SF-Net: Learning Scene Flow from RGB-D Images with CNNs

Yi-Ling Qiao[12]
qiaoyiling15@mails.ucas.edu.cn
Lin Gao[1]
gaolin@ict.ac.cn
Yu-Kun Lai[3]
LaiY4@cardiff.ac.uk
Fang-Lue Zhang[4]
fanglue.zhang@ecs.vuw.ac.nz
Ming-Ze Yuan[12]
yuanmingze@ict.ac.cn
Shihong Xia[1]
xsh@ict.ac.cn

[1] Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences
[2] School of Computer and Control Engineering, University of Chinese Academy of Sciences
[3] School of Computer Science & Informatics, Cardiff University
[4] School of Engineering and Computer Science, Victoria University of Wellington

Abstract

With the rapid development of depth sensors, RGB-D data has become much more accessible. Scene flow is one of the fundamental ways to understand the dynamic content in RGB-D image sequences. Traditional approaches estimate scene flow using registration and smoothness or local rigidity priors, which is slow and prone to errors when the priors are not fully satisfied. To address such challenges, learning based methods provide an attractive alternative. However, trivially applying CNN-based optical flow estimation methods does not produce satisfactory results. How to use deep learning to improve the estimation of scene flow from RGB-D images remains unexplored. In this work, we propose a novel learning based framework to estimate scene flow, which takes both brightness and scene flow losses. Given a pair of RGB-D images, the brightness loss is used to measure the disparity between the first RGB-D image and the deformed second RGB-D image using the scene flow, and the scene flow loss is used to learn from the ground truth of scene flow. We build a convolutional neural network to simultaneously optimize both losses. Extensive experiments on both synthetic and real-world datasets show that our method is significantly faster than existing methods and outperforms state-of-the-art real-time methods in accuracy.

1 Introduction

Fully Connected Layer is a operator where the output feature $\mathbf{Y}$ is densely connected to the input feature $\mathbf{X}$ by a weight matrix $\mathbf{W}$ and a bias $\mathbf{b}$, usually follwed by a non-linear activation function $O(\cdot)$, such that $\mathbf{Y} = O(\mathbf{W}\mathbf{X} + \mathbf{b})$.

Corresponding author: Lin Gao

With the rapid development of depth sensors such as the Microsoft Kinect, RGB-D images become much easier to acquire. Dynamic RGB-D data becomes much more popular recently in computer vision research, due to its important role in perceiving and continuously learning the environment. It has been utilized in diverse applications including scene reconstruction, object tracking, action recognition, etc. Scene flow, first introduced by Vedula *et al.* [21], plays a key role in understanding dynamic RGB-D image sequences, which describes the 3D motion flow of each RGB-D frame. Researchers have proposed several methods to estimate scene flow, such as the superpixel-rigid-based model [23] and the model using moving planar patches [15]. Since an RGB-D frame contains both a color image and a depth image, those works are mainly derived from optical flow estimation methods. Recently, convolutional neural networks (CNNs) are employed to estimate the *optical flow* for videos and show the capability of obtaining high-precision results. However, how to improve *scene flow* estimation from RGB-D images using deep learning remains unexplored. On the other hand, traditional methods are often time consuming, making them unsuitable for certain application scenarios, e.g., a robot in a new environment needs to analyze the scenes in real-time. Thus, we need both efficiency and effectiveness in the scene-flow estimation task.

In this work, we propose a novel CNN-based framework called SF-Net to estimate the scene flow from RGB-D images. The input to our network is a pair of consecutive RGB-D images and the output is the 3D motion flow between them. To take full consideration of the 3D scene flow from both color and depth images, our network has dedicated branches in shallow layers of the network that correspond to image and depth input respectively, due to their different characteristics. To train the network, we optimize a loss function involving both brightness and scene flow errors. The brightness error is used to measure the disparity between the second RGB-D image deformed by the scene flow and the first RGB-D image, and the scene flow error is for learning from training examples. Both losses provide complementary information, and by combining them our approach achieves both efficiency and accuracy.

The main contributions of this paper are: 1) We propose the first end-to-end CNN-based scene flow estimation method for RGB-D images, which achieves real-time prediction while producing high quality results. Our method outperforms classic methods on synthetic datasets by a large margin for both efficiency and accuracy. On real-world datasets, our method also achieves higher accuracy than existing real-time methods. 2) To achieve this, we utilize dedicated convolutional layers for RGB and depth channels and develop a loss function involving both brightness and scene flow errors in our deep architecture, which improves the performance.

