Deep Learning Architectures for Estimating Optical Flow

Paper for the 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, January 26, 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 obtained from different applications, such as autonomous driving vehicles or robots. Today's deep learning approaches still lack behind classical variational methods. Computational expensiveness and no sharp edges between flow fields are the pitfalls 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*

Contents

1 Introduction

This work presents state-of-the-art deep learning architectures for optical flow. After introducing the topic by emphasizing the importance of optical flow, the assumptions of optical flow are explained, which are valid in classical as well as in deep convolutional neural networks (DCNN) approaches. The DCNN are can be divided into supervised and unsupervised approaches.

1.1 Background and Motivation

The research topic "optical flow" was initialized by a paper from Horn and Schunk [3] and is the fundamental work of many ongoing research topics. Almost every classical computer vision method was replaced by deep 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 [10] approach has shown that optical flow can be estimated, but variational methods still yield better results. This year, another deep learning approach [16] was published; it lags slightly behind state-of-the-art performance algorithms.

1.2 Ranking in MiddleBury and Sintel Dataset

The following table shows a ranked list of different optical flow algorithms:

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 [2]

The Middlebury dataset is small and both FlowNets yield bad results. Results of the other DCNNs have not been published by [2].

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 [11]

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 section 1.2.

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 [2]
- Computer Graphics: video synthesis such as faces, fluid reconstruction
- HCI: hand gestures, facial expressions, pose estimation

2 What is the Optical Flow?

Optical flow is generally the representation of movement in a scene. A vector field shows the apparent direction of the movement.

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 not solvable with 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, e.g.: constant pixel brightness and only small movements happened from one frame to another.

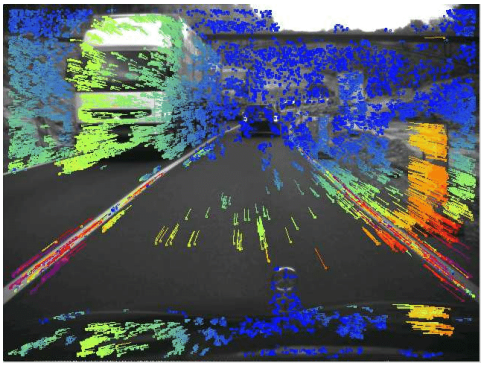

Figure 1: Optical flow vectors with oncoming traffic [18]

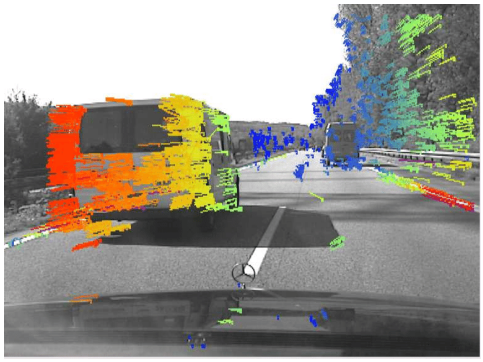

Figure 2: Optical flow vectors by following a car [18]

Best illustrating example is driving a car, where the flow vectors show the direction of pixels are moving. As in fig. 1 and fig. 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 intensity, the more the vector shows in a direction.

2.1 Assumptions and Constraints holding for Optical Flow

Three assumptions/constraints are necessary to describe the optical flow in terms of an algorithm. This 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:* $I(x(t), t) = Constant$, *where* $x = \binom{x_1}{x_2}$ *is the pixel position and the displacement vector* $\triangle x = \binom{\triangle x_1}{\triangle x_2}$.

Assumption 2-$\mathcal{F}$. *Assuming* $\triangle x$ *is small (less than a pixel, or smooth)* $I(x, t) \rightarrow I(x + \triangle x, t + \triangle t)$, *so we can make 1st order Taylor expansion and yields to the linearized optical brightness constancy assumption.*

t or t+1 the derivative yields the same.

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

2.1.2 Optical Flow Constrained (OFC)

Where u is the velocity vector $\binom{u_1}{u_2}$. In practice, we assume $\triangle_t = 1$ s.t. the velocity is equal to the displacement ($u = \triangle_x$) and we assume 2 images of a sequence are given $I_1,\ I_2$. 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

$$I_w(x) = I_2(x + u_0)$$

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

$$(\nabla_w(x))^T(u - u_0) + I_t(x) = 0$$

2.1.3 Spatial Coherence Assumption

For this assumption is it still difficult to find a good model.

