.779	0.652	0.832
	1	35										3.806	0.652	0.833
	1	40										3.792	0.656	0.836
	1	45										3.845	0.655	0.839
	1	50										3.790	0.654	0.834

Table 5.4: Comparison of the fine-tuned RAFT flow estimation network with method 2. "Factors" refers to the proportion of the training samples and validation-semi samples and the corresponding "x" and "y" are formulated in the same way as Equation (5.2).

The first example presented in the upper part of Figure 5.4 is Frame 530 of the validation set of Chairs2. The AEPE of the estimated flow generated by aseline[1] is 3.374. In contrast, the AEPE of the estimated flow generated by the fine-tuned network achieves better results of 0.940 and 0.982. The red circle in the figures in the upper part of Figure 5.4 demonstrates that the accuracy of the estimated optical flow in the non-occluded regions is improved after the RAFT flow estimation is fine-tuned with the help of the error estimation network GMA$^{\dagger *}$.

Let us take a look at the relatively "difficult" example in Figure 5.4: Frame 278. The results of Baseline[1] is 0.865. Both fine-tune methods show a promising improvement in the quality of the estimated optical flow by 46.5% (Method 1) and 41.5% (Method 2), especially of the motions in occluded areas (red circle in the lower part of Figure 5.4).

5.3 Discussion

In this chapter, we propose two methods to fine-tune the RAFT flow estimation network on FlyingChairs2 in a semi-supervised way. On account of the issue of overfitting, we use the estimated flow with offset to supervise the training during the fine-tuning phase instead of the ground truth flow. The estimated flow with offset is generated with the help of the error estimation network proposed and trained in the previous chapters.

The evaluation results suggest that both fine-tuning methods can boost the quality of the estimated flow compared to the RAFT flow estimation networks without fine-tuning with the validation set. It is important to note the benefit of adopting the combined training samples (training and the validation-semi samples of FlyingChairs2) as the training samples for the fine-tuning phase. We also favor the use of the entire validation-semi samples over its subsets. Further studies on the choice of $\mathrm{LR}_2^{\max}$ reveal the necessity of a lower value (for example 0.000013) to mitigate the accuracy degradation caused by the learning rate bump in the additional fine-tuning phase.

Although there are important discoveries revealed by these experiments, there are also limitations. There is uncertainty about the influence of the proportion of the mixed fine-tuning samples. Because of the workload and the limited time space, we do not perform semi-supervised flow estimation on GMA[JCL+21]. Another limitation is the target dataset. We only perform the experiments on the synthetic dataset FlyingChairs2. Further explorations may also focus on the performance of these flow-error-estimation-based fine-tuned methods on Sintel and KITTI-2015.

In Chapter 6 the summary and concluding remarks of this work are provided, along with the potential directions for future work.

Figure 5.4: Compariosn of visualized results of the estimated optical flow (Baseline).

6 Conclusion and Outlook

6.1 Conclusions and Limitations

The main goal of the current study is to investigate whether flow error can be learned in a supervised manner. We propose three architectures of error estimation networks, namely the Single-iteration Conv, RAFT-like, and GMA-like error estimation networks. The error estimation network takes in two consecutive images and the corresponding estimated flow error generated by a flow estimation network and outputs an estimated flow error field. Experiments on the combinations of flow estimation networks and various error estimation networks are performed. The error estimation networks are trained following the FlyingChairs2 → FlyingThings3D → Sintel-split schedule.

These experiments confirm that flow error can be learned in a supervised fashion in the first two training phases with RAFT-like and GMA-like error estimation networks. The loss error improves during the training process. The Single-iteration Conv error estimation networks fail to estimate the flow error in all three training phases. For phase 1, the most accurate combination is $\text{RAFT}_{\text{C+T}}$ + $\text{GMA}^{\dagger}$; While for phase 2, the combination of $\text{GMA}_{\text{C+T}}$ + $\text{GMA}^{\star}$ achieves the most accurate estimation on the clean pass of the Sintel training set, whereas $\text{RAFT}_{\text{C+T}}$ + $\text{GMA}^{\dagger}$ is the most favorable combination on the final pass; For phase 3, the best combination of the clean and final passes of Sintel-split are $\text{GMA}_{\text{C+T}}$ + $\text{GMA}^{\dagger}$ and $\text{RAFT}_{\text{C+T}}$ + $\text{GMA}^{\star}$ respectively. However, these results in phase 3 are not very encouraging. In this training phase, not all combinations can learn the flow error as expected. The major issue is the misleading estimation in non-occluded areas.

