Machine Perception Project 6 : Human optical flow

Ankush Panwar
apanwar@student.ethz.ch

Nivedita Nivedita
nnivedita@student.ethz.ch

Seyedmorteza Sadat
ssadat@student.ethz.ch

ABSTRACT

Human optical flow indicates the body movement, and hence is important for human behavior understanding. In this project, we learn the human optical flow from the MHOF dataset [7], where we have multiple synthetic humans moving in front of random backgrounds. We implement the RAFT [12] architecture together with data augmentation techniques to solve the problem at hand. We also generate additional data from [7] for training the network.

1 INTRODUCTION

Optical flow task is the estimation of apparent motion of pixels in two video frames as the camera and scene moves. There are many factors such as large motions, blur, featureless regions, occlusion etc which make optical flow prediction difficult. With advancement in deep learning approaches, many existing papers [11] [6] tries to solve this task with considerable accuracy and efficient inference and are gradually replacing the classical variational-based approaches [2] [10]. However, it is tough to collect the ground truth of dense optical flow in reality, which makes most supervised methods heavily dependent on the large-scale synthetic datasets.

In this task, we use one such generated synthetic dataset by Ranjan *et.al* [7] called Multi-Human Optical Flow (MHOF) dataset. They use a 3D model of the human body and motion to synthesize realistic flow fields in multi-person images. Video frames generated in MHOF contain multiple people with random backgrounds and consist of significant occlusion between persons.

SpyNet [6] and PWC-Net [11] are among few deep learning based approaches for solving the optical flow estimation problem. PWC-Net [11] uses well-established principles of pyramidal processing, warping, and the use of a cost volume together with CNN to predict the motion. It uses coarse-to-fine design where flow is first estimated at low resolution and then upsampled and refined at high resolution. However, due to this coarse-to-fine design, we face difficulty of recovering from errors at coarse resolutions and could potentially miss small fast-moving objects. Moreover, PWC-Net contains 8.8M parameters and could overfit when the training data is scarce.

Our key insights and techniques used to solve the optical flow task at hand are given below:

1. **RAFT Architecture**: Our implemented **RAFT** [12] (Recurrent All-Pairs Field Transforms for Optical Flow) network solves the coarse-to-fine error problem by having only a single fixed flow field at high resolution. It contains less parameters (4.5M compared to 8.8M inPWC-Net) that makes it less prone to over-fitting. It also handles the occlusion cases better.
2. **Data generation**: While training a deep learning based network like RAFT or PWC-Net, large amount of data is required to optimize the parameters without overfitting the model. However, we were given only around 8k image pairs as our training data. So, we generated additional data using the Ranjan *et.al* [7] to increase the size of the training set.
3. **Data Augmentation**: We also utilized various data augmentation techniques such as scaling, color jitter noise, horizontal and vertical flip etc to further boost our training examples and prevent overfitting.

We achieve a score of **0.437 EPE** on the test set in our final submission.

2 RELATED WORK

Dosovitskiy et al. [3] constructed two CNN networks namely, FlowNetS and FlowNetC respectively, inspired by the recent success of CNN networks in computer vision tasks. The network uses U-Net denoising autoencoder. Despite being pretrained largely on synthetic chairs, it performed well on fast moving objects on Sintel dataset. This network was then used by Mayer et al. [4] for disparity and scene flow estimates.

This was further improved by incorporating classical approaches into the network architecture in SpyNet [6] and PWC-Net [11]. In PWC-Net, traditional image pyramids are replaced with learnable feature pyramids in order to circumvent the condition of images being variant to light and shadows in the network. Large motions are estimated using the traditional warping operation as a layer. The network contains a layer to construct the cost volume. It is then further processed by CNN layers to estimate the flow. The flow is then processed using a context network. In PWC-Net however, dilated convolutions are used to integrate contextual information for optical flow.

In contrast to PWC-Net which uses feature pyramids, RAFT maintains and updates a single fixed flow field at high resolution. This architecture overcomes several limitations of cascaded architectures like PWC-Net [11] and SpyNet [6] that compound the difficulty of recovering from errors at coarse resolutions, and the considerable number of training iterations typically required in these networks. RAFT architecture consists of 3 main components namely, feature encoder, to extract per-pixel features, the correlation layer, which computes visual similarity between pixels and the update operator, which mimics the steps of an iterative optimization algorithm.

