UKRAINIAN CATHOLIC UNIVERSITY

MASTER THESIS

Hidden state refinement for optical flow forecasting

Author:
Anton BABENKO

Supervisor:
Roman RIAZANTSEV

A thesis submitted in fulfillment of the requirements
for the degree of Master of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2023

Declaration of Authorship

I, Anton BABENKO, declare that this thesis titled, "Hidden state refinement for optical flow forecasting" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Master of Science

Hidden state refinement for optical flow forecasting

by Anton BABENKO

Abstract

In recent years the topic of optical flow has become well-spread due to computation power support and optical flow estimation applications used on mobile phones and edge devices: video editors, frame stabilizations, and autonomous driving feature providers. This work analyzes multiple approaches to optical flow estimation and finds the main problems of the optical flow methods: slow convergence and long execution of the prediction algorithm. We propose to solve the slow convergence and long execution time with hidden state refinement to provide the initialization for optical flow estimation based on several previous frames and their hidden state transformations, which imitates the pixel movement at the hidden state level. The proposed method uses CNN, LSTM, and Transformer blocks which help to achieve the optical flow estimation and hidden state refinement to speed up the system. We used Sintel, KITTY-15, FlyingChairs, FlyingThings, HD1K, DAVIS, and YouTube-VOS datasets for our experiment.

Acknowledgements

I would like to sincerely thank Roman Riazantsev, my supervisor, for his prodigious role in generating countless insightful hypotheses, ideas, and suggestions for the final project.

Also, thanks to ADVA Soft for their exceptional support. The provision of high-performance GPUs for our experiments was instrumental in accelerating our work and enhancing the efficiency of our computational tasks.

Lastly, I would like to pay a heartfelt tribute to my family, who have been my pillars of strength throughout this journey. Their moral support has given me the strength to persevere through challenges and setbacks.

Contents

List of Figures

List of Tables

List of Abbreviations

EPE	**E**nd-to-end **P**oint **E**rror
SOTA	**S**tate-**O**f-**T**he-**A**rt
CNN	**C**onvolution **N**eural **N**etwork
RNN	**R**ecurrent **N**eural **N**etwork
LSTM	**L**ong **S**hort-**T**erm **M**emory
GRU	**G**ated **R**ecurrent **U**nit
ConvGRU	**Conv**olutional **G**ated **R**ecurrent **U**nit
AGT	**A**lternate-**G**roup **T**ransformer
GPU	**G**raphics **P**rocessing **U**nit
RGB	**R**ed, **G**reen, **B**lue

Dedicated to the Armed Forces of Ukraine

Chapter 1

Introduction

Optical flow is a technique used in computer vision to estimate the motion of objects between video frames. The core idea is that optical flow tries to predict the pixel displacement for the whole scene. Hence, it is an essential tool for various applications, such as video compression, tracking, action stabilization, video restoration, self-driving, and depth estimation.

Deep learning has become essential to solving different tasks because high-speed GPU accelerators have become more accessible, and such methods outperform classical algorithmic ones. The standard solving method for optical flow also uses deep learning and outperforms such methods, as presented in Horn and Schunck [Horn and Schunck, 1981] or Lucas-Kanade [Lucas, Kanade, et al., 1981].

Result improvements were caused by the number of weights in the recent model and the use of components that can understand the pixel movements. Most recent methods [Bai et al., 2022; Huang et al., 2022; Jiang et al., 2021b; Teed and Deng, 2020; Zhang et al., 2021; Zhao et al., 2022] use the standard pattern to get decent results:

1. Convolutional neural networks (CNNs) help achieve quicker network generalization and add the frame context. They allow getting low and high dimensional features to perform further computation;

2. Correlation matrix generation is used as a contextual feature. The systems perform it as a matrix or scalar multiplication of several frames;

3. Long iterative use of the method ($\sim 5 - 100$ times) to generate only one optical flow result. This part commonly uses recurrent neural networks (RNNs) or custom blocks to train pixel shift detection.

Due to the number of inner iterations of the third point, we highlight that it is the system bottleneck and takes significant time to execute. Especially now when mobile phones and edge devices can execute a vast amount of deep neural network systems, but they need more resources to perform as personal computers or servers. Indeed, optical flow in mobile phones and edge devices has become an essential part of such systems as video editors, frame stabilization, and autonomous driving feature providers. Still, sometimes they cannot use the proposed relatively computationally expensive model and requires lightweight versions of an original method.

Combining recent optical flow estimation models and edge devices, we can set the critical points:

1. Some system parts require a lot of computation power or execution time;

2. Models have slow convergence due to the complexity of the optical flow task.

The recent state-of-the-art methods (RAFT [Teed and Deng, 2020], DEQFlow [Bai et al., 2022], GMA [Jiang et al., 2021b], FlowFormer [Huang et al., 2022]) generate a

cost volume, which is the multiplication of two frames features. This method helps to add the context of possible pixel movements. Indeed, with the active use of transformer architecture for computer vision tasks, the method uses them for different computation parts in the optical flow task: feature extraction, cost volume, updating blocks, and others. FlowFormer [Huang et al., 2022] and GMFlowNet [Zhao et al., 2022] are the best representations of attention use.

Currently, many networks (DEQFlow, GMA, FlowFormer) use the RAFT method as the initial point and add new tricks or blocks which change architecture slightly to achieve better results. We also use RAFT-like architectures as the initial point because they achieve state-of-the-art results according to the endpoint error metric (EPE), especially FlowFormer and DEQFlow ones. The FlowFormer method uses transformer blocks to process the key RAFT components (cost volume function results and update block converge support). The DEQFlow method uses a custom optimization framework based on the initial RAFT architecture. We combine FlowFormer and DEQFlow approaches to use the transformer-based methods instead of the pure RAFT to achieve better convergence than the previous works and add pixel movement prediction as a custom initialization of the update block to test that with such a technique:

1. We can speed up the training and model evaluation achieving the same results as the original FlowFormer does without significant architecture change;

2. The hidden state refinement based on standard blocks (RNNs, LSTMs, CNNs, Transformers) can work as fast as the DEQFlow framework for model training and evaluation.

The idea of hidden state refinement follows that we expect linear movement changes in sequential datasets. FlowFormer supports warm-up training, where the following image pair uses the previous optical flow result to initialize the update block. This technique gives better metric results, unlike without it. Hence, it proves that reusing sequential data and previous optical flow predictions can use object movement information to achieve better results.

As an alternative for the FlowFormer and DEQFlow combination, we propose two hidden state refinement systems based on FlowFormer architecture and sequential movement forecasting. We estimate the next hidden state and optical flow based on the previous ones one step ahead before using the time-consuming update block. These forecasters have compact architectures, which help to reduce the update block iterations both in the train and test stages, which solves one of the main bottlenecks of the system and allows it to converge quicker as it uses the information from the previous frames.

For our experiments, we have used KITTI-2015 [Geiger et al., 2013] and SINTEL [Butler et al., 2012], FlyingChairs [Dosovitskiy et al., 2015], FlyingThings [Mayer et al., 2016], and HD1K [Kondermann et al., 2016] datasets, the standard ones for optical flow tasks, to compare results correctly. These datasets have a limited number of sequential data. Hence we additionally have used DAVIS [Pont-Tuset et al., 2017] and YouTube-VOS [Xu et al., 2018] datasets which contain long sequences of the frames.

Chapter 2

Related work

We now review prior work on our method, focusing on neural-network-based optical flow, iterative optical flow, and optical flow optimization.

2.1 Neural-network-based optical flow

With the advancement of deep learning support, we see the progressive use of neural networks to optimize different complex tasks, and optical flow estimation is no exception. Here we have many examples of coarse-to-fine pyramidic network usages [Dosovitskiy et al., 2015; Ilg et al., 2017; Hui, Tang, and Loy, 2018; Hui, Tang, and Loy, 2021] that generate features from image pairs and predict pixel displacement.

The FlowNet defined several ways to compute images: stack them into one tensor and pass them through the network, or extract features from each image separately and concatenate them to continue end-to-end training. The second approach became the standard practice for the following approaches because it gave better feature representation and metric results. This network was hard to train due to slow convergence and dataset size. FlowNet2 tried to solve both problems using dataset schedulers, a stack of several neural networks, and a warping method. However, this network took much time to train due to a convergence problem, and several internal networks made FlowNet2 a large one.

PWC-Net [Sun et al., 2017] was the next step for the optical flow estimation. This network had trainable feature encoders, a warping layer, and a cost volume layer, adding extra information about pixel movement. Also, PWC-Net was s 17 times smaller and easier to train than the FlowNet2 model. The cost volume function was the correlation between the first image features and the second image warped features.

2.2 Iterative optical flow

After the success of the cost volume layer, papers started to use and modify it. RAFT (Fig. 2.1) is a paper in which the cost volume function was modified: it contained several scales and 4D tensors, unlike PWC-Net 2D ones. The cost volume function was one of several things that increased the method accuracy. They also used the ConvGRU block, which iteratively predicted and corrected the optical flow estimation. This block iteratively worked and generated more sophisticated optical flow results. There are several reasons why iterative blocks can be beneficial:

1. Refinement. Iterative blocks allow the flow estimates to be refined over multiple iterations. This can help reduce errors and noise in the estimates, resulting in more accurate and reliable flow estimates.

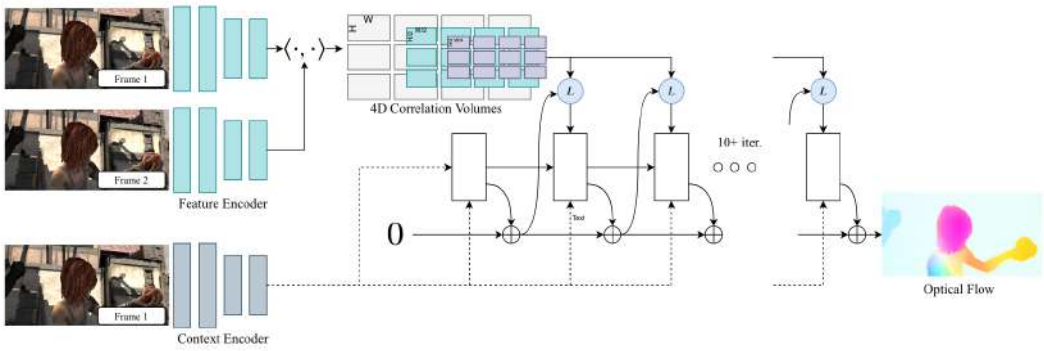

FIGURE 2.1: RAFT architecture(the image from [Teed and Deng, 2020]).