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 [3] introduced a global method to estimate the optical flow. It is a fundamental work for many following methods, e.g. [5] 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:

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

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:

$$\min_{U,V} \; ||DU||^2 + ||DV||^2 + \lambda ||I_w^x (U - U_0) + I_w^y (V - V_0) + I_t||^2 \tag{3}$$

Optimality conditions for U and V:

$$\begin{aligned} D^T DU + \lambda I_w^x I_w^x (U - U_0) + I_w^y (V - V_0) + I_t = 0 \\ D^T DV + \lambda I_w^y I_w^x (U - U_0) + I_w^y (V - V_0) + I_t = 0 \end{aligned} \tag{4}$$

The optimality conditions can be rewritten into a matrix notation.

$$\begin{pmatrix} D^T D + \lambda (I_w^x)^2 & \lambda I_w^x I_w^y \\ \lambda I_w^x I_w^y & D^T D + \lambda (I_w^y)^2 \end{pmatrix} \begin{pmatrix} U \\ V \end{pmatrix} = \lambda \begin{pmatrix} I_w^x (I_w^x U_0 + I_w^y) V_0 - I_t \\ I_w^y (I_w^x U_0 + I_w^y) V_0 - I_t \end{pmatrix} \tag{5}$$

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 e.g. 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 fig. 3. Classical Computer Vision approaches as listed in the tables section 1.2 and table 2. Another ongoing challenge is the drawback of end-to-end architec-

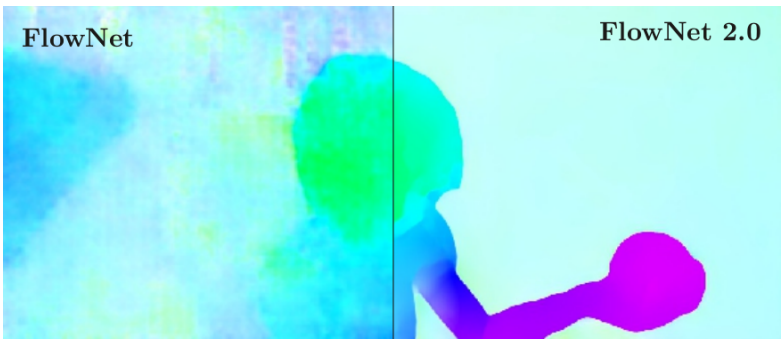

Figure 3: Comparison of FlowNet with FlowNet2 [10]

tures, which suffers often from noise over smooth areas. There are some postprocessing or refinement methods as recently introduced by Chen and Pock [7]. They formulated a reaction diffusion model and can apply it to image image denoising tasks.

3 Deep Learning Architectures for Predicting Optical Flow

In this chapter, some deep learning approaches are chosen for predicting optical flow and are described layer by layer. One is a supervised learning approach FlowNet and 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 a supervised approach and does not include any labels to get feedback by a ground truth. Unflow [12] was recently published and is also treated shortly.

3.1 Supervised Deep Learning Approach: FlowNet and FlowNet2

FlowNet [8] was the first approach, which proved problem "optical flow" as learning problem, which is not competitive to variational approaches. FowNet2 [10] is the improved version and showed up to 50 percent increase of estimating optical flow and is competitive variational approaches, those have been dominating optical flow estimation since the publication of Horn and Schunck [3].

Dosovitskiy et. al [8] used the idea of Zuntar and LeCun [21] 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 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 chapter.

3.1.1 Correlation Layer

A correlation layer is added to improve the matching process, where each patch from f_1 is compared with each patch from f_2.

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

The equation in fig. 5 restrict 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 $D = 2d + 1$ [8]. Moreover, it describes (fig. 5) 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 those feature vectors giving a higher

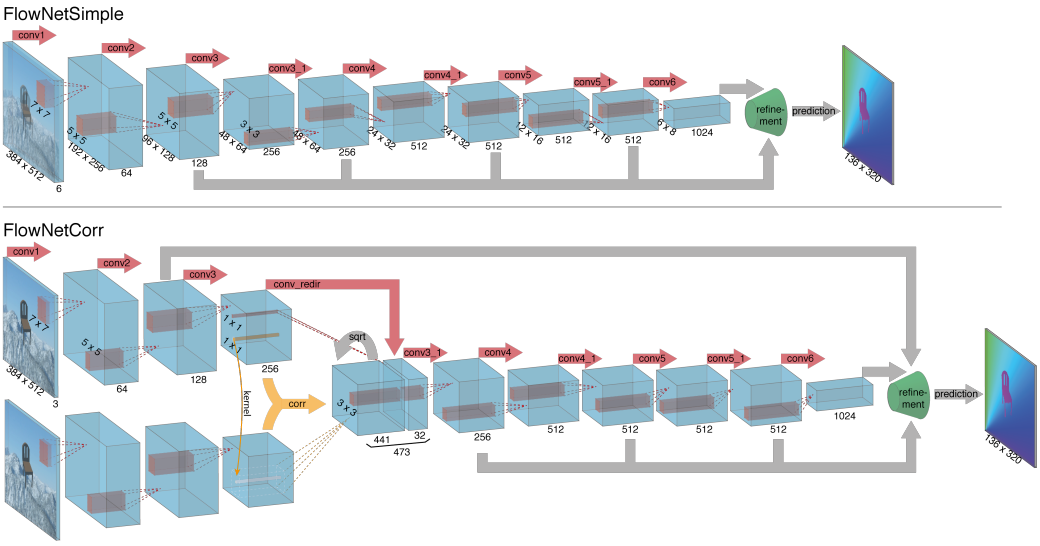

Figure 4: The network architectures: FlowNetSimple [8] (above) and FlowNetCorr [8] (underneath)

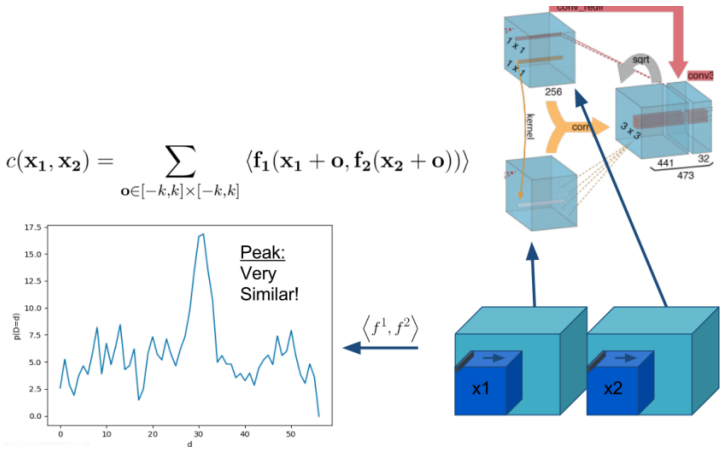

Figure 5: Correlation Layer [10] (right top) and comparison of the features

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 are trying to not comparing all patches $w^2 \cdot h^2$ [8] and is computational expensive [8]. Therefore, a maximal displacement d [8] is introduced, which compares pixel in maximum distance of d in the neighborhood (10 pixels [8]). Then, the authors concatenate feature map as one volume.

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 fig. 4) and afterwards refinement can be applied.

Pooling layers is a method to counteract the computational expensiveness, 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 smaller. Pooling takes the maximum value of a two by two kernel window. So less parameters must be learned.

This helps for filter noisy activation [14] by using a single value of a receptive field. A side effect is that it retains stable activations in upper layers [14]. A drawback is the loss of spatial information within the receptive field and is unbecomingly for sharp edges that is required for semantic segmentation [14].

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, which do exactly the inverse of convolution layers. Instead of the volume is getting smaller, it will be bigger. This bilinear (see section 5.3.1) up-sampling approach leads to a simple computational less expensive up sampling method [10].

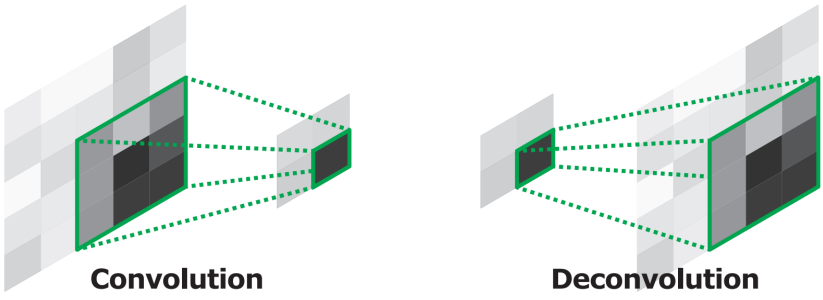

Figure 6: Example of convolution and Deconvolution [14]

Refinement [8] differs slightly to the official Github repository, because the second box not only consists of feature maps from deconv5 and con5_1, but also flow6 which is generated by the following:

$conv6(conv) \rightarrow conv6\_1(conv) \rightarrow predict\_flow6(conv) \rightarrow flow6$