3 METHOD

This section briefly describes the main components of our approach, namely data generation, data augmentation, network structure and training details.

3.1 Data Generation

We had access to limited amount of data, and so to circumvent the issue of overfitting, we generate a dataset of multi-human optical flow using [7]. We use a 3D model of the human body and motion capture data to synthesize realistic flow fields. We employ Blender2

and its Cycles rendering engine to generate realistic synthetic image frames and optical flow. It uses the SMPL+H [8] that models the body together with articulated fingers to have richer motion variation. We further obtain moving sequences from CMU [1] and HumanEva [9] MoCap datasets to augment motion variation. We generate the images and flo files containing 5-7 persons in each image which is in-lined with the given MHOF dataset images sequences. Optical flow networks are then trained to estimate human flow fields from pairs of images, the current image and the previous image.
Google drive Link to our generated data: Generated Data

3.2 Data Augmentation

We also use data augmentation to prevent overfitting and enhance training,. More specifically, we exploit the color augmentation technique, which changes the brightness, hue, and saturation of each image, as well as spatial augmentation such as rescaling, cropping, and random vertical and horizontal flips. Following data augmentations are performed to train the model:

(1) Random ColorWarp: It randomly introduces noise to the input image pixels from a uniform distribution of mean 10 and standard deviation of 0.1.
(2) Random Scale: Re-scales the inputs and target randomly between 0.8 to 1.5 scale.
(3) Random Translate: Translate the input and target randomly between -10 to +10 pixels.
(4) Random Rotate: Random rotation of the image from -10 to +10 degree.
(5) Random Crop: crops the image at a random location to have an output of size (320,448).
(6) Random VerticalFlip and HorizontalFlip: flip the image randomly in horizontal and vertical directions.

3.3 Network Architecture

For the final submission, we used RAFT architecture presented in [12]. This network has 3 main blocks: (1) a feature encoder that extracts useful features for each pixel; (2) a 4D correlation layer that measures the similarity between all pairs of extracted features; (3) a recurrent update block that incrementally updates the flow fields to calculate the final optical flow. It is inspired by traditional optimization based approaches for estimating human optical flow. The overall structure of the network is given in Figure 1. Next, we discuss each block in the RAFT architecture in detail.

3.3.1 Feature Encoder. This block takes an image $I \in \mathbb{R}^{H \times W \times 3}$ and computes a feature map $g(I) \in \mathbb{R}^{H/8 \times W/8 \times 256}$ at 1/8 resolution. This encoder consists of 6 residual blocks: 2 at 1/2 resolution, 2 at 1/4 resolution, and 2 at 1/8 resolution. Given two input images I_1 and I_2, we first map each image to its corresponding features $g(I_1)$ and $g(I_2)$. We then use a context network h with the same architecture as g to extract context information $h(I_1)$ form the first input image. Then, these three feature sets are given to the correlation and update block to estimate the final flow.

3.3.2 Correlation Block. This block computes the similarity between all pairs of pixels in the feature images. More specifically, given two feature images $g^{(1)}, g^{(2)} \in \mathbb{R}^{H \times W \times D}$, this 4D correlation $C \in \mathbb{R}^{H \times W \times H \times W}$ is given by

$$C_{ijmn} = \sum_{k=1}^{D} g^{(1)}_{ijk} g^{(2)}_{mnk}.$$

Next, we construct a 4-layer pyramid $\{C^1, C^2, C^3, C^4\}$ by pooling the last two dimensions of C with the kernel size of 1, 2, 4, and 8. Therefore, the volume C^k has dimension $H \times W \times H/2^k \times W/2^k$. In this way, we capture both low and high level resolution details.

Correlation Lookup: After computing the pyramid, we perform a correlation lookup by indexing from the pyramid. Given a pixel $\boldsymbol{x} = (u, v) \in I_1$ in the original image and a current estimate of the optical flow (f^1, f^2), we first map $\boldsymbol{x}$ to its corresponding pixel $\boldsymbol{x'} = (u + f^1, v + f^2) \in I_2$. Then, we define a local grid around $\boldsymbol{x'}$ as

$$\mathcal{N}_r(\boldsymbol{x'}) = \{\boldsymbol{x'} + \boldsymbol{dx} \mid \|\boldsymbol{dx}\|_1 \leq r\},$$

and use $\mathcal{N}_r(\boldsymbol{x'})$ to index from the correlation volume.