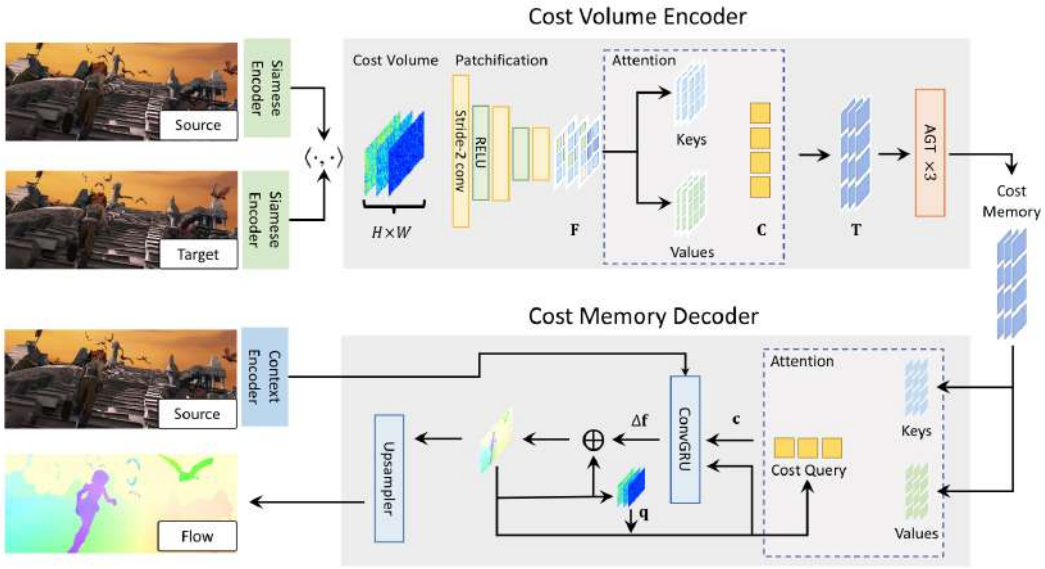

FIGURE 2.2: FlowFormer architecture (the image from [Huang et al., 2022]).

2. Regularization. Iterative blocks can also help to regularize the flow estimates, preventing the estimates from becoming too noisy or unstable.

3. Handling occlusions. Iterative blocks can help to handle better occlusions, which are areas in the image where one object occludes another, and the flow cannot be accurately estimated. By refining the estimates over multiple iterations, iterative blocks can better handle occlusions and improve the overall accuracy of the flow estimates.

The FlowFormer (Fig. 2.2) architecture upgraded the RAFT model, which contained the alternate-group transformer (AGT) layers in a novel latent space, and decodes the cost memory via a recurrent transformer decoder with dynamic positional cost queries. The work core points were to reduce the flow leakage around object boundaries, process details more precisely, and predict the hidden movement of objects: the objects could be covered by something, but even in the next appearance, the optical flow method worked fine.

The authors used attention blocks to predict the object movement to achieve such results. Our work also predicts the movement, but in our case, we predict the pixel

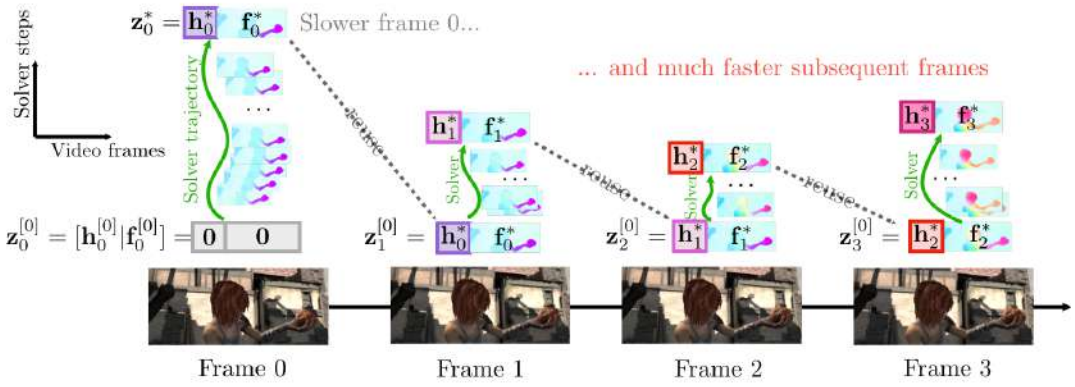

FIGURE 2.3: DEQFlow sequential fixed point reuse (the image from [Bai et al., 2022)].

movements(not as the whole object) based on previous pair representations and hidden states.

Paper with transformer architectures [Huang et al., 2022; Zhang et al., 2021; Zhao et al., 2022] usually add attention mechanism to transform features to cost volume and after decode with it. The attention usage helps add extra context and filter needed data from the cost volume layer while ignoring the layer massive noise.

2.3 Optical flow optimization

After the RAFT architecture success, it became the common starting point because the results were great, the code was refactored well, and it was easy to modify. Nonetheless, the RAFT model has several disadvantages:

1. 4D cost volume is too large, involving noisy data, which can complicate model calculations accuracy or require much computation power to train layers to understand this extra data.

2. The recurrent blocks use many iterations($\sim 5 - 100$) to achieve decent results. This behavior is another bottleneck of the system.

The paper [Jiang et al., 2021a] tried to solve the problem of the dense 4D cost volume. The key idea is to perform K-nearest neighbors to transform such data into a sparse one. Results showed that data size significantly decreased, and the metrics game had nearly the same results as the dense data presented.

The DEQFlow paper [Bai et al., 2022] tried to solve another problem of the system speed-up. They used previous hidden states to start model execution from them(Fig. 2.3) because we have an assumption that all our images were in the action sequence. They also used Newton [Byrd et al., 2016], Broyden [Broyden, 1965], and Anderson [Anderson, 1965] solvers to optimize the recurrent network, which was modified to the single-layer network. Our method combines the DEQFlow implementation and the FlowFormer (RAFT-based) approach. However, in our case, we want to add the pixel movement information via a custom recurrent network and initialization technique and use CNNs, RNNs, LSTMs, and Transformers.

Chapter 3

Background

As a result, we have several ways to research the optical flow speed-up: classical algorithms, coarse-to-fine pyramidical networks, recurrent blocks with a cost volume function, transformers, and custom optimization approaches.

Classical algorithms. We will consider the following algorithms: Lucas Kanade and Horn–Schunck methods. They are simple to code and work even on embedded devices with a good speed, but they are not accurate and lose the object pixel displacement quickly.

Coarse-to-fine pyramidical networks. Papers presented coarse-to-fine pyramidical network methods to improve the classical optical flow estimation accuracy. They were done with CNN usage and contained more parameters to be flexible and understand the frame movement better. Such systems took much time to train to meet their convergence, and the dataset selection was a crucial part of the process.

Recurrent blocks and cost volume function. This method solved the long convergence problem of the optical flow task. The pyramidical system helped to collect more sufficient features on different image scales, and the cost function added the image context and the direction of possible pixel displacement. On top of cost function and features, the recurrent blocks tried to estimate the best optical flow image. Recurrent blocks helped decode features and solved the task via small iterations instead of just the coarse-to-fine approach, where significant shifts were used immediately. Indeed, the recurrent blocks solved the problem, but they added extra time to compute the pair of images.

Transformers. Using transformers for computer vision became a general behavior. The attention mechanism added extra context and skillfully decoded difficult areas of image pairs features. The extra context helped to achieve better results than simple pyramidical networks with recurrent blocks. Hence, new custom architectures with attention blocks became state-of-the-art approaches (FlowFormer, GMNet). Unfortunately, transformer networks became larger, and the execution time was raised.

Custom optimization approaches. Using different tricks and non-standard optimization methods (e.g., Newton, Broyden, and Anderson solvers) was quite beneficial. Such methods achieved outstanding results, but the core architectures are related to the initial RAFT version without any modifications.

Hence, we define the problems which we want to cover:

1. Apply the DEQFlow optimization approach to the state-to-the-art method (FlowFormer);

2. Speed up the evaluation of the method with movement prediction to initialize the optical flow without deterioration of the metrics.

Chapter 4

Problem setting and Approach to Solution

This section highlights the research questions, problem statements, hypothesis, and research plan.

4.1 Problem formulation

Following **chapter 1**, our research concentrates on exploring and addressing issues related to optical flow slow convergence and inference time. These problems are commonly encountered across the spectrum of contemporary, state-of-the-art methodologies. We have identified two central bottlenecks associated with optical flow forecasting:

1. The voluminous number of update block executions needed during both the training and testing stages, which not only significantly demand computational resources but also time.

2. A sluggish convergence rate of models that is largely attributable to the inherently intricate nature of optical flow tasks, creating a bottleneck in refining and obtaining timely results.

Our strategy to tackle these challenges employs the use of hidden state refinement models on sequential data. This approach permits the recycling and update of hidden states and optical flow results derived from preceding image pairs, thereby serving to:

1. Expedite the computational process of successive optical flow, thereby optimizing efficiency and resource usage;

2. Introduce external information into the system, which acts as an additional data input, leading to quicker convergence during the model training phase, hence accelerating the overall computational process.

Our research revolves around two pivotal components, providing two perspectives on optical flow speedup:

1. A fusion of the DEQFlow framework with the FlowFormer method. This combination can leverage the inherent strengths of both methodologies. DEQ-flow design has increased the speed of the RAFT approach, while the use of attention blocks has drastically increased the precision of the recurrent method.

2. Forecasting optical flow based on previous hidden state. Refining FlowFormer hidden state can provide learned warm-stared initialization of the recurrent network. In RAFT, DEQ, authors show that reusing old flow and hidden state improves the speed and precision. Learned representation of forecasted motion could further improve both measurements.

4.2 Research questions

Based on our problem statement, there are three main research questions.

Question 1. (Combination of DEQFlow and FlowFormer approaches). Can we speed up the optical flow estimation using the DEQFlow framework with the FlowFormer method?

Question 2. (Hidden state refinement for the following optical flow estimation). Can we speed up the optical flow estimation using the network on the previous hidden state of the optical flow pair of images, which will imitate pixel displacement inertia prediction as the initialization for the update block?

Question 3. (Quality of hidden state refinement). Can the hidden state refinement of the previous optical flow produce most of the refine for the following optical flow estimation?

4.3 Combination of DEQFlow and FlowFormer approaches

Problem statement. Given image pairs, use the DEQFlow framework with the FlowFormer method.

Hypothesis 1. A combination of DEQFlow and FlowFormer approaches can speed up the model convergence and execution time.

