PYRAMID STRUCTURED OPTICAL FLOW LEARNING WITH MOTION CUES

Ji Dai, Shiyuan Huang, Truong Nguyen

University of California, San Diego

ABSTRACT

After the introduction of FlowNet and the large scale synthetic dataset *Flying Chairs*, we witnessed a rapid growth of deep learning based optical flow estimation algorithms. However, most of these algorithms rely on a very deep network to learn both large and small motions, making them less efficient. They also process each frame individually for the video dataset like *MPI Sintel* without using temporally correlated information across frames. This paper presents a pyramid structured network that estimates the optical flow from coarse to fine. We use a much shallower subnetwork at each pyramid level to predict an incremental flow, which contains relatively small motions, based on higher level's prediction. For video dataset, the network utilizes motion cues from previous frames' estimations for assistance. Evaluations show that the proposed network outperforms FlowNet on multiple benchmarks and has a slight edge on other similar pyramid structure networks. The shallow network design shrinks the parameter size by 88% comparing to FlowNet, allowing it to reach almost 100 frames per second prediction speed.

Index Terms— Optical flow, Deep learning, Pyramid Structure, Motion cue

1. INTRODUCTION

Optical flow estimation is a fundamental problem in computer vision with many applications such as autonomous driving, object tracking, 3D reconstruction. Classic approaches often focus on extracting robust features and solving energy minimization problems affiliated with various models [1, 2, 3, 4, 5]. With deep learning making significant contributions to computer vision tasks, researchers started to investigate its usage in flow estimation [6, 7]. Early works mostly replaced the traditional hand-crafted feature detectors with CNN and preserved the energy minimization framework.

In 2015, Dosovitskiy et al. [8] released *Flying Chairs*, a large synthetic flow dataset with ground truth. They also proposed two end-to-end trained networks, namely FlowNetS and FlowNetC, which learned to solve flow estimation from *Flying Chairs* and, surprisingly, generalized well enough on real data like *KITTI*. While they brought a paradigm shift, FlowNet performs not as well compared to traditional methods at that time. FlowNet2.0 [9], the following work by Ilg et al., cascaded multiple FlowNet units and a specifically trained network on a small displacement dataset called "ChairsSDHom". They managed to achieve on-par performance with state-of-the-art methods. Another interesting branch of research is unsupervised learning. Obtaining ground truth flow for real data is extremely laborious. To avoid that, Jason et al. proposed an unsupervised loss combining warping loss as fidelity term and local smoothness as regularization term [10]. Zhu et al. used FieldFlow [7] to estimate a proxy ground truth flow and trained a network to approximate that [11].

We observe that most of current learning based algorithms have very deep network design due to the ambition of learning both large and small motions simultaneously. We argue that such design leads to less efficiency. In fact, even for traditional methods, taking care of multi-scale motions is quite challenging as large motions often induce hefty computations. One practical solution is to decimate images into multiple resolutions to form a pyramid, thus the large motions span only within several pixels at the lowest resolution.

We also argue that, when dealing with video data, using estimated flows from previous frames should be helpful as motions are highly temporally-correlated.

To address previous two arguments, we present a coarse-to-fine pyramid structured network. Inputs to the network include two images, and two motion cues under constant velocity and constant acceleration assumptions generated from previously predicted flows. The network first decimates all inputs into five levels of resolution. At each level, a subnet computes an update flow based on the estimation from previous level. The details are elaborated in section 2.2.

SPyNet [12] by Ranjan et al. has the most similarity with the proposed method. The main differences are: 1) proposed method uses a different subnet design; 2) we incorporate motion cues to improve performance on video dataset.

Leveraging the pyramid structure, we successfully reduce the network size by 88% compared to the FlowNetS [8]. Yet performance-wise, we achieve better scores on multiple datasets as reported in Table 3. Benefiting from the reduced size, the prediction only takes 0.012s per frame. Adding motion cues is also proved to be useful as we gain an additional 4.37% performance on *MPI Sintel* dataset.

This work is supported in part by NSF grant CNS 1456638

2. METHOD

We first explain how to compute motion cues in section 2.1. Network structure is detailed in sections 2.2 and 2.3.

2.1. Computing Motion Cues

Let $I_{t-2}, I_{t-1}, I_t, I_{t+1}$ be four consecutive frames in a video and we are interested in estimating flow map F_t between I_t and I_{t+1}. F_{t-2}, F_{t-1} are estimated flow maps at previous frames.

Note that F_{t-1} indicates the correspondences between pixels in I_{t-1} and I_t, for example, pixel $x_{t-1}(i,j)$ in I_{t-1} is corresponding to pixel $x_t(i + F^y_{t-1}(i,j), j + F^x_{t-1}(i,j))$ in I_t, where F^x, F^y denote horizontal and vertical flow components.

Suppose x_t corresponds to x_{t-1} in I_{t-1} and x_{t-2} in I_{t-2}, we can predict F_t from F_{t-1}, assuming that pixels move in constant velocity:

$$F_t(x_t) = F_{t-1}(x_{t-1})$$

If we assume constant acceleration motions, F_t can be predicted from F_{t-1}, F_{t-2} as:

$$F_t(x_t) = 2F_{t-1}(x_{t-1}) - F_{t-2}(x_{t-2})$$

Using ground truth flow maps F_{t-1} and F_{t-2} to predict F_t for the *MPI Sintel* yields results with 4.502 avg. end-point-error (EPE) for constant velocity model, and 3.400 avg. EPE for constant acceleration model. End-point-error (EPE) is a metric to evaluate flow estimation defined as:

$$EPE = \frac{1}{h \times w} \sum_{i,j} \sqrt{(F^x - \hat{F}^x)^2 + (F^y - \hat{F}^y)^2},$$

where $\hat{F}$ is the ground truth, h, w is the image width and height. Lower EPE means better estimation.

We here report average scores for pixels with predictions. Note that some pixels don't have correspondences at previous frames, and thus cannot be predicted. We observed that both models give better results comparing to the original FlowNet [8], as summarized in Table 3.

Fig.1 shows example results for two prediction models: prediction under constant velocity assumption (denoted as CV) and prediction under constant acceleration assumption (denoted as CA).

2.2. Network Architecture

The inputs to the network are two images and the motion cues from previous prediction as described in section 2.1. The network first decimates inputs into five pyramid levels (resolution from coarse to fine). At each level, downsampled inputs together with an upsampled flow prediction from previous level are fed into a subnet, as shown in Fig.2.

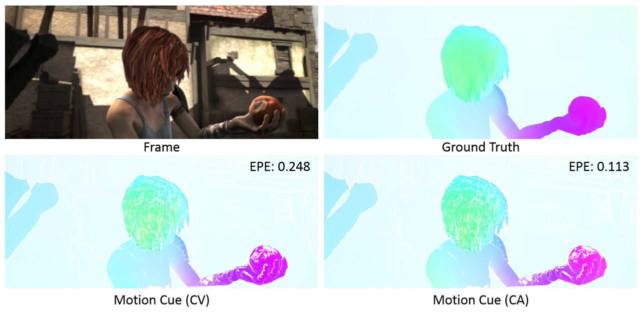

Fig. 1. Example motion cues with EPE scores. Constant acceleration model gives better prediction than constant velocity model.

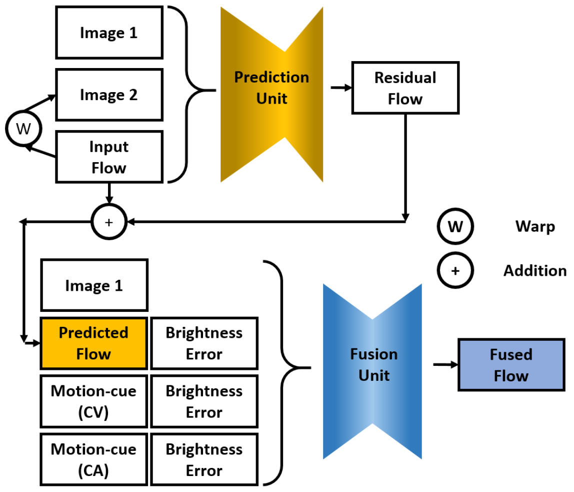

Fig. 3. Subnet architecture: The Subnet contains prediction unit and fusion unit. For static image datasets, subnet outputs predicted flow. For video datasets, subnet outputs fused flow.

2.3. Subnet Architecture

Fig.3 shows the subnet architecture, which contains two parts, a prediction unit and a fusion unit. The prediction unit estimates an incremental flow and adds it to the input flow from previous level to form the predicted flow. For non-video datasets, fusion unit is not activated, thus the subnet only outputs predicted flow.

For video dataset, we stack image 1, predicted flow, motion cues, together with corresponding warping brightness errors, and feed them into fusion unit. The subnet outputs fused flow.

2.3.1. Prediction Unit

Image 2 is first warped with input flow predicted at previous pyramid level. Image 1 and warped image 2 are then stacked as the inputs to the prediction unit. Detailed layer specs are shown in Table 1. We adopt skip connections between con-

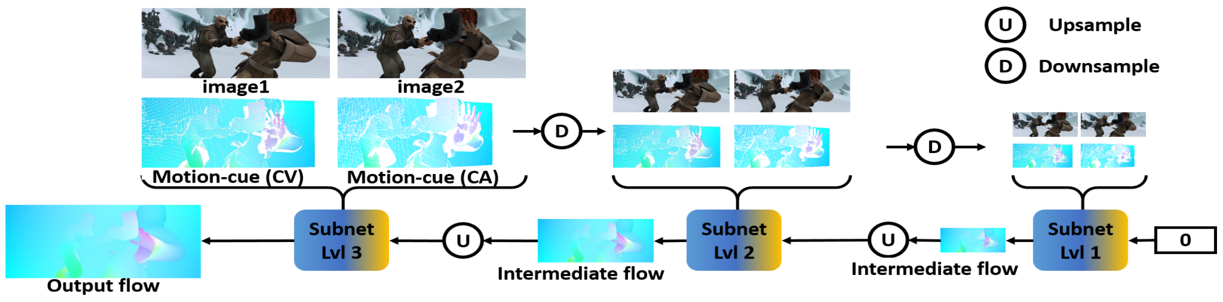

Fig. 2. High level network architecture: We show 3 pyramid levels here for demonstration purpose only, the network used in the paper has five levels in total.

tracting and expanding parts to convey image details, which is quite generic for per-pixel prediction vision problem [13].

Name	kernel	stride	Ch. I/O	Input
conv1_1	3×3	1	6/32	Images
conv1_2	3×3	1	32/32	conv1_1
conv2_1	3×3	2	32/64	conv1_2
conv2_2	3×3	1	64/64	conv2_1
conv3_1	3×3	2	64/128	conv2_2
conv3_2	3×3	1	128/128	conv3_1
upconv4	4×4	2	128/64	conv3_2
upconv3	4×4	2	128/32	upconv4 + conv2_2
upconv2	3×3	1	64/32	upconv3 + conv1_2
pred.	3×3	1	32/2	upconv2

Table 1. The implementation details of prediction unit

2.3.2. *Fusion Unit*

The inputs to fusion unit are image 1, predicted flow, motion cues, and corresponding warping brightness errors. Warping brightness error [9] is computed by first warping image 2 with either predicted flow or motion cue, then subtracting the warped images 2 from image 1. We reduce 3-channel (RGB) error to 1-channel with L2 norm. Warping brightness errors reflect the accuracy of the prediction and motion cues. Detailed layer specs are shown in Table 2.

Name	kernel	stride	Ch. I/O	Input
conv1	3×3	1	12/32	Img1 + flows + errors
conv2	3×3	1	32/64	conv1
conv3	3×3	2	64/128	conv2
upconv3	4×4	2	128/64	conv3
upconv2	4×4	2	64/32	upconv3
fused	3×3	1	32/2	upconv2

Table 2. The implementation details of fusion unit

3. EXPERIMENTS

3.1. Implementation Details

We implemented the network in PyTorch with a Titan Xp GPU. The network was trained using Adam optimizer [14] with $\beta 1 = 0.9, \beta_2 = 0.999$. Each subnet was trained individually from coarse to fine. The weights of trained higher-level subnets were fixed when training the lower-level ones.

We first trained prediction unit on *Flying Chairs* datasets with batch size = 16. The learning rate was set to 1e-4 for the first 50 epochs and reduced by half every 20 epochs after that until the network converged.

We then trained fusion unit on *MPI Sintel* clean dataset with the same batch size. The learning rate was set to 1e-4 at the beginning and reduced to 1e-5 after 50 epochs. Again, the training continued until network converged. Motion cues in training phase were all derived from ground truth flow of previous frames. Weights in prediction unit were fixed when training fusion unit. Both training/validation splits for *Flying Chairs* and *MPI Sintel* followed [8].

At each pyramid level k, the subnet was trained to minimized the average EPE loss L_k between output flow F_k and the decimated ground truth flow $\hat{F}_k$.

To enrich the training data, we applied the following data augmentations: random *translation* with a range of ± 10 in both horizontal and vertical directions; random *rotation* from $\pm 15°$; *Gaussian noise* with σ uniformly sampled from $[0, 0.04]$; random *permutation* of RGB channels; random *scaling* from $[0.8, 2.0]$; random *cropping* of size $[256, 256]$.

3.2. Evaluation on Benchmarks

MPI-Sintel. Since the fusion unit was trained on *MPI Sintel* clean dataset, we performed test on *MPI Sintel* final dataset. To fit in our network, we center-cropped images into 384×1024. The motion cues were all derived from previous estimations, no ground truth was used in any part during the testing. We also did an ablation test with only activating prediction unit to examine the gain from fusion unit.

Kitti2012. We evaluated the network on KITTI 2012

Method	Sintel Clean train	Sintel Final train	KITTI 2012 2012	Middlebury train	Flying Chairs test	Time (s)
FlowNetS [8]	4.50	5.45	8.26	1.09	2.71	0.018
FlowNetC [8]	4.31	5.87	9.35	1.15	2.19	0.032
FlowNet2.0 [9]	2.02	3.14	4.09	0.35	-	0.123
SPyNet [12]	4.12	5.57	9.12	0.33	2.63	0.069
UnsupFlowNet [11]	-	11.19	11.3	-	5.30	-
Pred. unit only	4.08	5.26	7.92	0.39	2.34	0.012
Pred. + Fusion unit	(3.61)	5.03	-	-	-	-

Table 3. Avg. EPE of proposed network compared to other state-of-the-art learning based methods on multiple datasets. We added parentheses as fusion unit is trained on *MPI Sintel* clean. Run times are measured on Flying Chair datasets and excluding data loading time.

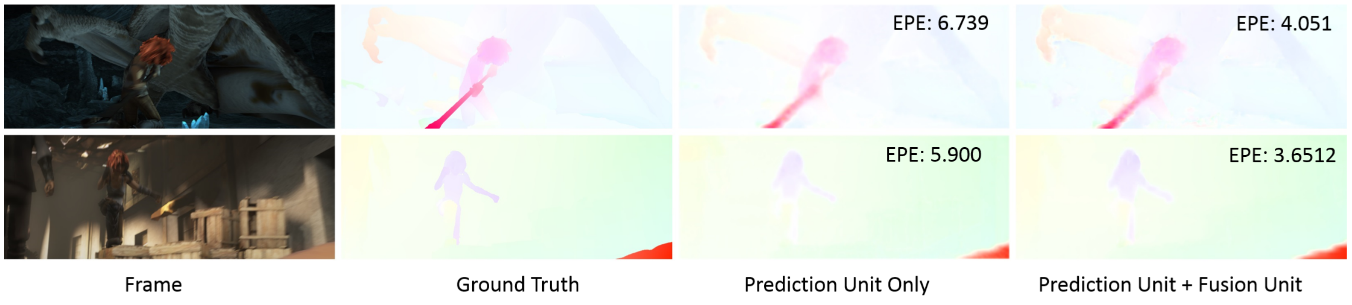

Fig. 5. Examples of optical flow prediction on *MPI Sintel* final dataset. We show both results with/without fusion unit. Fusion unit successfully improves the prediction. Some noise are removed and prediction errors are corrected.

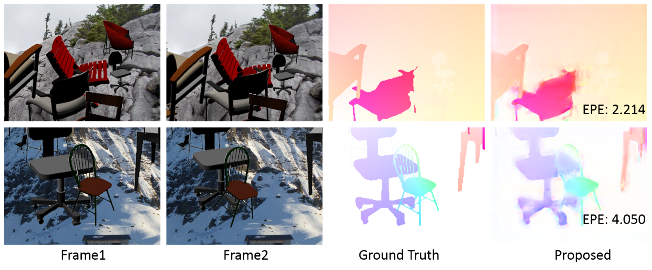

Fig. 4. Examples of flow prediction on *Flying Chairs* dataset.

training dataset. We resized the images to 384 $\times$ 1024 to fit in the network. There was no fine tuning process and we only activated prediction units as this is not a video dataset.

Middlebury. We evaluated on Middlebury flow datasets, the dataset contains 8 pairs of images with ground truth. Again, we only activated prediction units. We resized all images uniformly to 384 $\times$ 640.

3.3. Analysis

Table 3 reports the results of the proposed and other learning based algorithms on selected benchmark datasets. We show some example estimations on *Flying Chairs* and *MPI Sintel* in Fig.4 and Fig. 5. Proposed network achieves better scores than FlowNetS and FlowNetC on most tested dataset. We also perform better than SPyNet on *MPI Sintel* dataset. FlowNet2.0, which is much deeper than our network and partially trained on additional dataset, has the lowest EPE overall.

By using motion cues, we slightly increased the performance on *MPI Sintel* by 11.52% for training and 4.37% for testing. Less gain in testing was expected as the motion cues derived from previously estimated frames contained errors in the first place. Size-wise, the proposed network has a total of 3,815,360 trainable weights. Comparing with 32,070,472 in FlowNetS, our network is 88.1% smaller. The proposed network is also the fastest among all tested networks, reaching approximately 100 frames per second prediction speed on *Flying Chairs* dataset.

4. CONCLUSION

We proposed a compact pyramid structured network that learns to estimate optical flow efficiently. The network uses previous frames' estimations for assistance on video datasets. Evaluations on multiple benchmarks have shown promising results achieved by the proposed network.

5. REFERENCES

[1] Berthold KP Horn and Brian G Schunck, "Determining optical flow," *Artificial intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[2] Bruce D Lucas, Takeo Kanade, et al., "An iterative image registration technique with an application to stereo vision," 1981.

[3] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert, "High accuracy optical flow estimation based on a theory for warping," in *European conference on computer vision*. Springer, 2004, pp. 25–36.

[4] Michael J Black and Paul Anandan, "The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields," *Computer vision and image understanding*, vol. 63, no. 1, pp. 75–104, 1996.

[5] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid, "Epicflow: Edge-preserving interpolation of correspondences for optical flow," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1164–1172.

[6] Jia Xu, René Ranftl, and Vladlen Koltun, "Accurate optical flow via direct cost volume processing," *arXiv preprint arXiv:1704.07325*, 2017.

[7] Christian Bailer, Bertram Taetz, and Didier Stricker, "Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4015–4023.

[8] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2758–2766.

[9] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, vol. 2.

[10] J Yu Jason, Adam W Harley, and Konstantinos G Derpanis, "Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness," in *European Conference on Computer Vision*. Springer, 2016, pp. 3–10.

[11] Yi Zhu, Zhenzhong Lan, Shawn Newsam, and Alexander G Hauptmann, "Guided optical flow learning," *arXiv preprint arXiv:1702.02295*, 2017.

[12] Anurag Ranjan and Michael J Black, "Optical flow estimation using a spatial pyramid network," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, vol. 2.

[13] Jonathan Long, Evan Shelhamer, and Trevor Darrell, "Fully convolutional models for semantic segmentation," in *CVPR*, 2015, vol. 3, p. 4.

[14] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.