Although there are some important discoveries revealed by these studies, there are also limitations. One limitation of these methods however is that we do not perform all the ablation studies on the correlation look-up process of all the combinations due to restricted time. Furthermore, not all the ablation studies on the extended training iterations are performed. The Sintel-split dataset may not be perfectly divided into training and validation samples. The training samples of Sintel-split may not fully present all the motions that appear in the validation samples.

From the above discussion, the conclusion can be reached that RAFT-like and GMA-like error estimation networks can learn the flow error with supervised learning. The extended training steps can improve the accuracy of the estimated flow error to some extent.

We further investigate the effectiveness of error-based semi-supervised optical flow estimation with RAFT[TD20] on the FlyingChairs2 dataset. We propose two methods to fine-tune the flow estimation network. We construct the validation-semi samples by applying the estimated flow with offset to perform the function of ground truth flow during training. This enables the flow estimation networks to learn the motions of the validation samples without the concerns of overfitting. The experiments are limited by the lack of further investigation into the influence of

various proportions of training and validation-semi samples on the accuracy of the estimated optical flow. Notwithstanding these limitations, the study suggests that error estimation networks can be used to promote optical flow estimation in a semi-supervised manner.

6.2 Outlook

As for the error estimation networks, more research aiming to investigate the impact of each component of the networks and hyperparameters of training settings needs to be carried out. For example, the correlation level of the all-pairs correlation pyramid and the maximum learning rate are potential experimental directions.

Furthermore, there is abundant room for further progress in improving the architecture of the error estimation network. The visual results of the estimated flow error suggest that the flow error around the fine structure of objects is still a big issue. Inspired by FlowFormer[HSZ+22] and Transflow[LWM+23], one feasible method could be embracing the Transformer blocks in the CNN encoder and decoder structure to ameliorate the blurry edges of the structure boundaries.

Taking the misleading estimation in non-occluded areas and the improved accuracy in occluded areas in phase 3 into account, additional studies will be needed to explore an alternative training loss for error estimation networks. The training loss could conduct an additional occlusion detection to promote the error estimation of the occluded pixels and mitigate the deviated estimation of non-occluded pixels.

For the semi-supervised learning of optical flow with the aid of error estimation networks, further experimental investigations are needed to estimate the flow on other datasets, for example on Sintel. Furthermore, the estimated flow error is ideally beneficial for the flow estimation of unseen samples. This means that there is no pretrain on the target dataset. A natural progression of this work is to analyze the feasibility of error-based unsupervised optical flow estimation. This would also be a fruitful area for further investigation.

Bibliography

[AME+14] M. Aubry, D. Maturana, A. A. Efros, B. C. Russell, J. Sivic. "Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 3762–3769. DOI: 10.1109/cvpr.2014.487. URL: https://doi.org/10.1109%2Fcvpr.2014.487 (cit. on p. 14).

[BVS17] C. Bailer, K. Varanasi, D. Stricker. "CNN-based patch matching for optical flow with thresholded hinge embedding loss". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 3250–3259. DOI: 10.1109/cvpr.2017.290. URL: https://doi.org/10.1109%2Fcvpr.2017.290 (cit. on pp. 7, 9).

[BWSB12] D. J. Butler, J. Wulff, G. B. Stanley, M. J. Black. "A naturalistic open source movie for optical flow evaluation". In: *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VI 12*. Springer. 2012, pp. 611–625. DOI: 10.1007/978-3-642-33783-3_44. URL: https://doi.org/10.1007%2F978-3-642-33783-3_44 (cit. on pp. 13, 14, 33).

[DFI+15] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, T. Brox. "FlowNet: Learning optical flow with convolutional networks". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015. DOI: 10.1109/iccv.2015.316. URL: https://doi.org/10.1109%2Ficcv.2015.316 (cit. on pp. 7, 13, 14, 33).

[GLU12] A. Geiger, P. Lenz, R. Urtasun. "Are we ready for autonomous driving? the kitti vision benchmark suite". In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 3354–3361. DOI: 10.1109/cvpr.2012.6248074. URL: https://doi.org/10.1109%2Fcvpr.2012.6248074 (cit. on p. 15).

[GW16] D. Gadot, L. Wolf. "PatchBatch: A batch augmented loss for optical flow". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4236–4245. DOI: 10.1109/cvpr.2016.459. URL: https://doi.org/10.1109%2Fcvpr.2016.459 (cit. on p. 9).

[HCL06] R. Hadsell, S. Chopra, Y. LeCun. "Dimensionality reduction by learning an invariant mapping". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. IEEE. 2006, pp. 1735–1742. DOI: 10.1109/cvpr.2006.100. URL: https://doi.org/10.1109%2Fcvpr.2006.100 (cit. on pp. 7, 9).