DEQFlow and FlowFormer represent two state-of-the-art methods. Their respective characteristics allow for an amalgamation that can yield significant benefits.

Based on the distinct properties of these two methods, their combination allows for optimized utilization of each method's unique capabilities: use a better-performed model for optical flow estimation and allow to converge with DEQFlow methodology. It facilitates a more rapid functioning of the model system. This synthesis opens avenues for update block iterations reduction and better convergence.

4.4 Hidden state refinement for the following optical flow estimation

Problem statement. Given the sequence of the images (e.g., eight images), use the hidden state refinement of the previous optical flow estimation as the initialization for the following image pairs.

Hypothesis 2. The hidden state refinement network can imitate pixel displacement inertia prediction based on previous hidden states. The refinement network output is used to initialize the update block, which leads to system speed up.

In the context of optical flow estimation, The update block of the optical flow estimation is the most time-consuming part of the system. The significant part is often used in the recurrent part, which can be executed $\sim 5 - 100$ times. By refining and reusing information from prior computations, we aim to reduce the iterations and, thus, the computational burden of the update block executions.

4.5 Quality of hidden state refinement

Problem statement. Given the sequence of the images (e.g., eight images), use the hidden state refinement of the previous optical flow estimation as the initialization for the following image pairs with frozen update block weights.

Hypothesis 3. The hidden state refinement can handle most of the optical flow estimation process based on pixel motion understanding; the update block is only used to fix small image details.

We are confident that pixel motion understanding from the previous image pair results can add an essential part to the optical flow estimation with the hidden state refinement network and speed up the system execution by reducing the number of recurrent update part iterations.

4.6 Approach

To structure the hypothesis testing, we need a structured plan. We have used the following steps for it:

1. Problem formulation: Formulate questions and hypotheses.
2. Data understanding: Research the previous works.
3. System preparation: Propose the solutions to answer the questions and test the hypothesis.
4. Modeling: Implement the proposed solutions and run experiments.
5. Evaluation: Evaluate the experiments and estimate the impact.

4.7 Compare with other solutions

In pursuit of a comprehensive understanding of the effectiveness of our proposed solution, we have outlined a plan to compare it with the traditional approach of optical flow estimation, with particular emphasis on the utilization of sequential data. During our research, we found out that there are no papers dedicated to the use of sequential data. This lack could be attributed to the limited number of sequential datasets available for optical flow analysis.

We have chosen to test our solution on the Sintel dataset to substantiate our findings and establish a benchmark. It contains sequences and is the standard benchmark for the most state-of-the-art approaches. Our analysis is methodically designed to examine three key aspects:

1. End-Point Error (EPE) Comparison: We intend to compare the precision of our method against others by evaluating the EPE, a commonly used metric for assessing the precision of optical flow estimation methods;
2. Update Block Iterations and Time Consumption: An integral part of our evaluation involves examining the number of update block iterations our solution requires and the corresponding time taken compared to other methods. This analysis would provide insights into the computational efficiency of our solution;

3. Differences in Flow Estimation: We also aim to assess the qualitative differences in flow estimation between our proposed solution and the current state-of-the-art (SOTA) methods. This comparison will enable us to measure our approach distinct advantages or potential limitations.

By focusing our evaluation across these three domains, we aim to present a complete view of the performance of our proposed solution in the context of optical flow estimation with sequential data usage.

4.8 Summary

Upon considering the problem statement, we formulated three research questions that have been instrumental in shaping our research trajectory. The development of these queries prompted us to hypothesize that the employment of hidden state refinement could speed up the stages of model training and testing.

The primary basis for this hypothesis is the conjecture that hidden state refinement, by leveraging information from previous image pairs, can infuse the system with additional data. This supplemental information could enhance the system's ability to converge more efficiently and effectively. This approach may offer a novel means of addressing the prevailing challenges in the field, ultimately leading to more efficient and robust model performance.

Our hypothesis highlights the potential utility of past data use in the form of hidden state refinement. This methodology augments the computational process with historical context and paves the way for more rapid convergence due to the increased data inputs. Thus, this approach could significantly accelerate both the model train and test stages, thereby enhancing our system overall efficiency and effectiveness.

Chapter 5

Datasets

In order to conduct our empirical investigation, we have chosen several datasets that are recognized as the gold standards for tasks relating to optical flow estimation. These include the Sintel, KITTY-15, FlyingChairs, FlyingThings, and HD1K datasets. Using these datasets lends credibility to our research and facilitates a more effortless comparison of our results with those obtained using prior methods. These include but are not limited to, LiteFlowNet [Hui, Tang, and Loy, 2018], LiteFlowNet2 [Hui, Tang, and Loy, 2021], PWC-Net [Sun et al., 2017], FlowNet2 [Ilg et al., 2017], RAFT [Teed and Deng, 2020], GMA [Jiang et al., 2021b], DEQFlow [Bai et al., 2022], and FlowFormer [Huang et al., 2022].

Nonetheless, it is essential to note that some of these datasets do not come with sequential annotation, which is crucial to our study. As a solution, we propose to create these annotations on DAVIS and YouTube-VOS datasets using pre-trained models such as FlowFormer, DEQFlow, RAFT, and GMA. By averaging the predicted optical flows from each of these models, we intend to generate a set of reliable ground truth optical flow images that will serve as the baseline for our study. This approach ensures the robustness of our experimental setup and helps maintain the consistency and reliability of our findings.

Given the complex nature of optical flow estimation, a considerable volume of datasets is employed to ensure comprehensive coverage of the problem scope. The dataset chosen can often influence the progression of optical flow training, which generally follows an interpolation from simple to complex configurations.

In the initial stages of training, we utilize the FlyingChairs dataset, which serves as the foundation of our model. This complete training phase is instrumental in establishing the next version of the model upon which subsequent layers of complexity are added.

Subsequently, using the checkpoint established with the FlyingChairs dataset as a reference point, we train the model on the FlyingThings3D dataset. This step enables us to introduce a more complex learning level, thereby enhancing our model sophistication.

The next step in our training procedure involves continued system training using a combination of four datasets: Sintel, Kitti, HD1K, and FlyingThings3D. Combining these datasets introduces a broader spectrum of scenarios and complexities, expanding the model's exposure and learning depth.

Lastly, we shift the focus towards training the hidden state refinement on sequential data. This final phase utilizes the Sintel, DAVIS, and YouTube-VOS datasets.

In this manner, we systematically graduate our model learning from relatively simple scenarios to more complex ones, ensuring its robustness and adaptability across various optical flow estimation tasks.

FIGURE 5.1: Two examples from the Flying Chairs dataset (the image from [Dosovitskiy et al., 2015]).

5.1 FlyingChairs

The FlyingChairs dataset (Fig. 5.1) represents a significant benchmark in optical flow estimation. Designed with a clear intention to capture and represent the intricacies of real-world motion, this dataset offers researchers a robust platform for evaluating and developing novel methodologies for optical flow estimation.

FlyingChairs is a synthetically generated dataset that is comprised of a multitude of image pairs, each accompanied by their respective ground truth optical flow maps. The dataset consists of randomly "flying" images of chairs against various background scenes. This approach to data generation ensures a broad coverage of motion patterns and image structures.

The strength of the FlyingChairs dataset lies in its ability to introduce a high degree of variability into the data. By encompassing a wide array of scenarios, including diverse object shapes, motion paths, and background scenes, it challenges estimators, demanding more than rote learning of typical patterns.

The dataset synthetic nature enables the exact computation of ground truth optical flows, providing a valuable reference for the evaluation of optical flow estimation methodologies. Providing a precise and comprehensive ground truth allows researchers to gain detailed insights into the performance and reliability of their models.

The FlyingChairs dataset consists of 22,872 image pairs. The authors split the FlyingChairs dataset into 22,232 training and 640 test samples to provide local training and validation.

5.2 FlyingThings3D

The FlyingThings3D dataset (Fig. 5.1) is an innovative resource within the field of optical flow estimation, extending beyond traditional dataset compositions to provide a comprehensive and challenging landscape for model development and evaluation.

As a synthetic dataset, FlyingThings3D consists of a broad collection of image pairs accompanied by their corresponding ground truth optical flow maps. The dataset distinguishes itself by creatively using randomly oriented 3D objects. This robust and varied representation of motion patterns provides a rich environment for optical flow estimators to learn from and adapt to.

A crucial feature of FlyingThings3D is its complexity, stemming from the diverse set of motion trajectories, object shapes, and background scenes it encompasses. The dataset is particularly renowned for its highly complex, non-rigid, and multi-object

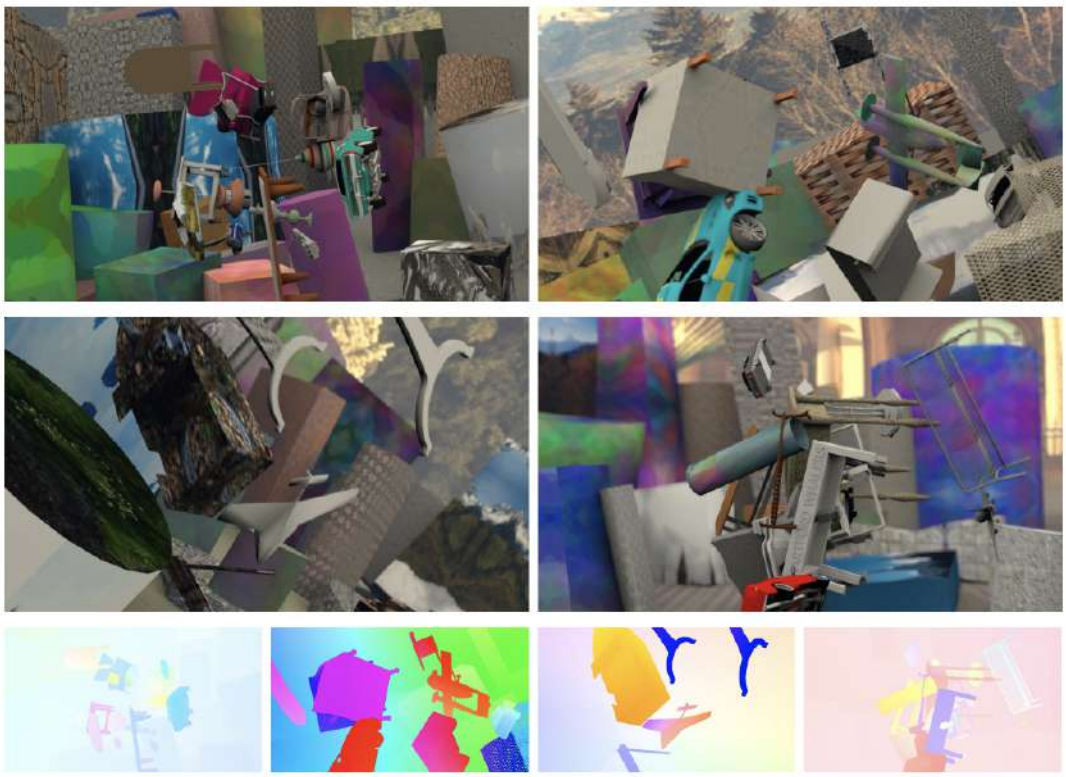

FIGURE 5.2: Examples from the FlyingThings3D dataset (the image from [Mayer et al., 2016]).

scenarios, which push the boundaries of what optical flow estimation models can handle.