3.3.3 Update Block. From an initial flow estimate $\boldsymbol{f}_0 = 0$, we perform a series of updates $\boldsymbol{f}_{k+1} = \boldsymbol{f}_k + \Delta \boldsymbol{f}_k$. Given the current estimate $\boldsymbol{f}^k$, we first compute the correlation feature by performing the lookup operation defined above. Then, we process the inputs with two convolutional layers and use the update block to find $\Delta \boldsymbol{f}_k$. We then repeat the process to get a full sequence of flows $\{\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_N\}$. A core component of the update operator is a gated activation unit based on the GRU cell, with fully connected layers replaced with convolutions. We first use a 1×5 convolution to get

$$\begin{aligned}
z_t &= \sigma(\mathrm{Conv}_{1\times5}([h_{t-1}, x_t], W_z)) \\
r_t &= \sigma(\mathrm{Conv}_{1\times5}([h_{t-1}, x_t], W_r)) \\
\widetilde{h}_t &= \tanh(\mathrm{Conv}_{1\times5}([r_t \odot h_{t-1}, x_t], W_r)) \\
\bar{h}_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \widetilde{h}_t.
\end{aligned}$$

Then, we use a similar network with 5×1 convolutions to get the final h_t:

$$\begin{aligned}
z_t &= \sigma(\mathrm{Conv}_{5\times1}([\bar{h}_t, x_t], W_z)) \\
r_t &= \sigma(\mathrm{Conv}_{5\times1}([\bar{h}_t, x_t], W_r)) \\
\widetilde{h}_t &= \tanh(\mathrm{Conv}_{5\times1}([r_t \odot \bar{h}_t, x_t], W_r)) \\
h_t &= (1 - z_t) \odot \bar{h}_t + z_t \odot \widetilde{h}_t.
\end{aligned}$$

3.4 Loss function

For a sequence of predictions $\{\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_N\}$, we define the loss function as

$$\mathcal{L} = \sum_{i=1}^{N} \gamma^{N-i} \|\boldsymbol{f}_{gt} - \boldsymbol{f}_i\|_1,$$

where $\boldsymbol{f}_{gt}$ is the ground truth flow. Hence, this loss measures the L1 distance between the ground truth model and each prediction $\boldsymbol{f}_i$ while giving less weights to initial predictions.

4 EVALUATION

We now discuss the dataset details and the experimental results.

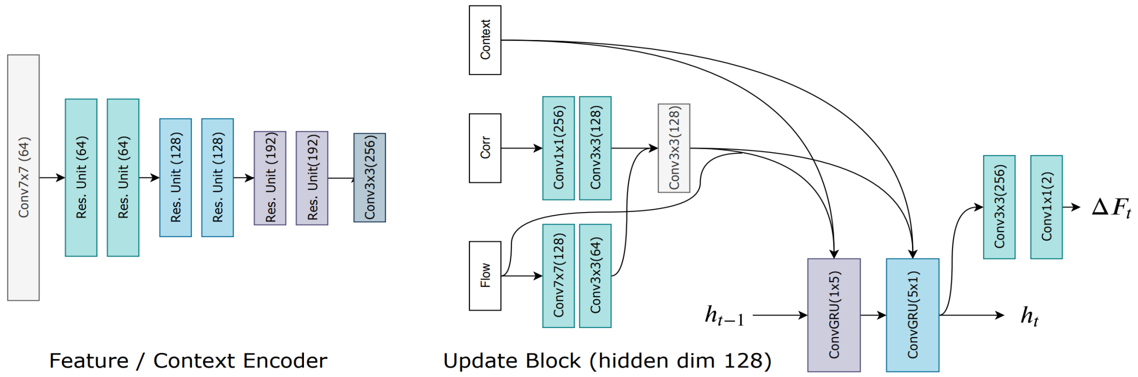

Figure 1: The details of the update block

4.1 Dataset Details

Our dataset is the MHOF that has around 8343 training image pairs. Using the code from [7], we generate 25% more data and finally train our model on 11078 image pairs. The images has a resolution of 640 x 640 so that we are able to reason about optical flow even in small body parts like fingers. It also contains a wide variety of human shapes, poses, actions and virtual backgrounds which simulates the real world examples to some extend.

4.2 Experiment Results

We carry out 4 different experiments to solve the task. The details are given below:

- **PWC-Net**: We initialize the PWC-Net with MHOF pre-trained weights and fine-tune the network on MHOF.
- **RAFT-Small**: It contains less parameters than RAFT model (around 1M compared to 4.8 M in Raft). Parameters are reduced since we take less convolutional kernels in each layer in contrast to RAFT. We initialize the weights with the pre-trained model and fine-tune the network MHOF.
- **RAFT**: We initialize the RAFT model with pre-trained checkpoint, which was trained on the FlyingThings dataset [5], and further fine-tune the network on MHOF.
- **RAFT***: It is our final model and training details are as same as the RAFT mentioned above. However in this experiment, we introduce the data augmentation and additional generated data to train the model. As we can see from Table 1, this data generation and augmentation can achieve the lowest EPE score among the other experiments.

All networks are trained with batch size of 6. We run each experiment for 20 epochs with 0.00005 learning rate which gets reduced to half after every 10 epochs.

Table 1: EPE score of experiments

Methods	EPE score
PWC-Net	0.876
RAFT-Small	0.605
RAFT	0.481
RAFT*	0.437

5 DISCUSSION

Here, we discuss Curriculum Learning as an interesting and novel approach which did not work out well in out case. The main idea of curriculum learning is to first identify the easy and hard training examples so that the network can be trained in different stages. It schedules the data such that the network learns simple examples first and then gradually gets trained on more complex data. We used the pre-trained checkpoint to identify the simple and hard examples according to their EPE score. We used these difficulty tags to fine-tune the model in stages. However, it did not produce significant improvement in our case. We suspect that it could be because our tagging evaluation metrics. The score might improve if we could evaluate difficulty tag based on the number of humans present in the image and the types of movements or use metrics that are based on feature vectors instead of EPE.

6 CONCLUSION

In this project, we worked on human optical flow prediction and learned how to use end to end deep learning based approaches to tackle this problem. We used a RAFT network trained on Flying Things and applied transfer learning to adopt it to the MHOF dataset. Since RAFT can handle the occlusion cases better than other networks such as PWC-Net, it usually results in better EPE scores. In the end, we observed that our RAFT network generally outperforms other architectures such as PWC-Net while having significantly lower number of parameters. In addition, we noticed that data generation and augmentation methods can reduce overfitting and stabilize the training phase. Our work can be improved further with designing a more refined network architecture, hyperparameter tuning, as well as generating more representative data using the current techniques.

REFERENCES

[1] [n. d.]. Carnegie-mellon mocap database.

[2] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. 2004. High Accuracy Optical Flow Estimation Based on a Theory for Warping. In *Computer Vision - ECCV 2004*, Tomás Pajdla and Jiří Matas (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 25–36.

[3] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. 2015. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*. 2758–2766.

[4] Nikolaus Mayer, Eddy Ilg, Philipp Fischer, Caner Hazirbas, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. 2018. What makes good synthetic training data for learning disparity and optical flow estimation? *International Journal of Computer Vision* 126, 9 (2018), 942–960.

[5] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. 2016. A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Jun 2016). https://doi.org/10.1109/cvpr.2016.438

[6] Anurag Ranjan and Michael J. Black. 2016. Optical Flow Estimation using a Spatial Pyramid Network. arXiv:cs.CV/1611.00850

[7] Anurag Ranjan, David T. Hoffmann, Dimitrios Tzionas, Siyu Tang, Javier Romero, and Michael J. Black. 2020. Learning Multi-human Optical Flow. *International Journal of Computer Vision* 128, 4 (Jan 2020), 873–890. https://doi.org/10.1007/s11263-019-01279-w

[8] Javier Romero, Dimitrios Tzionas, and Michael J. Black. 2017. Embodied Hands: Modeling and Capturing Hands and Bodies Together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)* 36, 6 (Nov. 2017), 245:1–245:17. https://doi.org/10.1145/3130800.3130883

[9] L. Sigal, A. Balan, and M. J. Black. 2010. HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision* 87, 1 (March 2010), 4–27.

[10] Deqing Sun, Stefan Roth, and Michael Black. 2014. A Quantitative Analysis of Current Practices in Optical Flow Estimation and The Principles Behind Them. *International Journal of Computer Vision* 106 (01 2014), 115–137. https://doi.org/10.1007/s11263-013-0644-x

[11] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. 2018. PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. arXiv:cs.CV/1709.02371

[12] Zachary Teed and Jia Deng. 2020. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. arXiv:cs.CV/2003.12039