[HHH+22] H.-P. Huang, C. Herrmann, J. Hur, E. Lu, K. Sargent, A. Stone, M.-H. Yang, D. Sun. "Self-supervised AutoFlow". In: *arXiv preprint arXiv:2212.01762* (2022). DOI: 10.48550/ARXIV.2212.01762. URL: https://arxiv.org/abs/2212.01762 (cit. on p. 13).

[HLL+22] Y. Han, K. Luo, A. Luo, J. Liu, H. Fan, G. Luo, S. Liu. "RealFlow: EM-Based Realistic Optical Flow Dataset Generation from Videos". In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XIX*. Springer. 2022, pp. 288–305. DOI: 10.1007/978-3-031-19800-7_17. URL: https://doi.org/10.1007%2F978-3-031-19800-7_17 (cit. on p. 13).

[HLVW17] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger. "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE, 2017, pp. 4700–4708. DOI: 10.1109/cvpr.2017.243. URL: https://doi.org/10.1109%2Fcvpr.2017.243 (cit. on p. 26).

[HSL16] Y. Hu, R. Song, Y. Li. "Efficient coarse-to-fine patchmatch for large displacement optical flow". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 5704–5712. DOI: 10.1109/cvpr.2016.615. URL: https://doi.org/10.1109%2Fcvpr.2016.615 (cit. on p. 9).

[HSZ+22] Z. Huang, X. Shi, C. Zhang, Q. Wang, K. C. Cheung, H. Qin, J. Dai, H. Li. "Flowformer: A transformer architecture for optical flow". In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVII*. Springer. 2022, pp. 668–685. DOI: 10.1007/978-3-031-19790-1_40. URL: https://doi.org/10.1007%2F978-3-031-19790-1_40 (cit. on pp. 7, 72).

[IMS+17] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, T. Brox. "Flownet 2.0: Evolution of optical flow estimation with deep networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2462–2470. DOI: 10.1109/cvpr.2017.179. URL: https://doi.org/10.1109%2Fcvpr.2017.179 (cit. on pp. 7, 13).

[ISKB18] E. Ilg, T. Saikia, M. Keuper, T. Brox. "Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 614–630. DOI: 10.1007/978-3-030-01258-8_38. URL: https://doi.org/10.1007%2F978-3-030-01258-8_38 (cit. on pp. 14, 15, 33).

[Jah18] A. Jahedi. *Improved descriptor learning for correspondence problems*. 2018 (cit. on pp. 7, 9).

[JCL+21] S. Jiang, D. Campbell, Y. Lu, H. Li, R. Hartley. "Learning to estimate hidden motions with global motion aggregation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9772–9781. DOI: 10.1109/iccv48922.2021.00963. URL: https://doi.org/10.1109%2Ficcv48922.2021.00963 (cit. on pp. 7, 8, 10, 11, 16, 19–21, 23–25, 28, 30, 31, 36, 37, 68).

[JLPK22] J. Jeong, J. M. Lin, F. Porikli, N. Kwak. "Imposing consistency for optical flow estimation". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2022. DOI: 10.1109/cvpr52688.2022.00318. URL: https://doi.org/10.1109%2Fcvpr52688.2022.00318 (cit. on p. 7).

[KNH+16] D. Kondermann, R. Nair, K. Honauer, K. Krispin, J. Andrulis, A. Brock, B. Gussefeld, M. Rahimimoghaddam, S. Hofmann, C. 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. 2016, pp. 19–28. DOI: 10.1109/cvprw.2016.10. URL: https://doi.org/10.1109%2Fcvprw.2016.10 (cit. on p. 16).

[LH17] I. Loshchilov, F. Hutter. "Fixing weight decay regularization in adam". In: abs/1711.05101 (2017). URL: http://arxiv.org/abs/1711.05101 (cit. on p. 37).

[LWM+23] Y. Lu, Q. Wang, S. Ma, T. Geng, Y. V. Chen, H. Chen, D. Liu. "TransFlow: Transformer as Flow Learner". In: *arXiv preprint arXiv:2304.11523* (2023) (cit. on p. 72).

[MES+21] E. Mohamed, M. Ewaisha, M. Siam, H. Rashed, S. Yogamani, W. Hamdy, M. El-Dakdouky, A. El-Sallab. "Monocular instance motion segmentation for autonomous driving: Kitti instancemotseg dataset and multi-task baseline". In: *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2021, pp. 114–121. DOI: 10.1109/iv48863.2021.9575445. URL: https://doi.org/10.1109%2Fiv48863.2021.9575445 (cit. on p. 7).