Despite its synthetic nature, FlyingThings3D mimics real-world challenges due to its design principles. The ground truth optical flows are accurately calculated, providing a reliable benchmark for gauging the performance and accuracy of optical flow estimation methods.

The FlyingThings3D dataset has more than 39,000 stereo frames in 960×540 pixel resolution, which covers a vast environment with significant randomness. The authors aimed to create the dataset without concentrating on a particular task or enforcing strong naturalism.

5.3 Sintel

The Sintel dataset (Fig. 5.3) is a widely recognized benchmark in optical flow estimation. Derived from the open-source animated short film "Sintel" by the Blender Institute, this dataset provides a versatile and challenging array of real-world optical flow scenarios.

Unlike synthetic datasets, the Sintel dataset captures the complexities and nuances of real-world motion. It encompasses various conditions, including varying lighting, diverse textures, and intricate motion patterns, providing a comprehensive and rigorous test for optical flow estimation algorithms.

The Sintel dataset comprises various image sequences with corresponding ground truth optical flow maps. Naturalistic factors such as changes in illumination, atmospheric effects, and detailed textural features exemplify its complexity. The variety

FIGURE 5.3: By row from the top: Albedo Pass (flat, unshaded), Clean Pass (smooth shading, reflections), Final Pass (rendered with large number of effects) (the image from [Butler et al., 2012]).

in the types of motion, from slow and smooth to fast and abrupt, adds another level of challenge, making it an ideal dataset for pushing the boundaries of optical flow estimation techniques.

Moreover, the Sintel dataset offers two distinct versions for each sequence: a "clean" version and a "final" version. The clean version lacks atmospheric effects and includes basic shading and motion blur. In contrast, the final version incorporates additional realistic elements such as defocus blur, complex illumination effects, and intricate textures.

The dataset under consideration comprises 1,064 synthetic stereo images, each of which is accompanied by its corresponding ground truth disparity data. Encompassing 23 unique sequential scenes, the dataset offers a varied landscape for the execution of computer vision tasks.

Each image within the dataset, both RGB and disparity, boasts a resolution of 1024×436 pixels. This degree of detail challenges optical flow estimation models, pushing their capacity to discern and precisely predict fine-grained motion patterns. Thus, the dataset provides a substantial and challenging resource for developing and evaluating sophisticated optical flow estimation methodologies.

5.4 KITTI

The KITTI dataset, named after the Karlsruhe Institute of Technology and Toyota Technological Institute, where it was developed, stands as an integral benchmark in optical flow estimation.

Uniquely tailored to vehicular scenarios, the KITTI dataset captures the complexities of autonomous driving. It comprises a diverse collection of images captured from a moving vehicle, thereby introducing an assortment of real-world scenarios and providing a rigorous environment for developing and evaluating optical flow estimation methodologies.

The KITTI dataset is distinguished by its high-resolution stereo images, which are accompanied by accurate and meticulously annotated ground truth data. These include labels for various scenarios, including road, city, residential, and campus environments, providing various real-world scenes. The high resolution of the images in the KITTI dataset, combined with their capture from a moving vehicle, presents

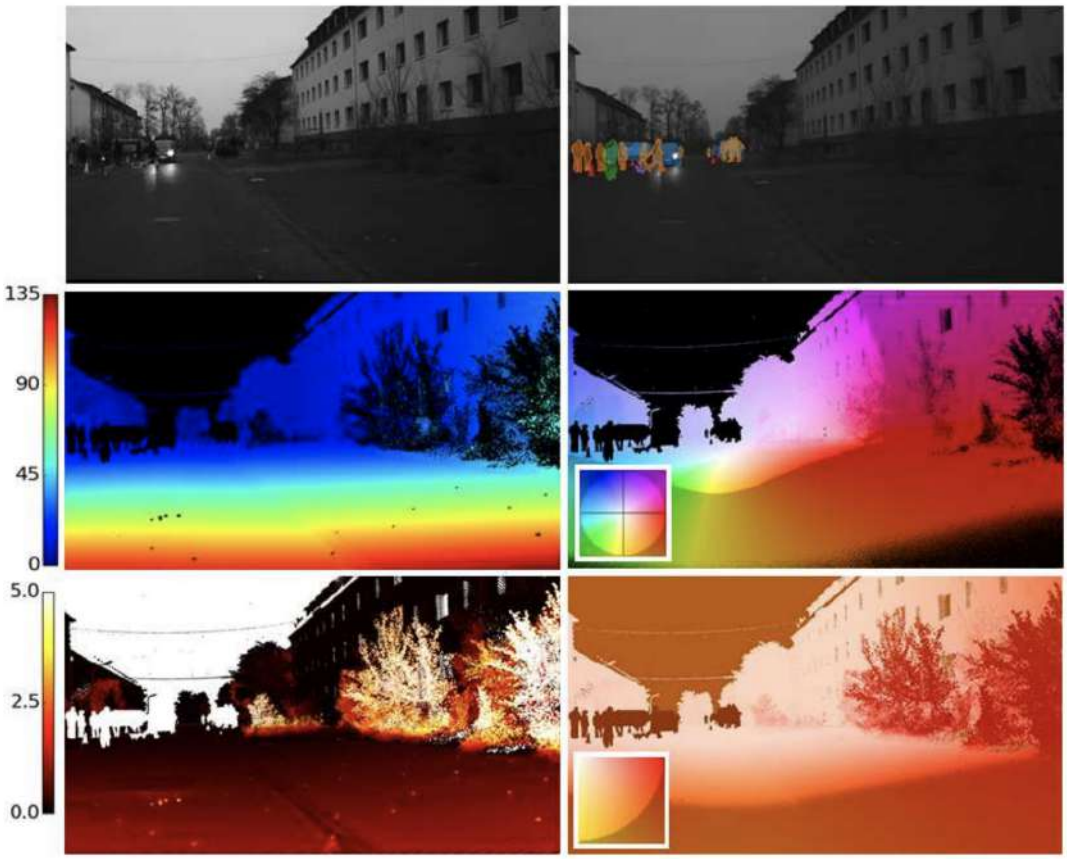

FIGURE 5.4: By row from the top: sample image (left) and labeled masks of dynamic regions (right), stereo image (left) and flow ground truth (right), uncertainties for stereo image (left) and flow (right) (the image from [Kondermann et al., 2016]).

a challenging environment for optical flow estimators. The need to precisely estimate the motion of other vehicles, pedestrians, and static objects within these complex urban environments pushes the boundaries of what optical flow algorithms can achieve.

The dataset has 12,919 images and ground truth data for the optical flow estimation. Both image pairs and optical flow ground truth have a 1392×512 pixels resolution.

5.5 HD1K

The HD1K dataset (Fig. 5.4) is a benchmarking resource within the domain of optical flow estimation that is noted for its high-definition content. This dataset contains images of a much higher resolution than those typically found in other optical flow datasets. The resolution of image pairs and optical flow samples is 2560×1080 pixels. This feature elevates the complexity and sophistication of the dataset, pushing the boundaries of optical flow estimation research.

HD1K comprises many image pairs (1000 samples), each provided alongside its corresponding ground truth optical flow map. The high resolution of these images enables the dataset to capture finer details and more intricate motion patterns, thereby challenging the capabilities of optical flow estimators. The images in the

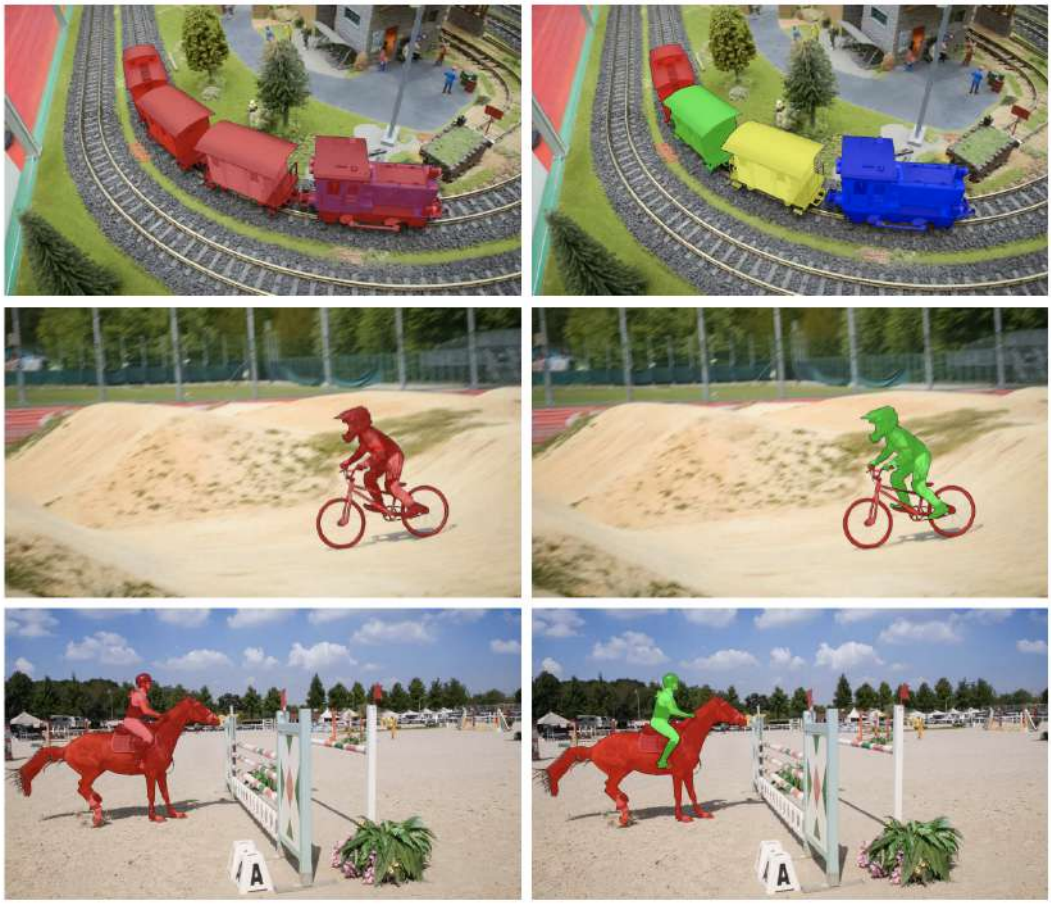

FIGURE 5.5: Example of the samples from the DAVIS dataset (the image from [Pont-Tuset et al., 2017]).

dataset depict a diverse range of scenarios, offering a robust environment for algorithm training and evaluation.