The paper is organized as follows: After reviewing related work in Sec. 2, we introduce the structure of our network in Sec. 3 and our loss function in Sec. 4. In Sec. 5 we compare SF-Net with several state-of-the-art RGB-D scene flow methods on both synthetic and real-world datasets, and finally conclusions are drawn in Sec. 6.

2 Related Work

In this section, we first review the related work about scene flow estimation. Then the CNN-based motion estimation methods are reviewed.

Scene Flow Estimation using Traditional Methods. Some pioneer research works estimate the 3D motion flow from images captured using multi-camera systems or RGB-D cameras. In the earlier research works, scene flow is estimated from data collected using

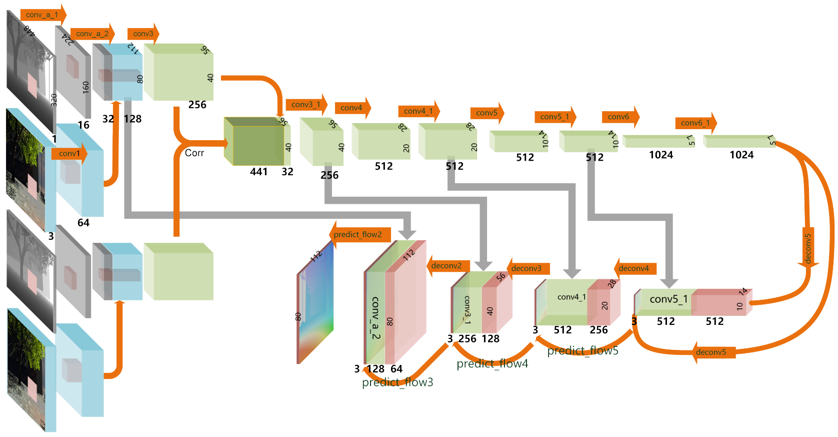

Figure 1: Our SF-Net architecture. For both input RGB-D images $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$, share-weight convolution layers are applied with separate layers for color and depth channels, which are then stacked together. A correlation layer combines branches from $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$, followed by a conv/deconv architecture. Multi-resolution scene flows are predicted along the deconv path, each of which computes loss with downsampled ground truth. In the end, the full resolution scene flow is used to warp 3D point cloud generated from input depth of $\boldsymbol{I}_2$, which is then compared with the original RGB-D image $\boldsymbol{I}_1$ to work out the brightness loss.

a multi-view camera system [21]. Vogel *et al.* [23] propose a superpixel-rigid-based scene flow model which uses view and temporal consistency simultaneously. Menze *et al.* [15] use a set of rigidly moving planar patches to model scenes as a collection of decomposed rigid moving objects. With the development of depth cameras, RGB-D images become much more popular. To estimate the scene flow from such kind of depth images, various methods have been developed. To address the ill-posed problem, the main idea of these works is to add proper priors on the motion. Quiroga *et al.* [17] use local and global rigidity to regularize the 3D motion with piecewise smooth regularization. Hornacek *et al.* [8] assign 6 DoFs (degrees of freedom) to each 3D RGB-D patch, which is further constrained with local rigidity to get the dense scene flow. Sun *et al.* [20] propose a layered RGB-D method to improve scene flow estimation in the occlusion regions. Jaimez *et al.* [10] propose a real-time RGB-D approach which solves for scene flow on GPU. These unsupervised methods based on optimization can produce high quality results if the scene and motion follow the priors, but most of the methods are slow (far from real time) and they may not perform well when the priors are not fully satisfied.

Motion Estimation using CNNs. With the popularity of CNNs, deep learning methods are adopted in various fields of computer vision with state-of-the-art performance. We review papers relevant to our work.

Dosovitskiy *et al.* [4] first introduce an end-to-end CNN for optical flow estimation. Their FlowNet is inspired by Long *et al.* [13], whose Fully Convolutional Network predicts pixel-wise values, and the works [5, 6] that refine results by coarse results and input images. FlowNet extracts features from both frames with a correlation operation and increases the resolution through an upconvolutional architecture. Ilg *et al.*'s FlowNet2 [9] obtains more accurate optical flow by stacking [16] FlowNet and applying warping operations [3], reaching the same level of performance as state-of-the-art methods.

Alternative learning-based approaches combine CNNs with other methods for optical flow estimation. Bailer *et al.* [2] utilize a CNN to extract and match features, which substitute handcrafted features in [1]. Wulff *et al.* [25] improve optical flow estimation by learning a semantic segmentation over static and moving regions, where the former is solved using strong constraints and the latter is unconstrained. Yu *et al.* [11] design an unsupervised model via brightness constancy. Vijayanarasimhan *et al.* [22] predict depth, segmentation, camera and rigid object motions with an unsupervised loss. They recover the motion field and get optical flow by projection. A similar idea is used in our loss function, although our approach does not assume rigid motion masks, and can extract features from RGB-D input, which is popular nowadays.

Fewer methods are designed to estimate scene flow. Mayer *et al.* [14] propose the first CNN-based scene flow estimation method for stereo images and obtain promising results. Their SceneFlowNet combines a FlowNet which predicts optical flow and two DispNet estimating disparities. To train the large CNN end-to-end, they also introduce a large synthetic dataset containing ground truth. However, their problem setting is different from ours since our inputs are RGB-D images. A key point for RGB-D methods is to properly process depth data, *e.g.* Saurabh *et al.* [7] individually extract color and depth features for object detection.

3 Network Architecture

Classic methods often treat scene flow estimation as an optimization problem based on brightness error and some motion model. However, such priors may not be fully satisfied in input data, leading to a performance drop. We therefore propose a learning based approach such that we do not enforce specific prior models but instead learn from training examples. Here, we design an end-to-end CNN model, SF-Net, to estimate the scene flow between two neighboring images in an RGB-D video.

Figure 1 shows the structure of our network. Given a pair of RGB-D images $\boldsymbol{I}_1, \boldsymbol{I}_2 \in \mathbb{R}^{h\times w\times 4}$ as input, where w and h are the width and height of images, we predict scene flow $\boldsymbol{V} \in \mathbb{R}^{h\times w\times 3}$. The loss function consists of two terms: brightness and scene flow errors. The scene flow error measures the difference between predicted scene flow and the ground truth, while the brightness error measures the difference between the warped image $\boldsymbol{I}_2$ using the scene flow and the original image $\boldsymbol{I}_1$. We will give more details on how to compute them in Section 4. In our method, depth and color images are assumed to be aligned, otherwise some alignment algorithms like [27] can be used to preprocess the input.

A straightforward idea of utilizing depth information is to treat depth as an additional channel and stack it with RGB channels in the input layer. However, since depth and color are two heterogeneous data sources, simply treating them in a similar fashion then adding together at the very beginning could not produce good results (see experiments in Table 2). We therefore take another approach which better exploits the depth information by performing convolutions on color and depth data separately to extract intrinsic features and fusing them at later stages.

For clarity, fed with an image pair $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$, the network first convolves depth and color channels *separately*, and then concatenates the features extracted from RGB and depth channels. The operations above are identical for both $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$ by using convolutional layers with shared weights, considering that the analysis on images should be temporally invariant. A correlation layer [4] is then applied to compare feature patches from $\boldsymbol{I}_1$ and

$\boldsymbol{I}_2$ and combine them. Later we downsample feature maps using convolution layers. After reaching the lowest resolution, the network begins to upsample the features using transpose convolutional layers. Every scale also takes the predicted flow and previous convolutional layer as input. At last, the network generates a scene flow map with the same size as the input images.

To fully utilize the depth information, we map pixels in the image domain $\Omega \subset \mathbb{R}^2 \times \mathbb{R}$ into the 3D point cloud domain $\mathbb{R}^3$ with the help of the depth map. The predicted scene flow then transforms generated point clouds. We can get the warped image by projecting point clouds back to the image domain. The per-pixel Euclidean distance between the first image and the second image warped with the flow forms the brightness error in our network.

4 Loss Function

Given ground truth flow, our network can be trained in an end-to-end manner with a loss function composed of two parts, scene flow error and brightness error. The total loss is defined as $L = L_{sceneflow} + \alpha L_{brightness}$, where $L_{sceneflow}$ denotes the scene flow error calculated by comparing prediction with the ground truth, and $L_{brightness}$ is the brightness error produced by warping. α controls the importance of the brightness error term, and is chosen through experiments presented in Table 1. Once trained, our network is able to predict the scene flow given a pair of RGB-D images by a forward pass, which is very efficient (see Table 3).

4.1 Scene Flow Error

In the network, taking the feature map in the lowest resolution, we apply a series of transpose convolutions on it and predict a scene flow map at each layer. In a forward process, we obtain multi-resolution scene flow outputs, which can be compared with down-sampled ground truth flow. In practice, we compute the endpoint error (EPE) between predictions and the ground truth as existing optical flow algorithms do [3, 9]. For prediction with r scales, the scene flow loss is defined as $L_{sceneflow} = \sum_{i=1}^{r} \omega_i \|\hat{\boldsymbol{V}}_i - \boldsymbol{V}_i\|_2$, where ω_i represents the weight for the i^{th} scale, $\hat{\boldsymbol{V}}_i$ is the predicted result, and $\boldsymbol{V}_i$ is the ground truth. We have to notice that a pixel in a lower resolution represents more pixels in the original image, so the accuracy of low resolution is more important. Therefore, we assign a greater weight ω_i for the scene flow error in a lower resolution as parameters in Sec. 5.1.

4.2 Brightness Error

In the real world, ground truth scene flow is nearly impossible to acquire, while end-to-end learning architectures need such data. This causes a major limitation for previous algorithms. Recently, in optical flow estimation, a warping operation along with a brightness error, which is similar to the data term in classical algorithms [11, 22], has been shown to be effective as a loss term. Inspired by this, we introduce the brightness loss by warping images. For a more efficient gradient back-propagation, we warp $\boldsymbol{I}_2$ instead of $\boldsymbol{I}_1$ as FlowNet 2.0 [9].

At the end of our network, a scene flow map $\boldsymbol{V}$ with the same resolution as input is predicted. We use this flow to warp $\boldsymbol{I}_2$ to obtain $\boldsymbol{I}$, where $\boldsymbol{I}$ is the RGB-D image such that when $\boldsymbol{V}$ is applied leads to $\boldsymbol{I}_2$, i.e. the pixel $\boldsymbol{p}_{x,y}$ in $\boldsymbol{I}$ is transformed to the pixel $\hat{\boldsymbol{p}}_{x,y}$ in $\boldsymbol{I}_2$ by predicted scene flow $\boldsymbol{V}$. We further assume that we have known projection parameters

(f_x, f_y, c_x, c_y) (the focal lengths and center point), which are usually available when using RGB-D sensors.

First, we map each pixel $\boldsymbol{p}_{x,y}$ in the warped RGB-D image $\boldsymbol{I}$ to its corresponding point $\boldsymbol{X}_{x,y}$ in 3D space. Second, we translate it to $\hat{\boldsymbol{X}}_{x,y}$ according to the scene flow vector. Third, by projecting the point $\hat{\boldsymbol{X}}_{x,y}$ back to the image plane, we obtain the new position of the pixel $\boldsymbol{p}_{x,y}$ in $\boldsymbol{I}_2$. New coordinates are written as $\hat{\boldsymbol{p}}_{x,y} = (\hat{x}, \hat{y}, \hat{z}) = \boldsymbol{f}(\boldsymbol{p}_{x,y}, \boldsymbol{V}_{x,y})$, such that

$$\hat{x} = \frac{(x-c_x)z}{z+v_z} + \frac{f_x v_x}{z+v_z} + c_x, \quad \hat{y} = \frac{(y-c_y)z}{z+v_z} + \frac{f_y v_y}{z+v_z} + c_y, \quad \hat{z} = z + v_z \tag{1}$$

where $\boldsymbol{f}$ represents the whole process of mapping, translating and projection. Here x, y are the original coordinates in the image plane and z is the depth value. $\hat{\boldsymbol{p}}_{x,y} = (\hat{x}, \hat{y}, \hat{z})$ is the new pixel in $\boldsymbol{I}_2$ corresponding to the pixel $\boldsymbol{p}_{x,y} = (x, y, z)$ in $\boldsymbol{I}$ by the scene flow vector $\boldsymbol{V} = (v_x, v_y, v_z)$.

As the pixel $\boldsymbol{p}_{x,y}$ in $\boldsymbol{I}$ is transformed by scene flow to $\hat{\boldsymbol{p}}_{x,y}$ in $\boldsymbol{I}_2$, we know $\boldsymbol{I}(x,y) = \boldsymbol{I}_2(\hat{x}, \hat{y})$ where $\boldsymbol{I}(x,y)$ denotes the RGB-D value (color and depth) of $\boldsymbol{p}_{x,y}$. However, the transformed pixels $(\hat{x}, \hat{y})$ are not likely to be at integer positions, so we bilinearly interpolate $\boldsymbol{I}_2$ to get continuous $\widetilde{\boldsymbol{I}}_2(x,y)$, which is formulated as

$$\widetilde{\boldsymbol{I}}_2(x,y) = \bar{\theta}_x \bar{\theta}_y \boldsymbol{I}_2(\lfloor x \rfloor, \lfloor y \rfloor) + \theta_x \bar{\theta}_y \boldsymbol{I}_2(\lceil x \rceil, \lfloor y \rfloor) + \bar{\theta}_x \theta_y \boldsymbol{I}_2(\lfloor x \rfloor, \lceil y \rceil) + \theta_x \theta_y \boldsymbol{I}_2(\lceil x \rceil, \lceil y \rceil). \tag{2}$$

where $(\theta_x, \bar{\theta}_x, \theta_y, \bar{\theta}_y)$ denotes the coefficients of interpolation, i.e. $\theta_x = x - \lfloor x \rfloor$, $\bar{\theta}_x = 1 - \theta_x$, $\theta_y = y - \lfloor y \rfloor$, $\bar{\theta}_y = 1 - \theta_y$. In case the motion vector is large and the new pixel $(\hat{x}, \hat{y})$ is out of boundaries, we assign zero value to $\boldsymbol{I}(x,y)$. Therefore, we obtain the forward pass as

$$\boldsymbol{I}(x,y) = \begin{cases} \widetilde{\boldsymbol{I}}_2(\hat{x}, \hat{y}) & \text{if } (\hat{x}, \hat{y}) \in \boldsymbol{\Omega} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where Ω is the defined image domain. With the warped $\boldsymbol{I} = warp(\boldsymbol{I}_2, \boldsymbol{V})$, the brightness loss is defined as $L_{brightness} = \|\boldsymbol{I} - \boldsymbol{I}_1\|_2$. Note that, both $\boldsymbol{I}$ and $\boldsymbol{I}_1$ have RGB-D channels. In the network, each channel is normalized so that the weights of the four channels are equal in this case. This loss assembles the data consistency in traditional methods.

Since bilinear interpolation is differentiable, we can pass the gradients from $\boldsymbol{I}$ to $\widetilde{I}_2$ and calculate gradients on $\boldsymbol{V}_{x,y} = (v_x, v_y, v_z)$, based on the forward-pass equation (Eq. 1). Take v_x for example,

$$\begin{aligned} \frac{\partial \widetilde{\boldsymbol{I}}_2(\hat{x}, \hat{y})}{\partial v_x(x,y)} &= -\frac{f_y}{z+v_z} \bar{\theta}_y \boldsymbol{I}_2(\lfloor x \rfloor, \lfloor y \rfloor) + \frac{f_y}{z+v_z} \bar{\theta}_y \boldsymbol{I}_2(\lceil x \rceil, \lfloor y \rfloor) \\ &\quad - \frac{f_y}{z+v_z} \theta_y \boldsymbol{I}_2(\lfloor x \rfloor, \lceil y \rceil) + \frac{f_y}{z+v_z} \theta_y \boldsymbol{I}_2(\lceil x \rceil, \lceil y \rceil). \end{aligned} \tag{4}$$

5 Experiments

In this section, we first describe the implementation details and parameters of SF-Net. Further tests on variants of SF-Net show the effects of network components and loss function terms. In the end, we compare our SF-Net qualitatively and quantitatively with other methods on several datasets.

In the following experiments, the scene flow error is measured in the image domain using standard root mean square error (RMS) between the predicted and ground truth flow, which is commonly used in motion flow estimation [11, 24]. To analyze the effect of the weight α for the brightness loss, we experiment several alternatives and list the results in Table 1. $\alpha = 0.002$ gives the best performance for both training and testing and thus is used by default in our pipeline.

Brightness Loss	$\alpha = 0$	$\alpha = 0.001$	$\alpha = 0.002$	$\alpha = 0.005$
Test	7.84	6.09	**5.03**	5.28
Training	5.26	4.81	**4.22**	4.65

Table 1: Different weights for the brightness loss. We conduct a series of experiments on Monkaa and Driving [14] to find a proper weight for the brightness loss. In our experiments, the weight $\alpha = 0.002$ achieves the best performance for both training and testing. In addition, compared with $\alpha = 0$, we can conclude that the introduction of the brightness loss improves the performance of the network.

5.1 Implementation Details

Figure 1 depicts the basic structure and parameters of SF-Net. Similar to FlowNet [4], we use a conv/deconv architecture with stride 1 and stride 2. The loss weights are $\omega = [0.32, 0.08, 0.02, 0.01, 0.005]$ and $\alpha = 0.002$.

Our implementation is based on the TensorFlow framework. We performed experiments on a PC with an Intel Core i7-2600 CPU and an NVIDIA TitanX GPU. We make use of the Adam optimizer [12] with Adam parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ as in [12]. We follow the training schedule of [18]. The network was first trained on Flytingthings3D for 200,000 iterations, and then fine-tuned on Monkaa and Driving [14] for 100,000 iterations.

5.2 Performance of Different Networks

In order to illustrate effects of different components and loss terms in our network, we train and test them on the Monkaa [14] dataset. Monkaa is a large scene flow dataset, in which frames are rendered from various synthetic image sequences. It has accurate depth maps and ground truth scene flow so as to give a reasonable comparison between algorithms under ideal conditions. In total, we compare our SF-Net with three variants. Recall that our SF-Net has branches for RGB and depth input, followed by a body part of FlowNetC, and a warping layer at the end. To test the performance of FlowNetS and FlowNetC [4] architectures in RGB-D scene flow estimation, we simply expand the number of output channels from 2 to 3 and add depth as another input channel, without using branches in the beginning. Accordingly, these two modified networks are named as FlowNetS+RGB-D and FlowNetC+RGB-D. The third network SF-noWarping is the lite-version of SF-Net without brightness loss, and SF-brightness is trained with only brightness loss. Figure 2 shows qualitative results.

From Table 2, simply adapting FlowNet to scene flow prediction will cause a higher error, showing that separated convolution for depth information indeed helps with 3D scene flow estimation. Furthermore, FlowNetC outperforms FlowNetS, which indicates that the correlation layer contributes to this task as well. We also observe that supervision is important in this task, since training with only brightness error has poor performance. The last column shows that our approach with separated convolutions and brightness loss improves

Method		FlowNetS+RGBD	FlowNetC+RGBD	SF-noWarping	SF-brightness	SF-Net
RMS	Test	17.24	13.81	10.82	92.64	**5.03**
	Training	13.65	9.27	8.83	70.81	**4.22**
Time		**16ms**	18ms	20ms	23ms	23ms

Table 2: Architecture test. We show the root mean square error (RMS) during training and testing on the Monkaa dataset [14]. All the networks can run in real-time. Our SF-Net gets the least error, while the network trained with only brightness error has worse performance. We conclude that depth information is important for this task.

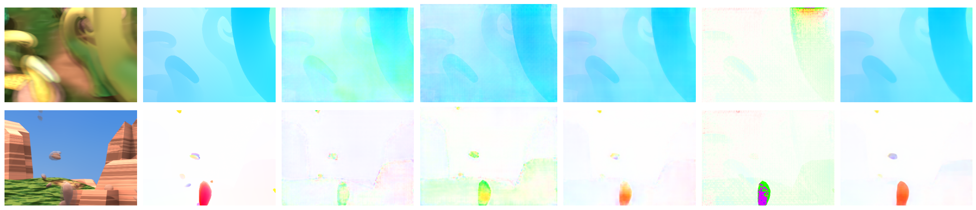

Figure 2: Scene flow prediction on a synthetic dataset (Monkaa [14]). We show different variants of our network and direct generalization of FlowNet [4], (a) input, (b) ground truth scene flow, (c) result of FlowNetS+RGB-D, (d) result of FlowNetC+RGB-D, (e) SF-noWarping (SF-Net without the warping layer), (f) SF-brightness (trained with only brightness error), (g) our proposed SF-Net. We can see that SF-Net with 3D warping and separated convolutions achieves the best result.

the performance and achieves the best result among all these architectures. Moreover, the *Time* row indicates that all the architectures tested can run in realtime due to the end-to-end CNN structure, which shows potential for many applications.