[MG15] M. Menze, A. Geiger. "Object scene flow for autonomous vehicles". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3061–3070. DOI: 10.1109/cvpr.2015.7298925. URL: https://doi.org/10.1109%2Fcvpr.2015.7298925 (cit. on pp. 12, 13, 15, 16).

[MIH+16] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, T. 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*. 2016, pp. 4040–4048. DOI: 10.1109/cvpr.2016.438. URL: https://doi.org/10.1109%2Fcvpr.2016.438 (cit. on pp. 13, 14, 33).

[PGC+17] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer. "Automatic differentiation in pytorch". In: (2017). URL: https://openreview.net/forum?id=BJJsrmfCZ (cit. on pp. 35, 57).

[REYE19] H. Rashed, A. El Sallab, S. Yogamani, M. ElHelw. "Motion and depth augmented semantic segmentation for autonomous navigation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 0–0. DOI: 10.1109/cvprw.2019.00049. URL: https://doi.org/10.1109%2Fcvprw.2019.00049 (cit. on p. 7).

[RHK17] S. R. Richter, Z. Hayder, V. Koltun. "Playing for benchmarks". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2213–2222. DOI: 10.1109/iccv.2017.243. URL: https://doi.org/10.1109%2Ficcv.2017.243 (cit. on p. 13).

[SCH15] M. Savva, A. X. Chang, P. Hanrahan. "Semantically-enriched 3D models for common-sense knowledge". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 2015, pp. 24–31. DOI: 10.1109/cvprw.2015.7301289. URL: https://doi.org/10.1109%2Fcvprw.2015.7301289 (cit. on p. 14).

[Sch21] J. Schäufele. *Improved RAFT architectures for optical flow estimation*. 2021 (cit. on pp. 7–9).

[SLG+19] L. Sevilla-Lara, Y. Liao, F. Güney, V. Jampani, A. Geiger, M. J. Black. "On the integration of optical flow and action recognition". In: *Pattern Recognition: 40th German Conference, GCPR 2018, Stuttgart, Germany, October 9-12, 2018, Proceedings 40*. Springer. 2019, pp. 281–297 (cit. on p. 7).

[ST19] L. N. Smith, N. Topin. “Super-convergence: Very fast training of neural networks using large learning rates”. In: *Artificial intelligence and machine learning for multi-domain operations applications*. Vol. 11006. SPIE. 2019, pp. 369–386. DOI: 10.1117/12.2520589. URL: https://doi.org/10.1117%2F12.2520589 (cit. on pp. 37, 58).

[SYLK18] D. Sun, X. Yang, M.-Y. Liu, J. 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*. 2018, pp. 8934–8943. DOI: 10.1109/cvpr.2018.00931. URL: https://doi.org/10.1109%2Fcvpr.2018.00931 (cit. on pp. 7, 11, 16, 21, 26).

[TD20] Z. Teed, J. 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*. Springer. 2020, pp. 402–419. DOI: 10.24963/ijcai.2021/662. URL: https://doi.org/10.24963%2Fijcai.2021%2F662 (cit. on pp. 7–11, 16, 17, 19, 21, 23–26, 28, 29, 35–37, 57–59, 61, 71).

[TD21] Z. Teed, J. Deng. “Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras”. In: *Advances in neural information processing systems* 34 (2021), pp. 16558–16569 (cit. on p. 7).

[VSP+17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017). URL: http://arxiv.org/abs/1706.03762 (cit. on pp. 19, 20, 30).

[WCWT17] S. Wang, R. Clark, H. Wen, N. Trigoni. “Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks”. In: *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2017, pp. 2043–2050. DOI: 10.1109/icra.2017.7989236. URL: https://doi.org/10.1109%2Ficra.2017.7989236 (cit. on p. 7).

[ZJB+22] Z. Zhang, P. Ji, N. Bansal, C. Cai, Q. Yan, X. Xu, Y. Xu. “CLIP-FLow: Contrastive learning by semi-supervised iterative pseudo labeling for optical flow estimation”. In: *arXiv preprint arXiv:2210.14383* (2022) (cit. on pp. 7, 9).

[ZUB18] H. Zhou, B. Ummenhofer, T. Brox. “Deeptam: Deep tracking and mapping”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 822–838. DOI: 10.1007/978-3-030-01270-0_50. URL: https://doi.org/10.1007%2F978-3-030-01270-0_50 (cit. on p. 7).

All links were last followed on May 2, 2023.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature