Deep Learning Architectures for Estimating Optical Flow

Seminar "Pattern Recognition", 710.103

Peter LORENZ

Inst. of Computer Graphics and Vision
Graz University of Technology, Austria

Seminar/Project Computer Vision
Dr.techn. Peter Roth
Graz, April 9, 2018

contact: Peter LORENZ peter.lorenz@student.tugraz.at

Abstract

Optical flow describes the motion of pixels through an image sequence. An image sequence can be obtained from different applications, such as autonomous driving vehicles or robots. Today's deep learning approaches are still lacking behind classical variational methods. Computational expensiveness and no sharp edges between flow fields are the shortcomings of state-of-the-art deep learning convolutional networks. To solve these problems, various deep learning approaches make use of hybrid models, where variational methods are used in post-processing. We present an overview of state-of-the art deep learning networks (supervised and unsupervised) and analyze their architecture as well their results.

Keywords: *Optical Flow, Stereo Vision, Deep Learning, Optimization, Pixel-Wise Prediction*

1 Introduction

Deep Learning has replaced classical approaches in many computer vision tasks, because of its overwhelming results. Optical flow is one task that is dominated by classical approaches. Due to the demand of autonomous driving vehicles a better vision system is needed to achieve better results in image segmentation. A scene can be decomposed by several independently moving objects and this helps for several tasks like collision detection. We want to show the connection between deep learning approaches and classical approaches. Thus, we present state-of-the-art deep learning architectures in chronological order and explain some basic assumptions that holds for classical as well deep learning approaches. After decomposing the architectures and their connection to classical approaches, we want to discuss the results in the end.

1.1 Background and Motivation

Optical flow was originally introduced by a paper from Horn and Schunk [1] and is the fundamental work of many ongoing research topics. Almost every classical computer vision method was replaced by deep deep convolutional neural networks (DCNN), with the reasons being the concisely better results with DCNN. Except for optical flow, there still is not a single deep learning architecture which is significantly better than classical methods. A supervised learning [2] approach has shown that optical flow can be estimated, but variational methods still yield better results. Last year, another deep learning approach [3] was published; it lags slightly behind state-of-the-art performance algorithms.

1.2 Ranking in MiddleBury and Sintel Dataset

The Table 1 shows a ranked list of different optical flow algorithms. Only for this paper interesting algorithms were selected to show the gap between classical and deep learning approaches.

The Middlebury dataset is small and both FlowNets yield bad results. Results of the other DCNNs have not been published by [4]. Only 8 frames are provided per scene and the frame sizes differs per scene.

On a larger dataset as given by Sintel in Table 2, the FlowNet architecture gives a lot better results than on the small dataset given by Middlebury in Table 1. Sintel provides more than 1000 frames with the same image size (1024×436) in each scene.

Rank	Name	EPE	Type
1	NNF-Local	3.5	Nearest Neighbor, PAMI
2	PMMST	9.5	Minimal Spanning Tree
3	OFLAF	10.2	Minimal Spanning Tree
...	...	...	...
94	FlowNetS+ft+v	84.5	Supervised Deep Learning
96	FlowNet 2.0	85	Supervised Deep Learning

Table 1: Selected algorithms of Middlebury Dataset [4].

Rank	Name	EPE	Type
1	PWC-Net	5.042	Unknown
2	DCFlow	5.119	Direct Cost Volume Processing.
3	FlowFieldsCNN	5.363	Minimal Spanning Tree
...	...	...	...
18	FlowNet2-ft-sintel	5.739	Supervised Deep Learning
24	FlowNet2	6.016	Supervised Deep Learning
44	FlowNetS+ft+v	7.218	Supervised Deep Learning
...	...	...	...
95	Horn + Schunk	9.61	Variational Method

Table 2: Selected Algorithms from Sintel Dataset [5]

1.3 Applications

Optical Flow belongs to the low-level computer vision problems and therefore a number of different application areas can be listed:

- Computer Vision: video segmentation, object tracking, structure from motion, SLAM [4]
- Computer Graphics: video synthesis such as faces, fluid reconstruction
- HCI: hand gestures, facial expressions, pose estimation

2 Optical Flow

Optical flow is generally the estimation of object motions in a scene. This motion is represented as a vector field. The direction of each vector is pointing, shows in which direction an object is moving.

For determining optical flow in image processing a sequence of (at least 2) images is needed, because optical flow algorithms estimate motions of pixels in consecutive image frames as they move from one to another frame. However, the full optical flow equation is given by one equation and two unknown (x and y direction). More details in Section 2.1.2. In classical computer vision approaches, there will be in the second image looked for the pixel in the first image in the local neighborhood. Therefore, some constrains can be assumed, constant pixel brightness and only small movements happened from one frame to another.

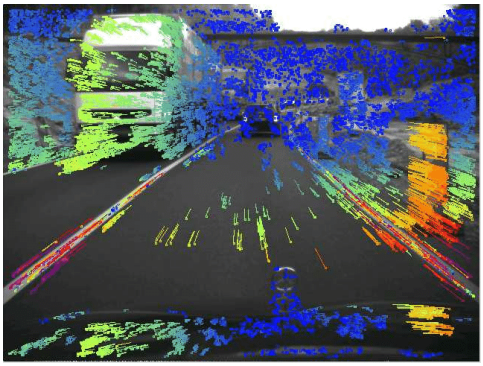

Figure 1: Optical flow vectors with oncoming traffic [6].

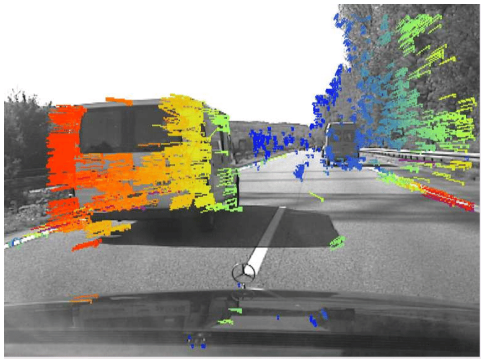

Figure 2: Optical flow vectors by following a car [6].

Best illustrating example is driving a car, where the flow vectors show the direction of pixels are moving. As in Figure 1 and Figure 2 it can be observed the flow field vectors. The colors denote the direction of the vector, which is explained in the Section 5.2. The higher the intensity, the more

the vector shows in a direction. In cities, optical flow can help for detecting pedestrians, because they would suddenly come from left or right into the frame and continue walking. Autonomous vehicles could react by estimating the flow, predicting the there will be a pedestrian in front of the car and stops the vehicle.

2.1 Assumptions and Constraints Holding for Optical Flow

Three assumptions/constraints are necessary to describe the optical flow in order to restrict the definition, so that the methods such as [1] can build algorithms on it. These assumptions/constraints still can be found in todays machine learning approaches.

2.1.1 Brightness Constancy Assumption (BCA) and Small Displacement

For computing the optical flow, some assumption must be made fulfilled to derive some mathematical laws. First of all, if we focus only on one scene point, we assume that the brightness of this one scene point stays the same, just the location is different.

Assumption 1-$\mathcal{F}$. *The brightness of this scene point does not change at time t:*
$I(x,t) = Constant$, *where* $x = \binom{x_1}{x_2}$ *is the pixel position and the displacement vector* $\triangle x = \binom{\triangle x_1}{\triangle x_2}$ *that shows a shift of a pixel.*

Assumption 2-$\mathcal{F}$. *Assuming* $\triangle x$ *is small (less than a pixel, or smooth), then* $I(x,t) \rightarrow I(x+\triangle x, t+\triangle t)$. *Thus, we can made 1st order Taylor expansion and obtain to the linearized BCA. t or t+1 the derivative yields the same.*

2-$\mathcal{F}$ can be proved the 1st order Taylor expansion by setting the pixel brightness at time t and $t+1$ equal.

$$
\begin{aligned}
0 &= I(x+u, t+1) - I(x,t)\ Brightness\,Constancy\,Equation \\
0 &\approx I(x, t+1) + I_x u - I(x,t) \\
0 &\approx [I(x,t+1) - I(x,t)] + I_x u = \frac{\partial I}{\partial x} \qquad (1) \\
0 &\approx I_t + I_x u \\
0 &= I_t + \nabla_x I \cdot u \qquad \square
\end{aligned}
$$

2.1.2 Optical Flow Constrained (OFC)

Let u be the velocity vector $\binom{u_1}{u_2}$. In practice, we assume the time difference $\triangle_t = 1$ s.t. the velocity is equal to the displacement ($u = \triangle_x$) and we assume 2 images of a sequence are given image I_1 and image I_2. Let u_0 a flow field that is thought as ground truth, we want to minimize such that u is the approximated flow field. The OFC is differential, but he flow field u_0 can be approximated via a warped image I_w and geometrically transform I_2 by u_0:

$$I_w(x) = I_2(x + u_0). \tag{2}$$

Apply Taylor to the above BCA and defining $I_t = I_w - I_1$ the OFC can also be written as

$$(\nabla_w(x))^T(u - u_0) + I_t(x) = 0, \tag{3}$$

what we want to minimize.

2.1.3 Spatial Coherence Assumption

For this assumption is it still difficult to find a good model. If a neighboring pixel is also the neighbor of another flow field, which directs into the opposite direction.

Assumption 3-$\mathcal{F}$. *Neighboring pixels have the same motion.*

2.2 Global Method by Horn and Schunck to determine Optical Flow

Horn and Schunk [1] introduced a global method to estimate the optical flow. It is a fundamental work for many following methods, e.g. [7] used for loss layers, see Section 3.3.3. There is an incorporate spatial coherence of the flow field $u = \begin{pmatrix} u(x) \\ v(x) \end{pmatrix}$ by minimizing a global model:

$$\min_{u,v} \int_\Omega |\nabla u(x)|^2 + |\nabla v(x)|^2 + \lambda |(\nabla I_w(x))^T(u - u_0) + I_t(x)|^2 dx. \tag{4}$$

Parameters: λ is a regularization constant and the larger λ is chosen the smoother the flow. $D = \binom{D_x}{D_y} \in \mathbb{R}^{2MNxMN}$ is the discretized gradient vector. $U, V \in \mathbb{R}^{MN}$ are the components of the flow field and that what we want

to minimize. $U_0, V_0 \in \mathbb{R}^{MN}$ is the initial solution. $I_w^{\{x,y\}} = diag(D_{\{x,y\}} I_w)$ are the finite difference approximation of the partial derivatives of I_w. The continuous model is not suitable for computer vision tasks, so in further steps an equivalent discrete model can be used:

$$\min_{U,V} \; ||DU||^2 + ||DV||^2 + \lambda ||I_w^x (U - U_0) + I_w^y (V - V_0) + I_t||^2. \tag{5}$$

Optimality conditions for U and V:

$$\begin{aligned} D^T DU + \lambda I_w^x I_w^x (U - U_0) + I_w^y (V - V_0) + I_t = 0 \\ D^T DV + \lambda I_w^y I_w^x (U - U_0) + I_w^y (V - V_0) + I_t = 0. \end{aligned} \tag{6}$$

The optimality conditions can be rewritten into matrix notation.

$$\begin{pmatrix} D^T D + \lambda (I_w^x)^2 & \lambda I_w^x I_w^y \\ \lambda I_w^x I_w^y & D^T D + \lambda (I_w^y)^2. \end{pmatrix} \begin{pmatrix} U \\ V \end{pmatrix} = \lambda \begin{pmatrix} I_w^x (I_w^x U_0 + I_w^y) V_0 - I_t \\ I_w^y (I_w^x U_0 + I_w^y) V_0 - I_t \end{pmatrix} \tag{7}$$

The system is large (2MN x 2MN) and reminds to the matrix from the equation of the Harris corner detector, which also reacts on intensities changes. It is very sparse and can be solved efficiently by conjugate gradient (CG). One of the drawbacks are there are no sharp edges between flow fields, because of the euclidean distance and outliers cannot not be handled robustly.

2.3 Challenges of Optical Flow

One of the challenges of optical flow is to predict sharp edges as show in Figure 3. Classical Computer Vision approaches as listed in the tables Table 1 and Table 2.

Another ongoing challenge is the drawback of end-to-end architectures, which suffers often from noise over smooth areas. There are some post processing or refinement methods as recently introduced [8]. Pock formulated a reaction diffusion model [8] and can apply it to image image denoising tasks.

3 Deep Learning Architectures for Predicting Optical Flow

In this section, some deep learning approaches are chosen for predicting optical flow and are described layer by layer. One is a supervised learning approach FlowNet and FlowNet 2. Supervised learning means that each training example is a pair (x, l) input data x and its label l (e.g. $l \in \{optical\ flow,\ no\ optical\ flow\}$). The second one is based on UnFlow [9], a supervised approach, and does not include any labels to get feedback by a ground truth. UnFlow was recently published and is also treated shortly.

Figure 3: Comparison of FlowNet with FlowNet2 [2]

Convolutional Neural Network convolutional neural networks (CNN) are very similar to neural networks: they consist of neurons, Figure 5 that have learn-able weights and biases. Neurons receive an input, that performs a dot product Figure 4 and optionally follows it with a non-linearity (tanh, ReLU, Leaky ReLU, sigmoid). The network has a forward path, and a backward path, which is the 1st order derivation of the forward path. The backward path is responsible for the learning process, where weights and bias are adapted. This process is called backpropagation. The hidden layers are all layers in between first and last layer, are usually not fully connected which scales down the huge number of parameters. The last layer have a loss function (e.g. Softmax) and is always fully connected.

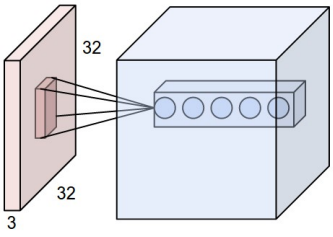

Figure 4: Kernel window over a layer, where the dot product is applied.

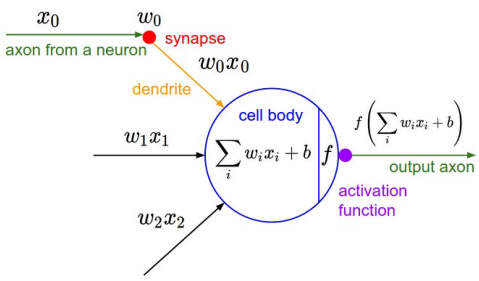

Figure 5: Neuron with 3 inputs.

In common practice, not only same hidden layers are stacked together as well different types of layers (convolutional, pooling, fully-connected, dropout layer) or different activation functions different parameters as kernel size are used. Pooling layers (more details: Section 3.1.2) are used to downsample the resolution of the image.

3.1 Supervised Deep Learning Approach: FlowNet and FlowNet2

FlowNet [10] was the first approach, which showed "optical flow" as learning problemFowNet2 [2] is the improved version and increases up to 50 percent accuracy of estimating optical flow and is competitive variational approaches, those have been dominating optical flow estimation since the publication of Horn and Schunck [1].

Dosovitskiy et al. [10] used the idea of Zuntar and LeCun [11] by training a siamese architecture (two input consecutive images, mostly chosen from stereo cameras setup or from a scene or videos) predict the similarities of image patches and altered it instead of patches the whole images, so that the flow field can be directly computed. End-to-end network structures are the first choice of the recent two or three years, but suffer mostly of noise and some post processing is necessary, which are will be discussed in the following section.