5.3 Comparison with Previous Methods

When compared with previous methods for scene flow estimation [10, 17, 20], SF-Net not only achieves better accuracy, but also predicts scene flow in much shorter time. For example, the running time of [20] can reach up to 4 minutes per frame, making it difficult to apply in realtime applications. By contrast, our method only needs 0.02s for a pair of frames. In this section, we choose Quiroga *et al.* [17], Sun *et al.* [20] and another real-time algorithm PD-Flow [10] to compare with our method. For all four approaches, we test on synthetic datasets, real-world data and Kinect data. The qualitative results are shown in Figure 3, where scene flow is projected onto the image plane and visualized as its optical flow maps.

Monkaa & Driving. Similar to Sec. 5.2, we test on synthetic datasets Monkaa and Driving since they involve non-rigid motion and have ground truth scene flow. We randomly divide the RGB-D frames contained in Monkaa and Driving into training and testing groups by a ratio of 9:1. Columns 5 and 6 in Figure 3 show the results of an example. RMS errors of different methods are reported in Table 3 (rows 1-2) which shows our method is much more accurate and faster than existing methods.

Middlebury Dataset. We run our SF-Net on Middlebury 2003 [19]. It is commonly

Method		PD-Flow [10]	SRSF [17]	Sun *et al.* [20]	SF-Net
RMS	Monkaa	43.62	21.81	19.54	**4.91**
	Driving	35.80	16.65	15.38	**9.30**
	Cones	8.26	0.49	**0.09**	0.86
	Teddy	7.32	0.45	**0.17**	1.14
Time		0.12s	120s	240s	**0.02s**

Table 3: Comparison with previous work. By measuring root mean square error (RMS) in multiple datasets, we compare our SF-Net with existing RGB-D scene flow methods [10, 17, 20]. Our SF-Net achieves the highest accuracy with the shortest runtime in scenes with non-rigid motion, while optimiazation-based methods perform better in rigid scenes such as Cones and Teddy [19].

used to evaluate RGB-D scene flow methods. The RMS errors are in Table 3 (rows 3-4: Cones & Teddy). Motion in Middlebury Dataset is almost rigid, so some classic algorithms with strong priors achieve impressive results. Our method does not have any explicit motion prior, but still works reasonably well. Furthermore, our method is also much faster than alternative methods.

Microsoft Kinect Data. We now investigate to what extent our learned SF-Net model generalizes to real-world data with noise. We use RGB-D frames captured by Yuan *et al.* [26] as examples in Figure 3, where the motion is mostly non-rigid. Although no ground truth is available, our method produces smooth and visually plausible results.

6 Conclusion

In this paper, we have proposed the first CNN-based learning framework, SF-Net, for RGB-D scene flow estimation. In SF-Net, we use dedicated convolutional layers for RGB and depth channels, and then fuse the extracted intrinsic features at later stages. We also develop the loss function involving both brightness and scene flow error terms, which can utilize the ground truth scene flow and the inherent relationship between the 3D point cloud and RGB-D images respectively to guide the model training. The above characteristics are shown to make our model more effective in the scene flow estimation task. In the experiments on both of the synthetic and real world datasets, our method outperforms state-of-the-art RGB-D scene flow estimation methods. Moreover, our computation cost is much less than previous methods, making it suitable for real-time applications using RGB-D data.

Acknowledgement

This work was supported by grants from the National Natural Science Foundation of China (No. 61502453 and No. 61772499), Young Elite Scientists Sponsorship Program by CAST (No. 2017QNRC001), Royal Society-Newton Mobility Grant (No. IE150731), and New Zealand National Science Challenge 10: "Science for Technological Innovation" - Robotics Spearhead Project (Callaghan Innovation: CRS-S6-2017).

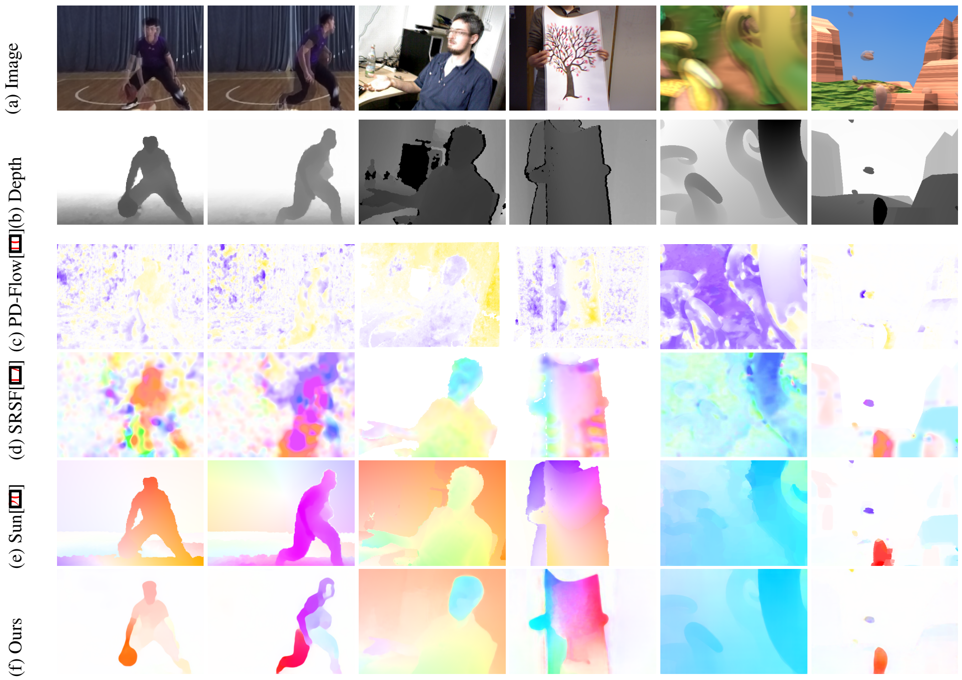

Figure 3: Comparison with previous work on synthetic and real-world data. We test (c) PD-Flow [10], (d) SRSF [17], (e) Sun *et al.* [20] and (f) our SF-Net on Kinect data [26], real-world data [17] and synthetic data. This figure depicts the predicted scene flow projected into the image domain. Compared to alternative methods, our prediction tends to be smoother and more robust.

References

[1] 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*, pages 4015–4023, 2015.

[2] Christian Bailer, Kiran Varanasi, and Didier Stricker. Cnn-based patch matching for optical flow with thresholded hinge embedding loss. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3250–3259, 2017.

[3] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. *Proceedings of the European Conference on Computer Vision*, pages 25–36, 2004.

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

[5] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 1538–1546. IEEE, 2015.

[6] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[7] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *Proceedings of the European Conference on Computer Vision*, pages 345–360. Springer, 2014.

[8] Michael Hornacek, Andrew Fitzgibbon, and Carsten Rother. Sphereflow: 6 dof scene flow from rgb-d pairs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3526–3533, 2014.

[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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Jul 2017.

[10] Mariano Jaimez, Mohamed Souiai, Javier Gonzalez-Jimenez, and Daniel Cremers. A primal-dual framework for real-time dense rgb-d scene flow. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 98–104. IEEE, 2015.

[11] 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 *Computer Vision–ECCV 2016 Workshops*, pages 3–10. Springer, 2016.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Computer Science*, 2014.

[13] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[14] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4040–4048, 2016.

[15] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3061–3070, 2015.

[16] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *Proceedings of the European Conference on Computer Vision*, pages 483–499. Springer, 2016.

[17] Julian Quiroga, Thomas Brox, Frédéric Devernay, and James Crowley. Dense semi-rigid scene flow estimation from rgbd images. In *Proceedings of the European Conference on Computer Vision*, pages 567–582. Springer, 2014.

[18] Anurag Ranjan and Michael J. Black. Optical flow estimation using a spatial pyramid network. 2016.

[19] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–I. IEEE, 2003.

[20] Deqing Sun, Erik B Sudderth, and Hanspeter Pfister. Layered rgbd scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 548–556, 2015.

[21] Sundar Vedula, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):475–480, 2005.

[22] Sudheendra Vijayanarasimhan, Susanna Ricco, Cordelia Schmid, Rahul Sukthankar, and Katerina Fragkiadaki. Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*, 2017.

[23] Christoph Vogel, Konrad Schindler, and Stefan Roth. 3d scene flow estimation with a piecewise rigid scene model. *International Journal of Computer Vision*, 115(1):1–28, 2015.

[24] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *IEEE International Conference on Computer Vision*, pages 1385–1392, 2014.

[25] Jonas Wulff, Laura Sevilla-Lara, and Michael J Black. Optical flow in mostly rigid scenes. *arXiv preprint arXiv:1705.01352*, 2017.

[26] Ming-Ze Yuan, Lin Gao, Hongbo Fu, and Shihong Xia. Temporal upsampling of depth maps using a hybrid camera. *IEEE Transactions on Visualization and Computer Graphics*, 2018.

[27] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.