5.6 DAVIS

The DAVIS (Densely Annotated Video Segmentation) dataset has been a cornerstone in video segmentation, acting as a platform to evaluate algorithms related to this area. The DAVIS dataset offers high-quality, annotated video sequences to advance the development and assessment of video object segmentation techniques.

The DAVIS dataset (Fig. 5.5) includes various scenes, covering different object categories and situations and featuring numerous challenges, such as occlusions, motion blur, and changes in scale, viewpoint, or appearance. The original DAVIS dataset (DAVIS16) consists of 50 high-definition video sequences, each featuring a single annotated object. In contrast, the DAVIS17 dataset, an extension of the original, comprises 150 video sequences, each featuring multiple annotated objects.

These videos were carefully selected to represent different aspects of the video object segmentation problem, such as the need to track objects as they move through space or accurately separate objects from their backgrounds when these objects have complex shapes or are partially occluded.

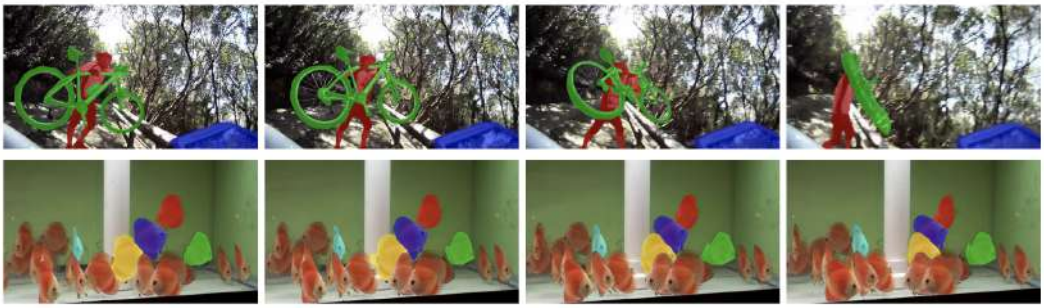

FIGURE 5.6: Example of the samples from the YouTube-VOS dataset (the image from [Xu et al., 2018]).

The DAVIS dataset, with its dense annotations and diverse video content, offers a robust and challenging benchmark for evaluating and improving video object segmentation methods. The dataset's wide use in the field signifies its importance in advancing state-of-the-art techniques.

In the case of optical flow and hidden state refinement tasks, the DAVIS dataset can help to improve the final metrics because it contains a large variety of sequences for the training stage, occlusions, and complex scenes with multiple objects, and the segmentation problem is related to the optical flow estimation one. We propose to use this dataset and annotate it with the FlowFormer, GMA, and DEQFlow methods.

5.7 YouTube-VOS

The YouTube-VOS (YouTube Video Object Segmentation) dataset (Fig. 5.6) is a crucial resource for developing and assessing video object segmentation approaches. This dataset provides a significant augmentation in size and diversity compared to its predecessors and has been instrumental in the evolution of video object segmentation.

The YouTube-VOS dataset consists of 4519 video sequences. The dataset is compiled from YouTube videos and comprises a broad range of scenarios, making it more representative of real-world situations.

What sets the YouTube-VOS dataset apart is its scale and diversity. The dataset is significantly larger than previous datasets, offering more than 90 object categories, and covers various scenarios and situations. The dataset also introduces new challenges for the segmentation task, such as the presence of multiple objects of the same class in a video and object appearance changes due to viewpoint changes, deformation, and occlusions. The YouTube-VOS dataset is divided into training and validation sets, enabling the development and validation of algorithms in a standardized manner. The training set contains 3,471 video sequences of 65 categories, while the validation set comprises 507 video sequences of 65 categories.

This dataset is also suitable for the optical flow estimation task due to its complexity, large number of sequences, support of occlusions, and different viewpoints. As for the DAVIS dataset, we annotate the YouTube-VOS dataset with the latest SOTA methods: FlowFormer, GMA, and DEQFlow methods

5.8 Summary

This chapter presents a comprehensive overview of the datasets used in our empirical study of optical flow estimation. We began by discussing the commonly used benchmark datasets: Sintel, KITTY-15, FlyingChairs, FlyingThings, and HD1K, elucidating their unique characteristics and their role in model training. The unavailability of sequential annotations in some of these datasets led us to propose generating annotations on DAVIS and YouTube-VOS datasets using pre-trained models like FlowFormer, DEQFlow, RAFT, and GMA. The chapter further detailed the systematic graduation of model training from simple to complex scenarios, ensuring the model robustness and adaptability. In-depth descriptions of individual datasets - FlyingChairs, FlyingThings3D, Sintel, KITTI, HD1K, DAVIS, and YouTube-VOS - were also provided, detailing their unique attributes and relevance to optical flow estimation tasks.

Chapter 6

Experiments

This chapter contains an experimental part to verify the hypothesis, evaluate them and compare it with the state-of-the-art results. The main focus is to check the combination of the DEQFlow and FlowFormer and hidden state refinement to produce an optical flow estimation speed up and model convergence based on sequential data and additional information from the previous pair of frames.

6.1 Preparation

As an initial point of our research, we considered state-of-the-art methods, especially DEQFlow and FlowFormer. This choice is motivated by three primary objectives:

1. They are achieving one of the best results in the Sintel and KITTI datasets;
2. We intend to explore the potential of combining the unique attributes of DEQFlow and FlowFormer, hypothesizing that the final combination may yield enhanced precision and novel insights into optical flow estimation;
3. We aspire to employ hidden state forecasting techniques based on the architectural design of FlowFormer.

A significant challenge in this research is the support of sequential data for the training and evaluation stages. This lack has resulted in many established pipelines, such as RAFT, FlowFormer, and GMA, being unequipped to use sequential data. To address this, we propose modifications to these pipelines, enabling them to process frame pairs sequentially and share relevant information to subsequent frame computations, thereby enhancing model performance and precision.

Recognizing the need for more sequential data, we have turned to video segmentation datasets, specifically DAVIS and YouTube-VOS. By applying the FlowFormer approach to these datasets, we have generated a vast supply of sequential data, which we then utilize in our hidden state forecasting approach. This strategy broadens our understanding of optical flow estimation and enables the development of more effective and efficient optical flow estimation techniques.

6.2 Hypotheses check

This section is aimed at testing the hypotheses that were derived earlier. We have three central hypotheses.

6.2.1 Combination of DEQFlow and FlowFormer approaches

Experiment 1. Combine the DEQFLow and FlowFormer approaches and train them as the standard optical flow method.

This experiment combines the FlyingChairs, FlyingThings3D, Sintel, KITTI, and HD1K datasets. We train the pipeline with the standard logic for optical flow estimation to converge systems better: from simple datasets to more complex ones.

Hardware. 4× NVIDIA RTX 3090 24Gb, AMD EPYC 7402 24-Core Processor, 128 GB RAM, 1T storage

Configuration. Batch size: 16, iterations (number of batches): 360,000, train time: 72 hours, feature and context encoders: Twins [Chu et al., 2021], GMA [Jiang et al., 2021b]: disabled, pretrain: from scratch

To integrate the abilities of DEQFlow and FlowFormer, we embarked on a multi-step training regime, each designed to progressively familiarize the model with the details of optical flow estimation.

1. The first phase involved training the model over 120,000 iterations on the FlyingChairs dataset. This relatively simple dataset served as a launchpad, introducing the model to the fundamental principles of optical flow estimation, thus establishing the rudimentary understanding needed to tackle more complex scenarios.

2. Building on the foundation established in the previous step, we subjected the model to a further 120,000 iterations using the FlyingThings3D dataset. This dataset, known for its intricate scenarios and instances of occlusions, allowed the model to learn the details of complex pixel movement and build robust strategies to handle occlusions effectively.

3. Finally, we trained the model to a further 120,000 iterations using the FlyingThings3D, Sintel, KITTI, and HD1K datasets. This diverse mixture of datasets, each reflecting a different aspect of the real-world environment, was instrumental in sharpening the model's ability to generalize across varied settings.

The decision to disable the GMA module was particularly significant during our training stage. While known for leveraging global and hidden object movement to evade occlusions, this module was potentially intrusive to our experimental objectives. By keeping it inactive, we ensured that our pipeline performance could be more precisely assessed, thus facilitating more explicit comparisons with other experimental results.

Results. We evaluated our results on the Sintel dataset with different sequence lengths (two and three sequential frames), previous hidden states and results share, and a warm start.

Method	Clean				Final			
	EPE	1px	3px	5px	EPE	1px	3px	5px
Seq=2	1.480	92.83	96.80	97.74	1.202	89.34	95.01	96.62
Seq=3, state share	1.618	92.44	96.48	97.45	1.393	89.01	94.59	96.22
Seq=3, warm start	1.558	92.56	96.60	97.56	1.247	89.28	94.87	96.48

TABLE 6.1: DEQFlow and FlowFormer combination metric results.

From Table 6.1, we can see that the combination of the FlowFormer and DEQFlow gives us worse results than the original FlowFormer paper (EPE: 0.48). We

Method	Clean			Final		
	Iterations	Seconds	EPE	Iterations	Seconds	EPE
Seq=2	53.3837	**0.4119**	**1.480**	53.6615	**0.4117**	**1.202**
Seq=3, state share	**53.1969**	0.4143	1.618	**53.2185**	0.4144	1.393
Seq=3, warm start	53.2746	0.4138	1.558	53.5059	0.4125	1.247

TABLE 6.2: DEQFlow and FlowFormer combination time usage.

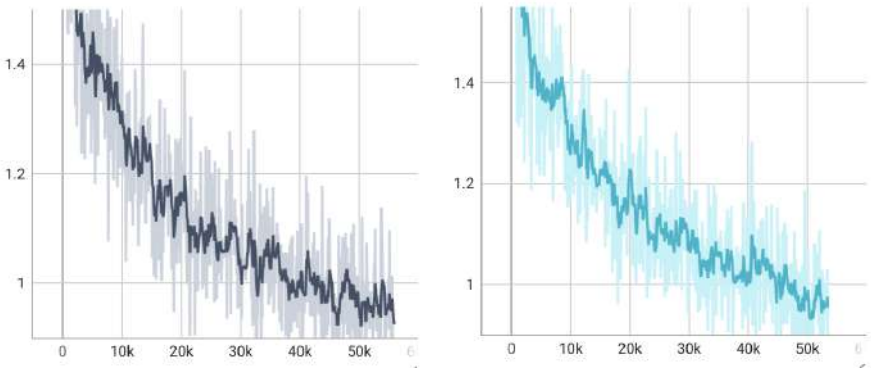

FIGURE 6.1: RefinerV1 and RefinerV2 EPE results during training.

also can see from Table 6.2 that state sharing gives us fewer iterations than other our experiment methods.

As a result, we can partially prove hypothesis 1 due to the reduction of the iterations, but the final EPE metrics are worse than the FlowFormer has.

6.2.2 Hidden state refinement for the following optical flow estimation

Experiment 2. Use hidden state refinement methods for the optical flow estimators . This experiment shows that hidden state refinement can speed up the optical flow system and reduce the number of update block execution, which is the bottleneck of the latest state-of-the-art systems (FlowFormer, GMA, DEQFlow).

Configuration. Batch size: 16, iterations (number of batches): 54,000, train time: 48 hours, feature and context encoders: Twins [Chu et al., 2021], GMA [Jiang et al., 2021b]: disabled, pretrain: from the FlowFormer Sintel model.

In this section, we propose two versions of the hidden state refinement models, which predict the next optical flow before update block execution, and reduce the number of the update block steps.

The first version (RefinerV1) of the hidden state forecaster contains two main blocks: StateRefiner and StateMixer.

The StateRefiner uses an attention mechanism to predict the pixel displacement from the first output of the optical flow to the following one. The keys of the attention approach are the source frame (n-1 position in the sequence), the queries are the target frame (n position in the sequence), and the values are the flow and hidden state of the current frame pair. As an output, we get a new optical flow image and an updated hidden state for the update block usage.

$$Attention(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d}})V \tag{6.1}$$

After it, we get information from the context network based on the frame (n position in the sequence) and send this data with the output of StateRefiner into the StateMixer. This block uses multiple convolutions and ResNet blocks to achieve proper data mixing. The output of the mixer is the combination of the refined optical flow and hidden state which the update block process can decode.

The second version (RefinerV2) of the hidden state forecaster uses the same StateRefiner as the first architecture contains, but we use RecurrentStateMixer.

RecurrentStateMixer is built with a combination of convolution and recurrent neural networks. The convolutions work as feature extractors, and recurrent networks help to converge the system.

During our experiment, we added the stop criteria to the update block, which helped to reduce the number of update block iterations. The logic of the stop criteria is: we stop iterations if the relative optical flow change is less than constant. In our case, we have used 0.001.

	Clean			Final		
Method	Iterations	Seconds	EPE	Iterations	Seconds	EPE
1	12	0.2143	**0.4339**	12	0.2134	**0.6309**
2	3.8904	**0.1734**	0.4605	4.0135	**0.173**	0.659
3	**3.8022**	0.1949	0.4587	**3.9173**	0.1946	0.6453
4	3.9577	0.1798	0.4775	4.0876	0.1794	0.6814
5	3.8307	0.1959	0.4545	3.9222	0.1953	0.6431
6	3.9577	0.1804	0.4775	4.0876	0.1794	0.6814

TABLE 6.3: Time spent based on refinement methods. From top to bottom: baseline with sequence 2; baseline with sequence 2 and stop criteria; RefinerV1, sequence 3, state share, stop criteria; RefinerV1, sequence 3, stop criteria, warm start; RefinerV2, sequence 3, state share, stop criteria; RefinerV2, sequence 3, stop criteria, warm start.

Results. From Table 6.3 we define that stop criteria effectively reduce the number of iterations without strong code changes. Both versions of the refiner slightly reduced the iterations, but the RefinerV1 gave better results. This means that sequence refinement methods can help reduce the update block iterations without wasting time on additional optical flow computation. Hence, hypothesis 2 is proven.

	Clean				Final			
Method	EPE	1px	3px	5px	EPE	1px	3px	5px
1	0.434	94.367	97.89	98.681	0.631	91.504	96.531	97.846
2	0.461	94.06	97.815	98.647	0.659	91.065	96.464	97.811
3	0.459	93.986	97.783	98.643	0.645	91.01	96.415	97.802
4	0.478	93.959	97.772	98.61	0.681	90.929	96.348	97.724
5	0.455	93.998	97.79	98.641	0.643	91.034	96.419	97.799
6	0.478	93.959	97.772	98.61	0.681	90.929	96.348	97.724

TABLE 6.4: Refiner metric results. From top to bottom: baseline with sequence 2; baseline with sequence 2 and stop criteria; RefinerV1, sequence 3, state share, stop criteria; RefinerV1, sequence 3, stop criteria, warm start; RefinerV2, sequence 3, state share, stop criteria; RefinerV2, sequence 3, stop criteria, warm start.

FIGURE 6.2: The example of optical flow forecasting. Left image: output from the RerinerV1, center and right images: last iteration of the update block.

6.2.3 Quality of hidden state refinement

Experiment 3. Use hidden state refinement methods for the optical flow estimators to prove that the output of the hidden state forecaster can contain most of the optical flow result . We assume that on short time spans movement is close to linear. Hence we can predict the following image pair optical flow result or most of it with good quality.

Configuration. Batch size: 16, iterations (number of batches): 54,000, train time: 48 hours, feature and context encoders: Twins [Chu et al., 2021], GMA [Jiang et al., 2021b]: disabled, pretrain: from the FlowFormer Sintel model.

In this section, we delve into the foundational building blocks of our empirical investigation, providing an exhaustive analysis of the datasets harnessed in our study of optical flow estimation. Sintel, KITTY-15, FlyingChairs, FlyingThings, and HD1K are used to train the primary FlowFormer approach. We have used the Sintel, DAVIS, and YouTube-VOS datasets to train the refinement models because they support image pair sequences. The sequential dataset ideology helps to detect pixel displacement to the nearest future with good results.

Method	Clean				Final			
	EPE	1px	3px	5px	EPE	1px	3px	5px
1	**5.415**	66.761	**77.818**	**82.062**	**5.148**	**66.86**	**77.679**	**82.03**
2	5.959	64.858	76.85	81.527	5.714	64.798	76.723	81.317
3	5.441	**66.849**	77.678	81.841	5.151	66.832	77.516	81.813
4	5.959	64.858	76.85	81.527	5.714	66.832	76.723	81.317

TABLE 6.5: Refiner output metric results. From top to bottom: RefinerV1, sequence 3, state share; RefinerV1, sequence 3, warm start; RefinerV2, sequence 3, state share; RefinerV2, sequence 3, warm start.

Results. From Table 6.5 we can conclude that with state sharing and warm start, we achieve decent optical flow results, which can follow the directed linear movement and forecast the new result event without seeing the next frame (n+1 in a sequence). RefinerV1 gives better results even than the light warm start approach. Hence hypothesis 3 is proven.

6.3 Models

This section is geared towards an in-depth dissection and exploration of refinement models, critically examining their structures, functionalities, strengths, and limitations in diverse scenarios. The proposed solution contains three models: DEQFlow with FlowFormer (deq-flow-former), RefinerV1, and RefinerV2.

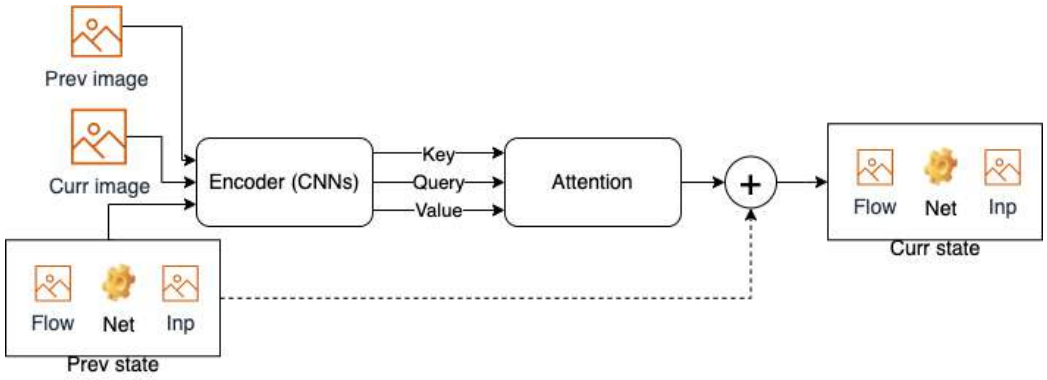

FIGURE 6.3: Refiner module architecture.

6.3.1 DEQFlow with FlowFormer

The proposed model in this study amalgamates two state-of-the-art methods: DEQFlow and FlowFormer. Several reasons underscore the rationale behind this amalgamation:

Firstly, both methods have demonstrated superior performance in achieving top results based on the Sintel and Kitti datasets. DEQFlow, as a framework, can be aptly fitted to most optical flow models. This compatibility underlines its value and applicability across various use cases.

Secondly, the foundational model (RAFT) upon which DEQFlow is based needs to be updated in light of the state-of-the-art results. This outmoded nature necessitates a refresh to maintain relevance and achieve improved performance.

Thirdly, the DEQFlow model inherently supports sequential data, which is advantageous when dealing with real-world dynamic environments. The ability to process sequential data makes the model more robust and accurate.

Lastly, the DEQFlow model offers significant benefits by accelerating model optimization and execution. Speed is an essential aspect of model performance, as real-time or near-real-time processing is often required in many applications of optical flow models.

Hence, we have elected to incorporate the FlowFormer as the new foundational model for the DEQFlow framework. We have designated a FlowFormer update iteration model segment as the primary target for DEQFlow optimization. The components of this iterative part encompass the following steps:

Firstly, the process involves encoding flow tokens. These flow tokens are needed for further convenient computation for the attention mechanism.

Secondly, the model decodes the cost volume into cost memory. The cost volume represents the dissimilarity between two images for all possible displacements, while the cost memory distillates them and stores these computed dissimilarities.

Lastly, the model employs a Gated Recurrent Unit (GRU) in its architecture. GRUs are equipped to handle iterative improvement of optical flow.

6.3.2 Hidden state refiners

The main goal for the hidden state refiners is to support the model convergence and its update block iteration reduction. This is facilitated by using sequential data and utilizing the outcomes of prior optical flow analyses to refine the following results based on pixel inertia. The two constituent elements of the hidden state refiners are the Refiner and the StateMixer.

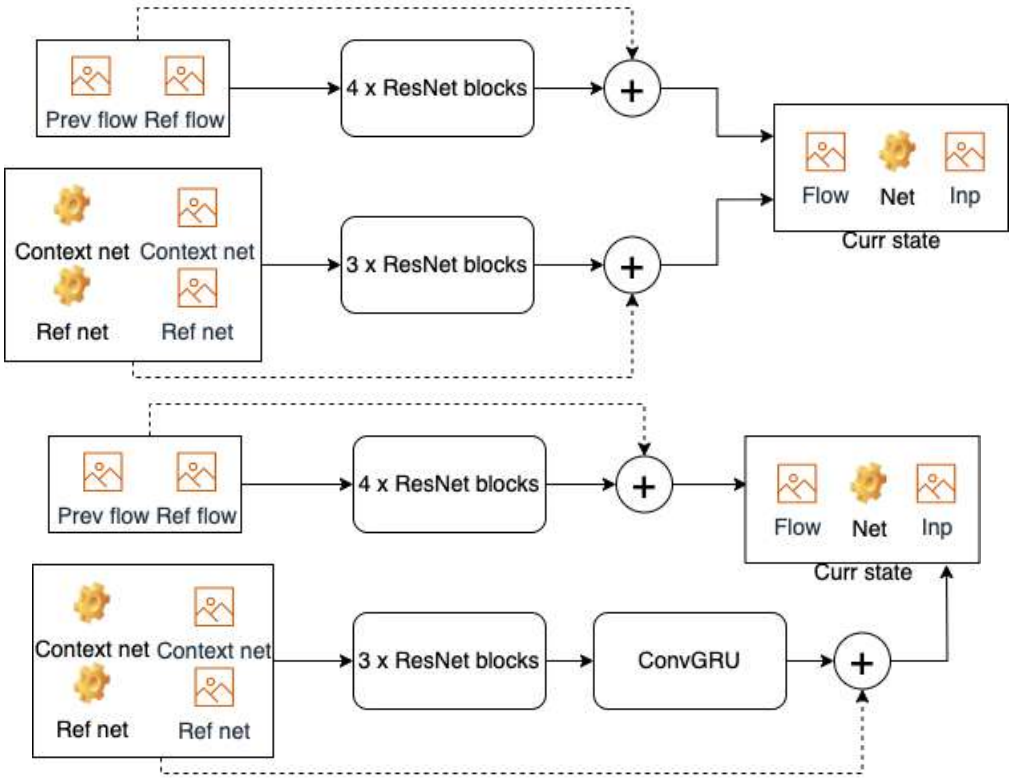

FIGURE 6.4: StateMixer module architecture from RefinerV1 (top) and RefinerV2 (bottom).

With its incorporated attention mechanism, the Refiner (Fig.6.3) aims to allocate the displacement of pixels from a prior image pair to the following one. By exploiting the inertia of motion, it contemporizes the optical flow movement and updates the hidden state of the block. In order to achieve this, the Refiner accesses images from the current pair to establish a correlation with optical flow and hidden state results, thereby forecasting the upcoming one. This model comprises an attention block and Convolutional Neural Networks (CNNs), which serve to project and encode the data.

After the application of the Refiner, the StateMixer (Fig.6.4) is executed. Its designated role is to optimally amalgamate features derived from the FlowFormer context network and the refined data. We offer two variants of the StateMixer; one devoid of a recurrent part and another inclusive of one. Consequently, we have two possible combinations of the Refiner and StateMixer: RefinerV1 (without a recurrent part) and RefinerV2 (with a recurrent part).

Both iterations of the StateMixer employ Residual Neural Network (ResNet) blocks to counteract the fading gradient, thereby facilitating network convergence. We have utilized four ResNet blocks to incorporate optical flow results (previous optical flow and refined one) and three additional blocks to predict the succeeding hidden state for the update block. This prediction is based on the refined output and one result from the context network.

In RefinerV2, we have enhanced the StateMixer by integrating a Gated Recurrent Unit (GRU) to the output of the results to forecast the hidden state. The rationale behind this modification was to mimic the update block logic and create a more significant constraint for the changes in the hidden state. This sophisticated methodological approach empowers the model to process temporal information more efficiently

while minimizing computational complexity and reducing training time.

6.4 Results

Method	Clean EPE	Final EPE
RAFT	0.76	1.22
DEQFlow	0.73	1.02
DEQFlow-FlowFormer	1.618	1.393
GMA	0.62	1.06
FlowFormer	0.434	0.631
1	0.461	0.659
2	0.459	0.645
3	0.478	0.681
4	0.455	0.643
5	0.478	0.681

TABLE 6.6: Refiner metric results. (1): baseline with sequence two and stop criteria; (2): RefinerV1, sequence three, state share, stop criteria; (3): RefinerV1, sequence three, stop criteria, warm start; (4): RefinerV2, sequence three, state share, stop criteria; (5) RefinerV2, sequence three, stop criteria, warm start.

We have reached specific practical observations in light of our empirical investigations and analysis. A key finding from our experimentation is the suboptimal performance of the combination of DEQFlow and FlowFormer for optical flow estimation. Despite this outcome, we have discerned that this combination still has utility in expediting the training process for optical flow estimators like FlowFormer, GMA, and RAFT, thus facilitating a more efficient computational model.

Our study further revealed the commendable performance of hidden state forecasters. These forecasters leverage the information from the preceding image pair to update it and provide the initialization for the update block to RAFT-like architectures.

Fig. 6.5 shows the reduction of the End-Point Error (EPE) as a function of iterative steps, compared against the ground truth data without applying any stopping criteria. The visual data underscores the superior efficacy of RefinerV2 when compared with the current state-of-the-art (SOTA) method, FlowFormer.

The enhanced performance of RefinerV2 can be largely attributed to the use of sequential data and the hidden state refinement methods, which use Convolutional Neural Networks (CNNs), Attention, and Recurrent Neural Networks (RNNs).

The use of sequential data is critical to RefinerV2 improved results. The refined hidden states, enhanced by the capabilities of CNNs, Attention, and RNNs, facilitate transmitting valuable information from previous frames to the current one, thereby minimizing the error rate. Thus, through the judicious employment of sequential data and the refinement of hidden states via convolutional, attention, and recurrent neural networks, RefinerV2 has demonstrated notable improvements over the state-of-the-art method, FlowFormer. These findings underscore the significance of these advanced methodologies in enhancing performance in optical flow prediction, paving the way for future research advancements in this domain.

As we turn to Table 6.6, encapsulates a comparative analysis of our methodologies with other well-established models. Even with the effort put into our methods,

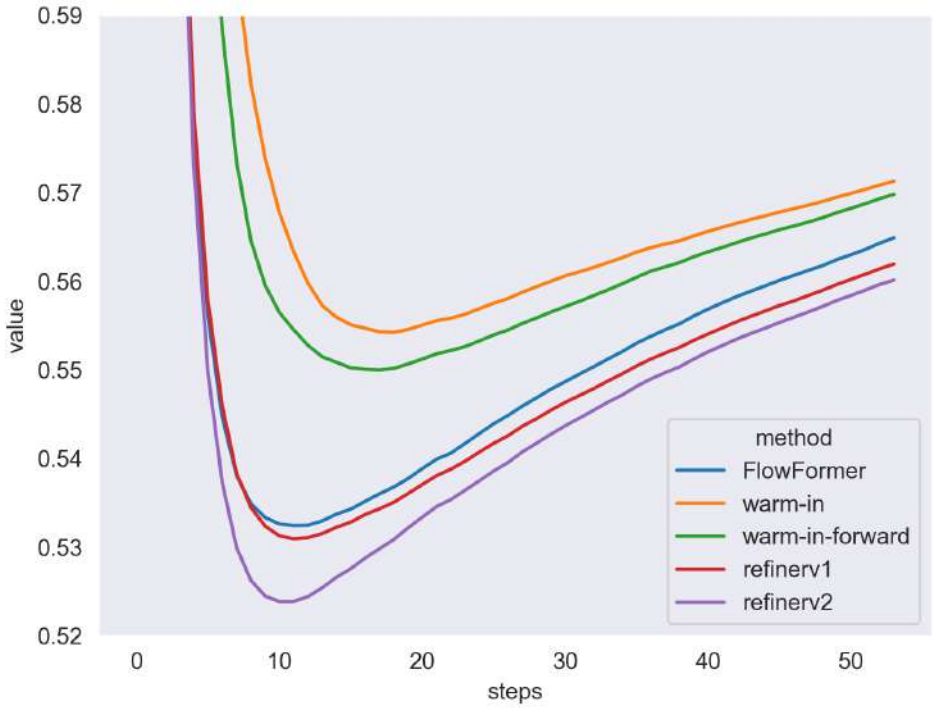

FIGURE 6.5: EPE convergence of the methods during iterations.

it becomes clear that they do not surpass the state-of-the-art FlowFormer method in precision. However, such comparisons ground our research within the broader field context, providing valuable insights for future work.

During our research, we gleaned several insights into the practical implications of our methods. Notably, they demonstrate the potential to decrease the number of update block executions, a critical factor when considering model speed. This capability effectively addresses the bottleneck issue in model speed up, thereby enhancing the efficiency of the entire optical flow estimation process.

Chapter 7

Conclusions

The thesis studies optical flow forecasting based on hidden state refinement. This research prime objective is to enhance the speed of RAFT-like architectures and optimize the convergence of training data by leveraging the information from the preceding image pair.

Throughout this thesis, we have explicated the state-of-the-art methods in detail, shedding light on their functionalities and pointing out their inherent limitations. Further, we meticulously cataloged the datasets employed and delineated the procedural pipeline for our distinctive combination of DEQFlow and FlowFormer, alongside two hidden state refinement methods variants. This comprehensive approach enabled us to rigorously test three hypotheses and establish comparable results with state-of-the-art methods.

As an academic endeavor, the implications of this thesis transcend mere empirical findings. Indeed, it enriches the scholarly understanding of the domain of hidden state refinement for optical flow forecasting. This study, in its entirety, contributes to the evolution of the discourse on optical flow forecasting by providing fresh perspectives on methodological approaches and their practical ramifications.

Moreover, our exploration into the domain of hidden state refinement and its applications to optical flow forecasting underscores the potential of these strategies in addressing contemporary challenges in the field.

Appendix A

Code

A.1 Refinement code

The full code you can find in the GitHub repository of the hidden-state-refinement-for-optical-flow-forecasting[1] and deq-flow-former[2].

1

```
class StateRefiner(nn.Module):
    def __init__(self, cfg, dim, heads=4, dim_head=128):
        ...

    def forward(self, prev_frame, curr_frame, prev_flow, prev_net, prev_inp):
        prev_frame, curr_frame = self.prepare_image_feats(
            prev_frame, curr_frame)
        prev_flow_features = self.increase_flow_dims(prev_flow)
        heads, _, _, h, w = self.heads, *prev_frame.shape
        q = self.to_q(curr_frame)
        k = self.to_k(prev_frame)
        q, k = map(
            lambda t: rearrange(t, 'b (h d) x y -> b h x y d', h=heads), (q, k)
        )
        q = self.scale * q
        sim = einsum('b h x y d, b h u v d -> b h x y u v', q, k)
        sim = rearrange(sim, 'b h x y u v -> b h (x y) (u v)')
        attn = sim.softmax(dim=-1)
        out_flow = self.precess_v(
            attn, prev_flow_features, heads, h, w, self.to_v_flow,
            self.project_flow
        )
        out_flow = self.reduce_flow_dims(out_flow)
        out_net = self.precess_v(
            attn, prev_net, heads, h, w, self.to_v_net,
            self.project_net
        )
        out_inp = self.precess_v(
            attn, prev_inp, heads, h, w, self.to_v_inp,
            self.project_inp
```

[1] https://github.com/unexpectedjourney/hidden-state-refinement-for-optical-flow-forecasting

[2] https://github.com/unexpectedjourney/deq-flow-former

```
        )
        out_flow = prev_flow + self.gamma_flow * out_flow
        out_net = prev_net + self.gamma_net * out_net
        out_inp = prev_inp + self.gamma_inp * out_inp
        return out_flow, out_net, out_inp

class StateMixer(nn.Module):
    def __init__(self):
        super(StateMixer, self).__init__()
        self.flow_out = nn.Sequential(
            nn.Conv2d(4, 128, 3, padding=1),
            nn.ReLU(),
            ResidualBlock(128, 128),
            ResidualBlock(128, 128),
            nn.Conv2d(128, 256, 3, padding=1),
            nn.ReLU(),
            ResidualBlock(256, 256),
            ResidualBlock(256, 256),
            nn.Conv2d(256, 2, 3, padding=1),
            nn.ReLU(),
        )
        self.net_inp_out = nn.Sequential(
            nn.Conv2d(512, 256, 7, padding=3),
            ResidualBlock(256, 256),
            ResidualBlock(256, 256),
            ResidualBlock(256, 256, activate=False),
        )
        self.gamma_flow = nn.Parameter(torch.tensor(0.5))
        self.gamma_net = nn.Parameter(torch.tensor(0.5))
        self.gamma_inp = nn.Parameter(torch.tensor(0.5))

    def forward(
            self,
            flow_init,
            net_init,
            inp_init,
            flow_ref,
            net_ref,
            inp_ref,
    ):
        flow = torch.cat([flow_init, flow_ref], dim=1)
        flow = self.flow_out(flow)
        net_inp = torch.cat([net_init, inp_init, net_ref, inp_ref], dim=1)
        net_inp = self.net_inp_out(net_inp)
        net, inp = torch.split(net_inp, [128, 128], dim=1)
        net = torch.tanh(net)
        inp = torch.relu(inp)
        flow = (1 - self.gamma_flow) * flow_init + self.gamma_flow * flow
        net = (1 - self.gamma_net) * net_init + self.gamma_net * net
        inp = (1 - self.gamma_inp) * inp_init + self.gamma_inp * inp
        return flow, net, inp
```

Bibliography

Anderson, Donald G. (1965). "Iterative Procedures for Nonlinear Integral Equations". In: *J. ACM* 12.4, 547–560. ISSN: 0004-5411. DOI: 10.1145/321296.321305. URL: https://doi.org/10.1145/321296.321305.

Bai, Shaojie et al. (2022). "Deep Equilibrium Optical Flow Estimation". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. IEEE, pp. 610–620. DOI: 10.1109/CVPR52688.2022.00070. URL: https://doi.org/10.1109/CVPR52688.2022.00070.

Broyden, Charles G (1965). "A class of methods for solving nonlinear simultaneous equations". In: *Mathematics of Computation* 19.92, pp. 577–593.

Butler, Daniel J. et al. (2012). "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*. Ed. by Andrew W. Fitzgibbon et al. Vol. 7577. Lecture Notes in Computer Science. Springer, pp. 611–625. DOI: 10.1007/978-3-642-33783-3_44. URL: https://doi.org/10.1007/978-3-642-33783-3_44.

Byrd, R. H. et al. (2016). "A Stochastic Quasi-Newton Method for Large-Scale Optimization". In: *SIAM Journal on Optimization* 26.2, pp. 1008–1031.

Chu, Xiangxiang et al. (2021). "Twins: Revisiting the Design of Spatial Attention in Vision Transformers". In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. Ed. by Marc'Aurelio Ranzato et al., pp. 9355–9366. URL: https://proceedings.neurips.cc/paper/2021/hash/4e0928de075538c593fbdabb0c5ef2c3-Abstract.html.

Dosovitskiy, Alexey et al. (2015). "FlowNet: Learning Optical Flow with Convolutional Networks". In: *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, pp. 2758–2766. DOI: 10.1109/ICCV.2015.316. URL: https://doi.org/10.1109/ICCV.2015.316.

Geiger, Andreas et al. (2013). "Vision meets robotics: The KITTI dataset". In: *Int. J. Robotics Res.* 32.11, pp. 1231–1237. DOI: 10.1177/0278364913491297. URL: https://doi.org/10.1177/0278364913491297.

Horn, B.K.P. and B.G. Schunck (1981). "Determining optical flow". In: *Artificial intelligence* 17.1-3, pp. 185–203.

Huang, Zhaoyang et al. (2022). "FlowFormer: A Transformer Architecture for Optical Flow". In: *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XVII*. Ed. by Shai Avidan et al. Vol. 13677. Lecture Notes in Computer Science. Springer, pp. 668–685. DOI: 10.1007/978-3-031-19790-1_40. URL: https://doi.org/10.1007/978-3-031-19790-1_40.

Hui, Tak-Wai, Xiaoou Tang, and Chen Change Loy (2018). "LiteFlowNet: A Lightweight Convolutional Neural Network for Optical Flow Estimation". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. Computer Vision Foundation / IEEE Computer Society, pp. 8981–8989. DOI: 10.1109/CVPR.2018.00936. URL: http://openaccess.thec

vf.com/content_cvpr_2018/html/Hui_LiteFlowNet_A_Lightweight_CVPR_2018_paper.html.

Hui, Tak-Wai, Xiaoou Tang, and Chen Change Loy (2021). "A Lightweight Optical Flow CNN - Revisiting Data Fidelity and Regularization". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 43.8, pp. 2555–2569. DOI: 10.1109/TPAMI.2020.2976928. URL: https://doi.org/10.1109/TPAMI.2020.2976928.

Ilg, Eddy et al. (2017). "FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, pp. 1647–1655. DOI: 10.1109/CVPR.2017.179. URL: https://doi.org/10.1109/CVPR.2017.179.

Jiang, Shihao et al. (June 2021a). "Learning Optical Flow from a Few Matches". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, USA: IEEE, pp. 16587–16595. ISBN: 9781665445092. DOI: 10.1109/CVPR46437.2021.01632. URL: https://ieeexplore.ieee.org/document/9577699/ (visited on 12/25/2022).

Jiang, Shihao et al. (2021b). "Learning to Estimate Hidden Motions with Global Motion Aggregation". In: *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, pp. 9752–9761. DOI: 10.1109/ICCV48922.2021.00963. URL: https://doi.org/10.1109/ICCV48922.2021.00963.

Kondermann, Daniel et al. (2016). "The HCI Benchmark Suite: Stereo and Flow Ground Truth with Uncertainties for Urban Autonomous Driving". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2016, Las Vegas, NV, USA, June 26 - July 1, 2016*. IEEE Computer Society, pp. 19–28. DOI: 10.1109/CVPRW.2016.10. URL: https://doi.org/10.1109/CVPRW.2016.10.

Lucas, Bruce D, Takeo Kanade, et al. (1981). "An iterative image registration technique with an application to stereo vision." In: *IJCAI*. Vol. 81, pp. 674–679.

Mayer, Nikolaus et al. (2016). "A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, pp. 4040–4048. DOI: 10.1109/CVPR.2016.438. URL: https://doi.org/10.1109/CVPR.2016.438.

Pont-Tuset, Jordi et al. (2017). "The 2017 DAVIS Challenge on Video Object Segmentation". In: *arXiv:1704.00675*.

Sun, Deqing et al. (2017). "PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume". In: *CoRR* abs/1709.02371. arXiv: 1709.02371. URL: http://arxiv.org/abs/1709.02371.

Teed, Zachary and Jia Deng (2020). "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*. Ed. by Andrea Vedaldi et al. Vol. 12347. Lecture Notes in Computer Science. Springer, pp. 402–419. DOI: 10.1007/978-3-030-58536-5_24. URL: https://doi.org/10.1007/978-3-030-58536-5_24.

Xu, Ning et al. (2018). "YouTube-VOS: Sequence-to-Sequence Video Object Segmentation". In: *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part V*. Ed. by Vittorio Ferrari et al. Vol. 11209. Lecture Notes in Computer Science. Springer, pp. 603–619. DOI: 10.1007/978-3-030-01228-1_36. URL: https://doi.org/10.1007/978-3-030-01228-1_36.

Zhang, Feihu et al. (2021). "Separable Flow: Learning Motion Cost Volumes for Optical Flow Estimation". In: *2021 IEEE/CVF International Conference on Computer*

Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021. IEEE, pp. 10787–10797. DOI: 10.1109/ICCV48922.2021.01063. URL: https://doi.org/10.1109/ICCV48922.2021.01063.

Zhao, Shiyu et al. (2022). "Global Matching with Overlapping Attention for Optical Flow Estimation". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. IEEE, pp. 17571–17580. DOI: 10.1109/CVPR52688.2022.01707. URL: https://doi.org/10.1109/CVPR52688.2022.01707.