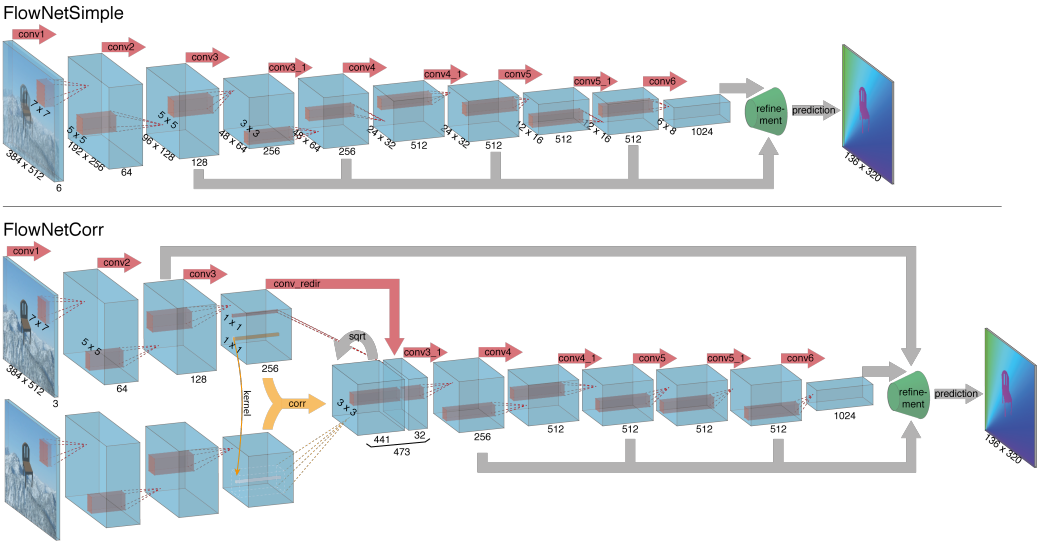

Figure 6: The network architectures: FlowNetSimple [10] (above) and FlowNetCorr [10] (underneath)

3.1.1 Correlation Layer

A correlation layer can be used as an hidden layer and improves the matching process, where each patch from f_1 is compared with each patch from f_2:

$$f_1, f_2 = \mathcal{R}^2 \rightarrow \mathcal{R}^c. \tag{8}$$

The Figure 7 restricts only on the comparison of 2 patches, where x1 and x2 are the centers of the first and second map and a patch size has $D = 2d + 1$ [10]. Moreover, it describes (Figure 7) a sliding dot product, but instead convolving with a filter (e.g. Sobel, Median), data of f_1 are convolved with data of f_2. Comparing these feature vectors gives a higher peak when sliding within the window, so that the a similar pixel can be found. The assumption of small displacement holds as well the BCA (see Section 2.1.1). Dosovitskiy et al. tries not to compare all patches $w^2 \cdot h^2$ and is computational expensive [10]. Therefore, a maximal displacement d [10] is introduced, which compares pixel in maximum distance of d in the neighborhood (10 pixels) [10]. Then, feature map is concatenated as one volume.

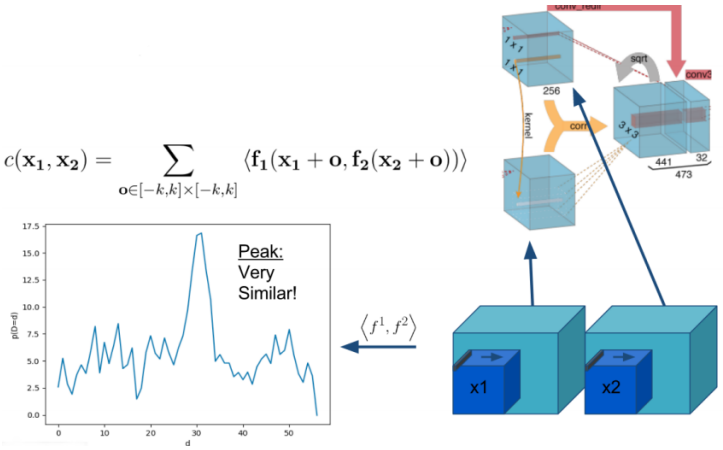

Figure 7: Correlation Layer [2] (right top) and comparison of the features

3.1.2 Pooling and Refinement Layer

After the concatenation, we have a huge volume which only knows the similar pixels. Those pixels are possible candidates for occlusion estimation. From now on, more information must be learned to estimate the optical flow. To do so, after a convolutional layer a pooling layer is added (see Figure 6) and a refinement step can be applied.

Pooling layers downgrades the number of parameters, but the drawback is the down sampling of the image quality. A detailed view[1] of the original code is given. Depends on the padding, but a convolution makes the volume

[1]Link to the original Layer: https://github.com/liruoteng/FlowNet/blob/master/models/flownet/model_corr/train.prototxt

smaller. Pooling takes the maximum value of a two by two kernel window. So less parameters must be learned.

This helps to filter noisy activation [12] by using a single value of a receptive field. A side effect is that it retains stable activations in upper layers [12]. A drawback is the loss of spatial information within the receptive field and is unbecomingly for sharp edges that is required for semantic segmentation [12].

In order to keep the end-to-end structure, a up-sampling procedure must be introduced to obtain the same height and width as of the input images. The refinement specifies some deconvolution layers Figure 8, which upsamples the images. The image size is growing during this process as well. This bilinear (see section 5.3.1) up-sampling approach leads to a simple computational less expensive up sampling method [2].

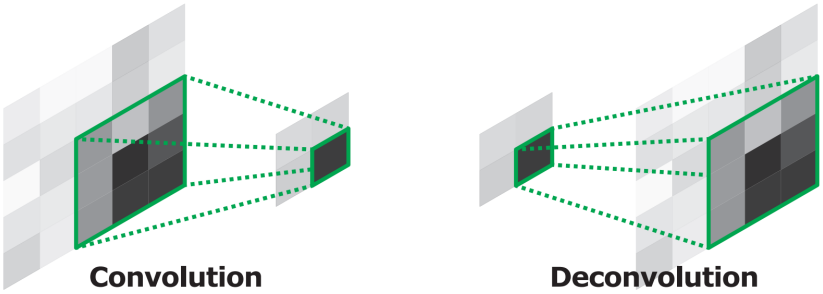

Figure 8: Example of convolution and Deconvolution [12]

3.2 Improved FlowNet: FlowNet2

In this section, we analyze the FlowNet2 in detail. we only want to describe the improved idea of an existing deep neural network architecture and which tricks they used for more competitive results. As we stated, FlowNet2 is an improvement based on FlowNet (see Section 3.1). Ilg et al. [2] state a decrease of the endpoint error (EPE) so that the accuracy is four times higher [2], but it is a little bit slower than the FlowNet [2]. The idea is that to stack several FlowNet architectures together. FlowNet2 performs better in sharper edges from one vector field to another as FlowNet (see Section 2.3).

FlowNet tried to learn small displacements (see Section 2.1.1) within a small area in the pictures. On the contrary, FlowNet2 learns a large displacement by combining several FlowNets which makes the assumption 2-$\mathcal{F}$ redundant. Ilg et al. state that to deal with small displacements, they used a smaller striding parameter [2] in convolutions in the FlowNetS architectures. This means that the kernel window does not shift for example every six pixels and does their convolution, it just shifts three pixels. So more pixels are

convolved and less information is lost. The trade-off is that more calculations must be done and therefore it lasts longer for finishing computations. Brightness Error [2] is estimated:

$$Brightness\ Error = 1st\ Img - Warp(2nd\ Img, Prev.\ Estimated\ Flow) \quad (9)$$

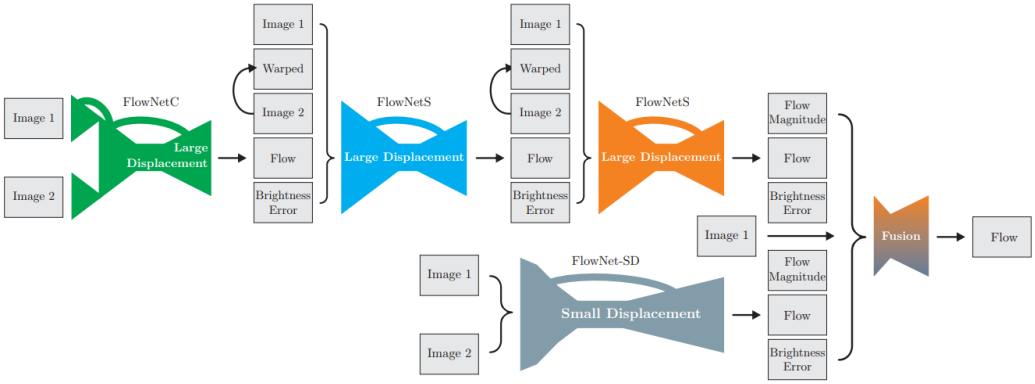

Figure 9: Schematic overview of FlowNet2 [2].

In Figure 9, it can be seen one FlowNetC (FlowNet with correlation architecture, compare Figure 6), two FlowNetS (FlowNet without correlation architecture, see Figure 6) and one FlowNet-SD (FlowNet for learning small displacement).

A problem that comes along by end-to-end networks is the per pixel prediction, it can result in very noisy. The use as refinement neural networks which relies on the formulation of [8] (see Section 2.3), but is not investigated in more detail.

3.2.1 Stacking CNN

To stack neural networks, we need to warp the output of one neural network with the input of another neural network. Warping is meant by Ilg et al. stated some different configurations, whereas warping as an big impact on the results. We take as an example two FlownetS as Net1 and Net2 that should be warped together. Different configuration can be applied such as pretrain the Net1 and random initialized Net2 or pretrain both.

1. Without warping: Increases performance on the dataset Chairs[2] [2], but decreases on Sintel [2]. The dataset Flying Chairs (20.000 images) is huge compared Sintel (1048 images) and that is why Sintel tends to overfit.

[2] a synthetic dataset

2. With warping: Stacking is helpful for better results [2].
3. Loss after Net1: This counteracts against the vanishing gradient problem.
4. Best result on Sintel: Net1 is trained, while Net2 is random initialized and trained with warping [2].
5. Best result on Chairs: It is the last configuration in Table 3. The first network is trained and the second is trained after warping [2].

Stacking networks has overfitting as a drawback. The neural network is melting to one big network. Ilg et al. suggest to train the networks one after another and warping helps from refine the optical flow [2].

Stack architecture	Training enabled		Warping included	Warping gradient enabled	Loss after		EPE on Chairs test	EPE on Sintel train *clean*
	Net1	Net2			Net1	Net2		
Net1	✓	–	–	–	✓	–	3.01	3.79
Net1 + Net2	✗	✓	✗	–	–	✓	2.60	4.29
Net1 + Net2	✓	✓	✗	–	✗	✓	2.55	4.29
Net1 + Net2	✓	✓	✗	–	✓	✓	2.38	3.94
Net1 + W + Net2	✗	✓	✓	–	–	✓	1.94	**2.93**
Net1 + W + Net2	✓	✓	✓	✓	✗	✓	1.96	3.49
Net1 + W + Net2	✓	✓	✓	✓	✓	✓	**1.78**	3.33

Table 3: FlowNet2 - comparison of two stacking FlowNetS [2]. Net1 is trained with the Chairs dataset. Net2 is initialized randomly. Net1 and Net2 trained with Chairs, or only Net2 is trained with Chairs. Training without warping easily overfits the network to the Chairs dataset.

3.3 Unsupervised deep learning approach: DSTFlow

Beside FlowNet2, dense spatial transform (DST) Flow [3], an end-to-end CNN, was published in 2017. In the previous described sections DCNN are supervised approaches. This section will discuss a supervised DCNN approach, that is working quite well. Unsupervised means that there is no labeled data. Application for such CNN is often meant to autonomous driving and video segmentation [13]. Unfortunately, no implementation is online and no performance evaluation could be found (compare Table 1 and Table 2). A DSTFlow network consists of 3 key components:

- Localization layer: FlowNetS provides already the needed functionality to predict a flow fields and is also an end-to-end approach. That is why it is adopted for this network.
- Sampling net: warping the input feature map [3].
- Loss layer: For large displacements, they decided to use the global estimation method of Brox and Malik, which is an refinement of the

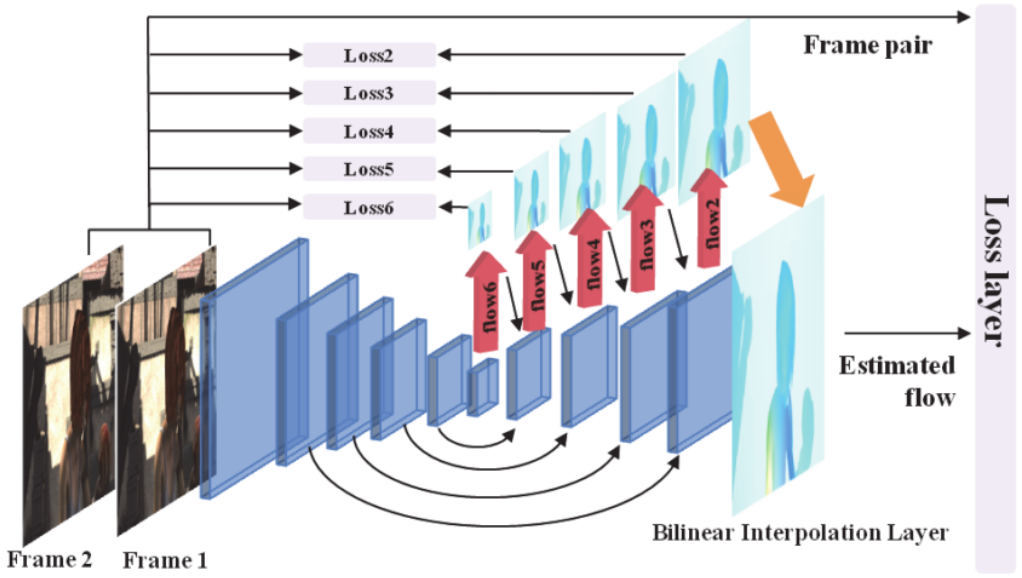

Figure 10: DSTFlow Architecture [14]. The convolutional hidden layers are akin to FlowNetS with convolution and deconvolution.

traditional method of Horn and Schunk.

3.3.1 Localization Network

The localization layer gets as input a pair of consecutive images and outputs the flow fields. The images are stacked together in the z-axis [3]. That is the depth of the input has $3 + 3 = 6$ [3] (compare Figure 10). 3 stands for the depth of each color channel RGB (Red, Green, Blue). After some convolution layers are following some deconvolution layers [3], whose purpose is to unpool/upsampling the quality of the features. Then, the output of the upsampling is concatenated with the corresponding convolutional layer and with the corresponding flow prediction from previous scale [3].

3.3.2 Sampling Network

The sampling network gets as an input the output of the localization network. The sampling kernel (bilinear, see Section 5.3.1) is applied at the input features U. The indices (m,n) indicating the value of U at position (m,n) [3]. While (x_i^s,y_i^s) defines the spatial location. V is the output.

The forward-path can be achieved through [3]:

$$V_i^c = \sum_{n=1}^{H} \sum_{m=1}^{W} U_{nm}^c max(0, 1 - |x_i^s - m|) max(0, 1 - |y_i^s - n|) n \tag{10}$$

The iterate over the input feature map and multiply only the positive values with the bilinear kernel. Negative values are ignored as in the ReLU activation function.

Back-propagation is also described in the paper [3, 15], what is essential that the DCNN learns. The $max()$ function makes the derivation easy as it is known from the ReLU activation function:

$$\frac{\partial V_i^s}{\partial U_{nm}^c} = \sum_{n=1}^{H} \sum_{m=1}^{W} max(0, 1 - |x_i^s - m|) max(0, 1 - |y_i^s - n|) \tag{11}$$

$$\frac{\partial V_i^s}{\partial x_i^s} = \sum_{n=1}^{H} \sum_{m=1}^{W} U_{nm}^c max(0, 1 - |y_i^s - n|) \begin{cases} 0 & if |m - x_i^s| \geq 1 \\ 1 & if m \geq x_i^s \\ -1 & if m < x_i^s. \end{cases} \tag{12}$$

3.3.3 Loss Layer

In [3], a traditional loss function used in learning-free variational methods, e.g. by Brox [16] based on the formulation of Horn and Schunck [1] (see Section 2.2) .

The data loss measures dissimilarity between one image and the sequential warped image with the predicted optical flow field. Smooth term is important for differencing between flow predictions in the neighborhood.

In Figure 11 you can see the blue line, which is produced by the Charbonnier penalty and is close to the l_1-norm. The l_1-norm is the best result and is the most robust convex function [17].

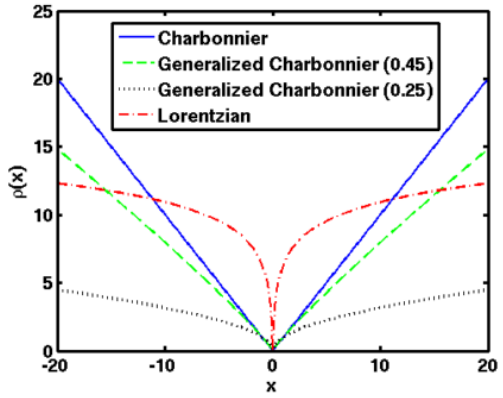

Figure 11: "Penalty functions for the spatial terms: Charbonnier ($\epsilon = 0.001$), generalized Charbonnier ($a = 0.45$ and $a = 0.25$)" [17].

To define the dense spatial transform DST loss, they used the Charbonnier penalty [18].

$$\Psi(s) = \sqrt{(s^2 + \beta^2)} \; with \; \beta = 0.001. \tag{13}$$

DST is composed of two parts. The dense part l_D is basically the Charbonnier penalty over the euclidean distance of the difference of the first image at position x and the second image of a area around x. Added to the Euclidean distance is the same scheme, but this time the gradients of the images describe a change of intensity. Intensity change can again found in Section 2.1.1 at the BCA assumption:

$$l_D = \int_\Omega \Psi(|I_2(x+w) - I_1(x)|^2 + \gamma|\nabla I_2(x+w| - \nabla I_1(x)|^2)dx. \tag{14}$$

The spatial part (often said smooth term) is again the Charbonnier penalty over the euclidean distance of gradient of the image pixels (u,v) and therefore differencing between neighboring flow predictions:

$$l_S = \int_\Omega \Psi(|\nabla u(x)|^2 + |\nabla v(x)|^2)dx. \tag{15}$$

The final loss is the addition of the dense and spatial part.

$$l_{DST} = l_D + \alpha l_S. \tag{16}$$

3.3.4 Multi-scale loss accumulation

This layer is for improving the accuracy of the predicted flow. As in Figure 10 shown, there 6 loss layers, which are used at different scales. This procedure is similar to FlowNet. At this point, the error EPE must be minimized. Hence, the parameters (weights, bias) are updated during back propagation.

3.4 UnFlow

UnFlow another end-to-end network, that is based on the principle of auto-encoding (AE). The DCNN is aimed for learning real world examples (City Shapes, KITTI), what is way more difficult to the syntactical datasets like Chairs or Sintel.
The optical flow is estimated in 3 steps:

1. Design of bidirectional (forward/backward) optical flow estimation [9]
2. Train of FlowNetC with unsupervised loss [9]

3. Iterative refinement: Stacking several CNNs like FlowNet2.0 [2, 9]

In Figure 12 there are two consecutive images put in two AE, which are sharing weights and bias. The goal is to estimate the optical flow of the forward path. The backward path is need for occlusion detection [9]. In the loss function occlusions is taken into account by penalize them with the Charbonnier penalty [9] (see Section 3.3.3). Occluded pixels will be penalized constant [9] to avoid that all pixels are occluded.

Moreover, Meister et al. decided not rely to on the BCA 1-$\mathcal{F}$, because it is not robust due to illumination changes, what is suitable for realistic environments. Better is to use the census transform [9], which "can compensate multiplicative and additive illumination changes" [9].

In Figure 13 a schematic representation of the algorithm can be seen. Two consecutive images are taken as input and the forward as well the backward flow is computed. The backward flow is the inverse of the forward flow. When the mismatch of these 2 flows is very large, then those pixels are marked as occlusion. In the last step, the occlusions are needed in the data loss.

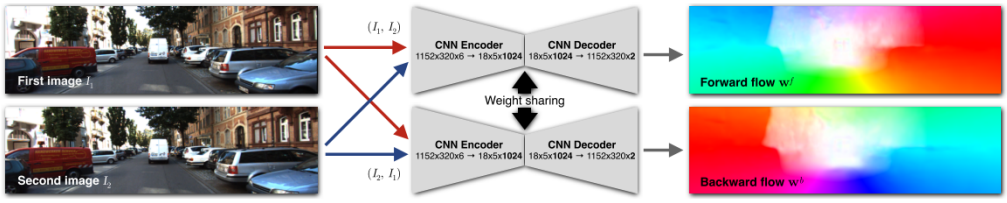

Figure 12: Schematic bidirectional [9, 19] training of FlowNet [2]

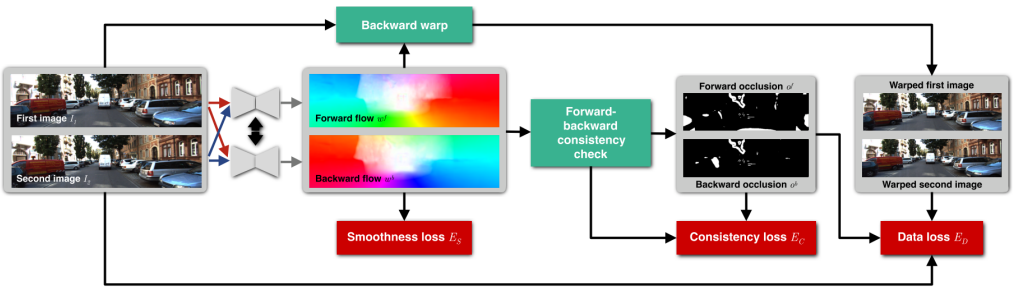

Figure 13: Schematic Architecture of UnFlow [9]

4 Comparison of results

Several DCNNs have been described in the previous sections. The results of each architecture differs, depending on the used dataset (Sintel, Middle-

bury, Chairs, Kitti). However, as we learned from the previous descriptions, FlowNet is often used as basis for other approaches. The results were discussed on the basis of various papers. An own execution of the DCNNs was not possible, due to the lack of available computational power.

4.1 FlowNet and FlowNet2

Method	Sintel Clean		Sintel Final		KITTI		Middlebury train		Middlebury test		Chairs	Time (sec)	
	train	test	train	test	train	test	AEE	AAE	AEE	AAE	test	CPU	GPU
EpicFlow [30]	2.40	4.12	3.70	6.29	3.47	3.8	0.31	3.24	0.39	3.55	2.94	16	-
DeepFlow [35]	3.31	5.38	4.56	7.21	4.58	5.8	0.21	3.04	0.42	4.22	3.53	17	-
EPPM [3]	-	6.49	-	8.38	-	9.2	-	-	0.33	3.36	-	-	0.2
LDOF [6]	4.29	7.56	6.42	9.12	13.73	12.4	0.45	4.97	0.56	4.55	3.47	65	2.5
FlowNetS	4.50	7.42	5.45	8.43	8.26	-	1.09	13.28	-	-	2.71	-	0.08
FlowNetS+v	3.66	6.45	4.76	7.67	6.50	-	0.33	3.87	-	-	2.86	-	1.05
FlowNetS+ft	(3.66)	6.96	(4.44)	7.76	7.52	9.1	0.98	15.20	-	-	3.04	-	0.08
FlowNetS+ft+v	(2.97)	6.16	(4.07)	7.22	6.07	7.6	0.32	3.84	0.47	4.58	3.03	-	1.05
FlowNetC	4.31	7.28	5.87	8.81	9.35	-	1.15	15.64	-	-	2.19	-	0.15
FlowNetC+v	3.57	6.27	5.25	8.01	7.45	-	0.34	3.92	-	-	2.61	-	1.12
FlowNetC+ft	(3.78)	6.85	(5.28)	8.51	8.79	-	0.93	12.33	-	-	2.27	-	0.15
FlowNetC+ft+v	(3.20)	6.08	(4.83)	7.88	7.31	-	0.33	3.81	0.50	4.52	2.67	-	1.12

Table 4: FlowNet results presented by Dosovitskiy et al. [10].

FlowNet is an overall term for basically two different network types, the FlowNetS (simple) and the FlowNetC (correlation), which can be seen in Figure 6.

Table 4 lists the the EPE (see Section 5.3) or AEE for average endpoint error. FlowNetS is better on the test set of the dataset Sintel Clean. The results with variational refinement (+v) and fine tuning (+ft) results the FlowNetC slightly better. On the other hand, the time for computing is increasing. The dataset Chairs is the largest dataset and hence gives the best results by the smallest EPE. This dataset is a syntactic one and Dosovitskiy et al. proved the generalization of the network to test it on realistic scenes and could even beat state-of-the-art methods [10].

In Figure 14 an example of the Sintel dataset is shown. On the left hand side there is the ground truth, how the optical flow should really look like. In the middle is just the tested FlowNetS. The EPEs on the right top corners are corresponding to the Table 4. The FlowNetS does not take any correlation into account, but with the variational refinement, it delivers better results. The FlowNetS+v is a good example illustrating that a mixture of a CNN and a variational method can successfully lead to better results.

In Table 5 the results comparing FlowNet2 to FlowNet and state-of-the-art algorithms are shown. FlowNet2 outperforms FlowNet by its segmentation performance and it can even compete with state-of-the-art algorithms.

	Method	Sintel *clean* AEE		Sintel *final* AEE		KITTI 2012 AEE		KITTI 2015 AEE	KITTI 2015 Fl-all		Middlebury AEE		Runtime ms per frame	
		train	*test*	*train*	*test*	*train*	*test*	*train*	*train*	*test*	*train*	*test*	CPU	GPU
Accurate	EpicFlow† [22]	2.27	4.12	3.56	6.29	**3.09**	**3.8**	9.27	27.18%	**27.10%**	0.31	0.39	42,600	–
	DeepFlow† [32]	2.66	5.38	3.57	7.21	4.48	5.8	10.63	26.52%	29.18%	**0.25**	0.42	51,940	–
	FlowFields [2]	**1.86**	**3.75**	**3.06**	**5.81**	3.33	3.5	**8.33**	**24.43%**	–	0.27	**0.33**	22,810	–
	LDOF (CPU) [7]	4.64	7.56	5.96	9.12	10.94	12.4	18.19	38.11%	–	0.44	0.56	64,900	–
	LDOF (GPU) [27]	4.76	–	6.32	–	10.43	–	18.20	38.05%	–	0.36	–	–	6,270
	PCA-Layers [33]	3.22	5.73	4.52	7.89	5.99	5.2	12.74	27.26%	–	0.66	–	**3,300**	–
Fast	EPPM [4]	–	**6.49**	–	**8.38**	–	9.2	–	–	–	–	**0.33**	–	200
	PCA-Flow [33]	**4.04**	6.83	**5.18**	8.65	**5.48**	**6.2**	**14.01**	**39.59%**	–	**0.70**	–	140	–
	DIS-Fast [16]	5.61	9.35	6.31	10.13	11.01	14.4	21.20	53.73%	–	0.92	–	70	–
	FlowNetS [11]	4.50	6.96‡	5.45	7.52‡	8.26	–	–	–	–	1.09	–	–	**18**
	FlowNetC [11]	4.31	6.85‡	5.87	8.51‡	9.35	–	–	–	–	1.15	–	–	32
FlowNet 2.0	FlowNet2-s	4.55	–	5.21	–	8.89	–	16.42	56.81%	–	1.27	–	–	**7**
	FlowNet2-ss	3.22	–	3.85	–	5.45	–	12.84	41.03%	–	0.68	–	–	14
	FlowNet2-css	2.51	–	3.54	–	4.49	–	11.01	35.19%	–	0.54	–	–	31
	FlowNet2-css-ft-sd	2.50	–	3.50	–	4.71	–	11.18	34.10%	–	0.43	–	–	31
	FlowNet2-CSS	2.10	–	3.23	–	**3.55**	–	**8.94**	29.77%	–	0.44	–	–	69
	FlowNet2-CSS-ft-sd	2.08	–	3.17	–	4.05	–	10.07	30.73%	–	0.38	–	–	69
	FlowNet2	**2.02**	**3.96**	**3.14**	6.02	4.09	–	10.06	30.37%	–	**0.35**	**0.52**	–	123
	FlowNet2-ft-sintel	(1.45)	4.16	(2.01)	**5.74**	3.61	–	9.84	**28.20%**	–	0.35	–	–	123
	FlowNet2-ft-kitti	3.43	–	4.66	–	(1.28)	**1.8**	(2.30)	(8.61%)	**11.48%**	0.56	–	–	123

Table 5: FlowNet2 results presented by Ilg et al. [2] (FlowNetS and FlowNetC are meant to be FlowNetS-ft and FlowNetC+ft). Compared to public benchmarks. AEE is the average endpoint error - similar to EPE. The best result per category is in bold.

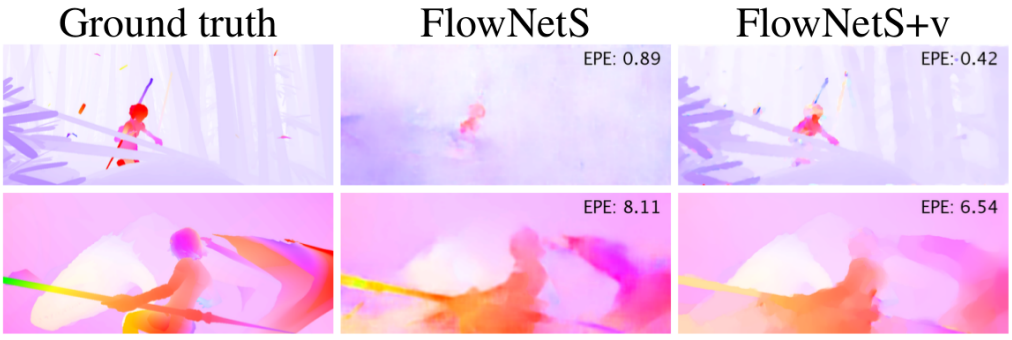

Figure 14: Results presented by Dosovitskiy et al. [10] on the dataset Sintel. Left is the ground truth. In the middle is FlowNetS. Right is the FlowNetS+v (FlowNet simple with variational refinement) that results better in in details.

Ilg et al. also mentioned that FlowNetC outperforms FlowNetS, because the of modified dataset and training schedules [2]. On the Sintel dataset, FlowNet2 is better than DeepFlow and EpicFlow. On Kitti it is slightly worse than EpicFlow and FlowField. The property of KITTI is that it contains very large displacements. They also stated that they have the best EPE on KITTI2012 [2] after some fine-tuning, which is not clearly described. On Middlebury, the training error is quite good, while the test error show signs of overfitting.

In FlowNet2, results are comparable to the results of FlowFields and the edges are quite sharp, whereas FlowNetS does not show quite good results on this dataset. On the other hand, FlowNetS is the fastest of all listed approaches with 18ms.

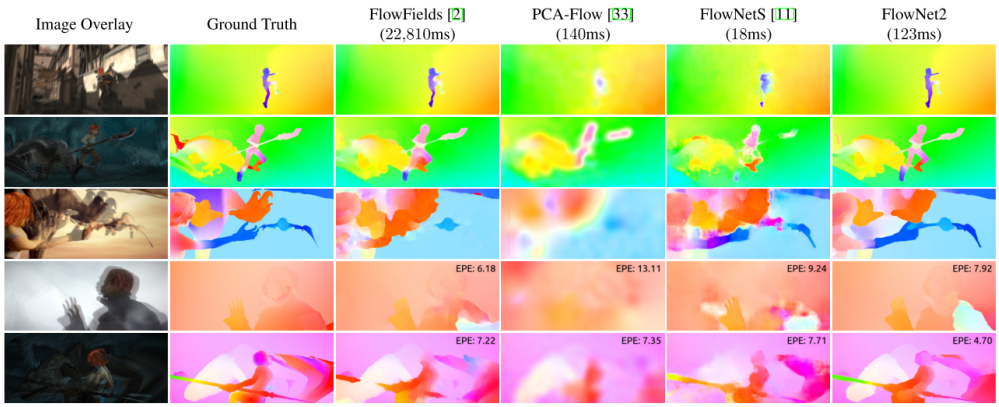

Figure 15: Results [2] of FlowNet2, which results close to FlowFields (best approach). FlowNet2 is very slow compared to FlowNetS and PCA-Flow.

4.2 DSTFlow with FlowNet and FlowNet2

Method	Chairs	KITTI2012				KITTI2015				Sintel Clean				Sintel Final				Time (Sec)	
	testing	train		test		train			test	train		test		train		test		CPU	GPU
		occ	noc	occ	noc	occ	noc	fl-all	fl-all	occ	noc	occ	noc	occ	noc	occ	noc		
Epicflow	2.94	3.47	1.48	3.8	1.5	9.57	4.45	0.28	0.27	2.40	1.23	4.12	1.36	3.70	2.63	6.29	3.06	16	—
Deepflow	3.53	4.58	2.22	5.8	1.5	13.89	6.75	0.31	0.30	3.31	1.78	5.38	1.77	4.56	3.07	7.21	3.34	17	—
EPPM	—	—	—	9.2	2.5	—	—	—	—	—	—	6.49	2.68	—	—	8.38	4.29	—	0.2
LDOF	3.47	13.73	4.62	12.4	5.6	18.23	9.05	0.37	—	4.29	2.83	7.56	3.43	6.42	4.41	9.12	5.04	65	2.5
FlowNetS(C+S)	3.04	7.52	4.25	9.1	5.0	14.19	8.12	0.51	—	*3.66	*2.82	6.96	—	*4.44	*3.99	7.76	—	—	0.08
DSTFlow(Chairs)	5.11	16.98	9.21	—	—	24.30	14.23	0.52	—	6.93	5.05	10.40	5.20	7.82	5.97	11.11	5.92	—	0.08
DSTFlow(KITTI)	6.86	10.43	3.29	12.4	4.0	16.79	6.96	0.36	0.39	7.10	5.26	10.95	5.87	7.95	6.16	11.80	6.70	—	0.08
DSTFlow(Sintel)	5.68	15.78	8.24	—	—	23.69	13.88	0.55	—	*6.16	*4.17	10.41	5.30	*7.38	*5.45	11.28	6.16	—	0.08
DSTFlow(C+K)	5.86	16.17	8.32	—	—	22.93	12.81	0.48	—	7.51	5.74	—	—	8.29	6.55	—	—	—	0.08
DSTFlow(C+S)	5.52	17.17	9.52	—	—	25.98	15.89	0.53	—	*6.47	*4.61	10.84	5.62	*6.81	*4.91	11.27	6.02	—	0.08

Table 6: DSTFLow results with EPE on OCC and NOC pixels. Asterisks mean that they were trained on the own data (overfitting). [3]

DSTFlow is the first unsupervised DCNN approach for learning optical flow. The performance lags behind those of state-of-the art algorithms.

In Table 6 several training combinations are shown, i.e. DSTFlow(C+K) [3] means DSTFlow trained with the dataset Chairs and refined with the dataset KITTI or DSTFLOW(C+S) [3] means that it is trained with Chairs and refined with Sintel. FlowNetS(C+S) performs better on all datasets than all variations of DSTFlow, but both yield the same time on a GPU. On dataset Sintel, although the data is augmented, the test error gives bad results, while the training error yields good results. Comparing DSTFlow to

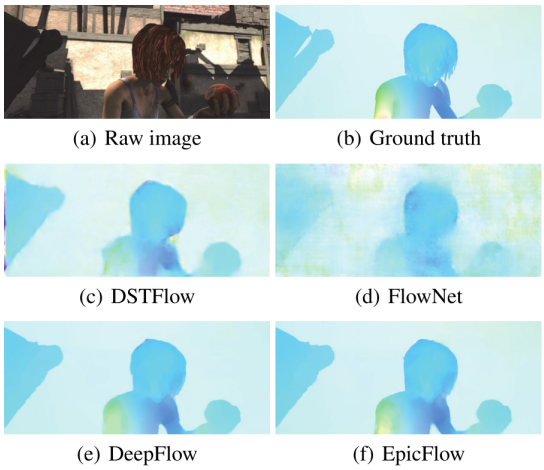

Figure 16: DSTFlow results on dataset Sintel. [3]

FlowNet Figure 10, the edges are not that blurry and the silhouette can be better recognized. The state-of-the-art algorithms like DeepFlow or EpicFlow cannot be outperformed.

4.3 Unflow with FlowNet and DSTFlow

In Table 7, several benchmarks are listed on datasets. On KITTI2012, the UnFlow-CSS-ft (trained with dataset city shapes with fine-tuning) performs best on the test set in non-occluded areas and is even better than DST-Flow. Even Unflow-C-Cityscapes, which was not trained on the dataset KITTI15 (see Figure 17, outperforms DSTFlow [9]. On Middlebury, Unflow C Cityshapes outperforms FlowNetS. This is an evidence that the network can generalize the learned data. In contrast to that, on Sintel, the Unflow CNN cannot outperform FlowNetS+ft nor FlowNet2, but it is better than DSTflow.

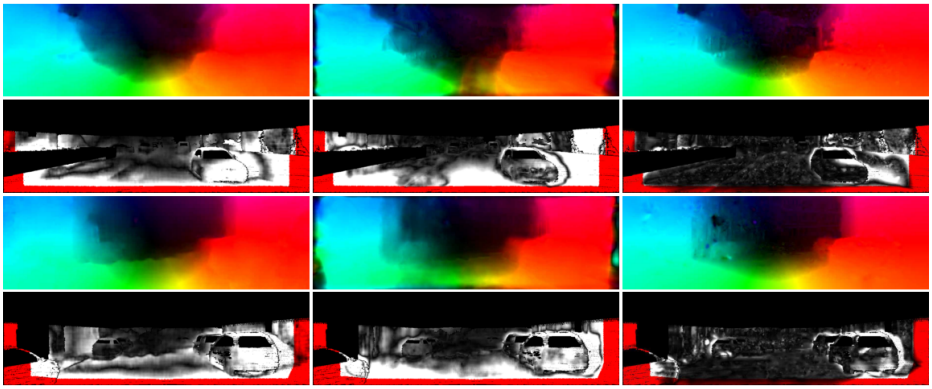

Figure 17: UnFlow Results [9]. From left to right: FlowNetS, UnsupFlowNet [20] and Unflow-CSS. Above flow, underneath flow error. KITTI2012 maps between 0 (black) and white if ≥ 5 pixels.

Method	KITTI 2012				KITTI 2015			Middlebury		Sintel Final	
	AEE (All)		AEE (NOC)		AEE (All)	Fl-all		AEE		AEE	
	train	*test*	*train*	*test*	*train*	*train*	*test*	*train*	*test*	*train*	*test*
DDF (Güney and Geiger 2016)	–	3.4	–	1.4	–	–	21.17%	–	–	–	5.73
PatchBatch (Gadot and Wolf 2016)	–	3.3	–	1.3	–	–	21.07%	–	–	–	5.36
FlowFieldCNN (Bailer, Varanasi, and Stricker 2017)	–	3.0	–	1.2	–	–	18.68%	–	–	–	–
ImpPB+SPCI (Schuster, Wolf, and Gadot 2017)	–	2.9	–	1.1	–	–	17.78%	–	–	–	–
SDF (Bai et al. 2016)	–	2.3	–	1.0	–	–	11.01%	–	–	–	–
FlowNetS+ft (Dosovitskiy et al. 2015)	7.5	9.1	5.3	5.0	–	–	–	0.98	–	(4.44)	7.76
UnsupFlownet (Yu, Harley, and Derpanis 2016)	11.3	9.9	4.3	4.6	–	–	–	–	–	–	–
DSTFlow(KITTI) (Ren et al. 2017)	10.43	12.4	3.29	4.0	16.79	36 %	39 %	–	–	7.95	11.80
FlowNet2-C (Ilg et al. 2017)	–	–	–	–	11.36	–	–	–	–	–	–
FlowNet2-CSS (Ilg et al. 2017)	3.55	–	–	–	8.94	29.77%†	–	0.44	–	3.23	–
FlowNet2-ft-kitti (Ilg et al. 2017)	(1.28)	1.8	–	1.0	(2.30)	(8.61%)†	10.41%	0.56	–	4.66	–
UnFlow-C-Cityscapes (ours)	5.08	–	2.12	–	10.78	33.89%	–	0.85	–	8.23	–
UnFlow-C (ours)	3.78	–	1.58	–	8.80	28.94%	–	0.88	–	8.64	–
UnFlow-CS (ours)	3.30	–	1.26	–	8.14	23.54%	–	0.65	–	7.92	–
UnFlow-CSS (ours)	3.29	–	1.26	–	8.10	23.27%	–	0.65	–	7.91	10.22
UnFlow-CS-ft (ours)	(1.32)	1.9	(0.75)	0.9	(2.25)	(9.24%)	12.55%	0.64	–	11.99	–
UnFlow-CSS-ft (ours)	(1.14)	1.7	(0.66)	0.9	(1.86)	(7.40%)	11.11%	0.64	0.76	13.65	–

Table 7: UnFlow Results [9]. The accuracy is compared on public benchmarks. AEE is the average endpoint error that is similar to EPE. Parentheses indcate that the network is trained on the same dataset as it is tested on. The cross are results of Ilg et al.

5 Appendix

5.1 Available Code

The source code can be very valuable due to sparse description in conference papers in order to understand better the meaning of some explanations.

1. FlowNet: official Caffe implementation - Video
   github.com/liruoteng/FlowNet
2. FlowNet2: official Caffe/Docker implementation - Video
   github.com/lmb-freiburg/flownet2
3. DSTFlow: The source code to "Unsupervised Deep Learning for Optical Flow Estimation" is announced to be published at the beginning of 2018.
4. Unflow: Offcial Tensorflow implementation
   github.com/simonmeister/UnFlow

5.2 Flow Field Graph

The flow direction, is basically given by a vector. For humans it is better to encode the direction and magnitude of such vectors in colors (see Figure 18).

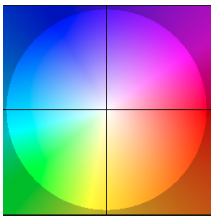

Figure 18: Color coding of optical flow images. No optical flow at all is indicated white and is in the middle of the circle. If the flow vector does look to the right, then the color is red. Cyan means to the right.

5.3 Endpoint Error (EPE)

The EPE corresponds to the euclidean distance of each pixel compared to the ground truth:

$$L_{epe} = \frac{1}{N} \sum \sqrt{(U - U')^2 + (V - V')^2}, \tag{17}$$

where N is the total number of pixels of the predicted optical flow image and function as the normalization. (U, V) are the flow estimated prediction and (U', V') is the ground truth.

5.3.1 Bilinear up-sampling

Assume you have an image of 3×3 pixels and we want to up-sample it to 9×9 pixels. Then, a filter (bilinear, linear, nearest-neighbor, bicubic, sinc and so on) can be applied to the 3×3 image. This filter interpolates the values between the original 3x3 pixels to fill the gaps in-between.

The filter can be seen as a simple mathematical function and apply this function for every x:

$$f(x) = \begin{cases} 1 - |x|, & \text{if } |x| < 1 \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

In Figure 19, an example from avisynth[3] is shown, if the values of x are taken in the interval of {-1, 0, 1}.

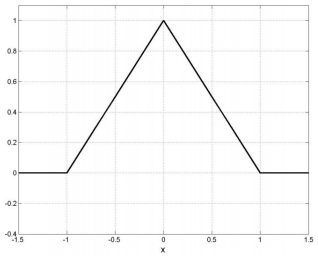

Figure 19: Bilinear interpolation for $x = \{-1, 0, 1\}$. The resampling kernel 18 is applied on x.

[3] http://avisynth.nl/index.php/Resampling

Bibliography

[1] H. Berthold and S. Brian, "Determining optical flow," *Artif. Intell.*, vol. 17, pp. 185–203, Aug. 1981. 1, 4, 5, 8, 14

[2] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 7, 8, 9, 10, 11, 12, 16, 18, 19

[3] Z. Ren, J. Yan, B. Ni, B. Liu, X. Yang, and H. Zha, "Unsupervised Deep Learning for Optical Flow Estimation," *Proc. AAAI Conference on Artif. Intell.*, 2017. 1, 12, 13, 14, 19, 20

[4] S. Baker, D. Scharstein, P. Lewis, S. Roth, M. Black, and R. Szeliski, "A Database and Evaluation Methodology for Optical Flow," *International Journal of Computer Vision*, vol. 92, no. 1, pp. 1–31, 2011. 1, 2, 3

[5] D. Butler, J. Wulff, G. Stanley, and M. Black, "A naturalistic open source movie for optical flow evaluation," *European Conference on Computer Vision (ECCV)*, pp. 611–625, Oct. 2012. 2

[6] F. Stein, "Efficient Computation of Optical Flow Using the Census Transform," *Pattern Recognition*, vol. 3175, pp. 79–86, 2004. 3

[7] T. Brox and J. Malik, "Large Displacement Optical Flow Descriptor Matching in Variational Motion Estimation," vol. 33, no. 3, pp. 500–513, 2011. 5

[8] Y. Chen and T. Pock, "Trainable Nonlinear Reaction Diffusion: A Flexible Framework for Fast and Effective Image Restoration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1256–1272, 2017. 6, 11

[9] "UnFlow: Unsupervised Learning of Optical Flow with a Bidirectional Census Loss," *CoRR*, vol. abs/1711.07837, 2017. 6, 15, 16, 20, 21

[10] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "FlowNet: Learning Optical Flow with Convolutional Networks," *IEEE International Conference on Computer Vision*, pp. 2758–2766, 2015. 8, 9, 17, 18

[11] J. Zbontar and Y. Lecun, "Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches," *Journal of Machine Learning Research*, vol. 17, pp. 1–32, 2016. 8

[12] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," *Proc. of the 2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1520–1528, 2015. 10

[13] M. Menze and A. Geiger, "Object Scene Flow for Autonomous Vehicles," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3061–3070, June 2015. 12

[14] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," *International Conference on Learning Representations 2016*, vol. abs/1511.06434, pp. 1–16, 2015. 13

[15] S. K. Sønderby, C. K. Sønderby, L. Maaløe, and O. Winther, "Recurrent Spatial Transformer Networks," *arXiv*, pp. 1–15, 2015. 14

[16] T. Brox and J. Malik, "Large Displacement Optical Flow: Descriptor Matching in Variational Motion Estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 500–513, 3 2011. 14

[17] D. Sun, S. Roth, and M. Black, "Secrets of Optical Flow Estimation and Their Principles Optical flow : motion of image pixels," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2432–2439, June 2010. 14

[18] A. Bruhn and J. Weickert, "Towards ultimate motion estimation: Combining highest accuracy with real-time performance," *Proc. IEEE International Conference on Computer Vision*, vol. I, pp. 749–755, 2005. 15

[19] J. Hur and S. Roth, "MirrorFlow: Exploiting Symmetries in Joint Optical Flow and Occlusion Estimation," *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 312–321, 2017. 16

[20] J. Yu, A. Harley, and K. Derpanis, "Back to Basics : Unsupervised Learning of Optical Flow via Brightness Constancy and Motion Smoothness," 21