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

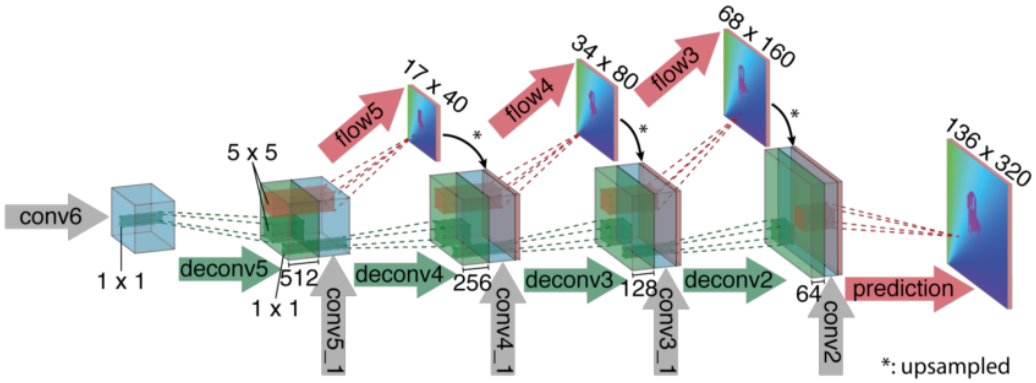

Figure 7: Refinement [8] differs slightly to the official Github repository.

3.2 Improved FlowNet: FlowNet2

In this chapter, I am not analyzing the FlowNet2 in detail. I only want to describe the improved idea of an existing deep neural network architecture and which tricks they used for more competitive results. As I stated, FlowNet2 is an improvement based on FlowNet (see section 3.1). Ilg et al [10] state a decrease of the EPE error so that the accuracy is four times higher [10], but it is a little bit slower than the FlowNet [10]. 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 wrote that to deal with small displacements, they used a smaller striding parameter [10] 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 [10] is estimated:

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

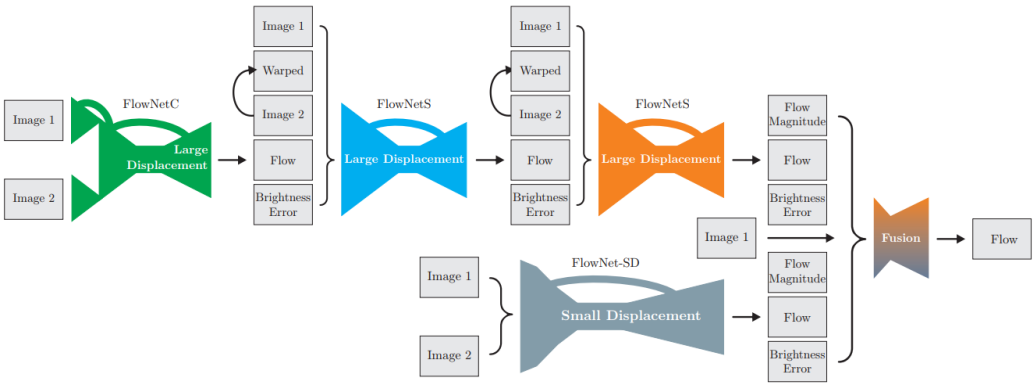

Figure 8: FlowNet2 schematic overview [10]

In fig. 8, it can be seen one FlowNetC (FlowNet with correlation architecture, compare fig. 4), two FlowNetS (FlowNet without correlation architecture, see fig. 4) and one FlowNet-SD (FlowNet for learning small displacement).

Problem by End-To-End Networks which predicts per pixel, it can result in very noisy. The use as refinement neural networks which relies on the formulation of Chen and Pock (see section 2.3).

3.2.1 Stacking Convolutional Neural Networks

Typically, classical approaches are iteratively to improve their performance. In this approach the authors tried to stack different networks together. Ilg et al stated some different configurations, whereas warping as an big impact on the results:

1. Without Warping: Increases performance on the dataset Chairs [10], but decreases on Sintel [10]. The dataset Flying Chairs (20.000 images) is huge compared Sintel (1048 images) and that is why Sintel tends to overfit.
2. With Warping: Stacking is helpful for better results [10].
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 [10].
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 [10].

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 [10]

Stacking networks has overfitting as drawback. The neural network is melting to one big network. Ilg et al suggest to train the networks one after another and warlpiung helps from refine the optcial flow [10].

3.3 Unsupervised deep learning approach: DSTFlow

Beside FlowNet2, Dense Spatial Transform (DST) Flow [16], an end-to-end CNN, was published in 2017. In the previous described chapters DCNN are supervised approaches. This chapter will discuss an supervised approach,

that is an quite well working. Unsupervised means that there is no hidden structure from "unlabeled" input 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 section 1.2 and table 2).

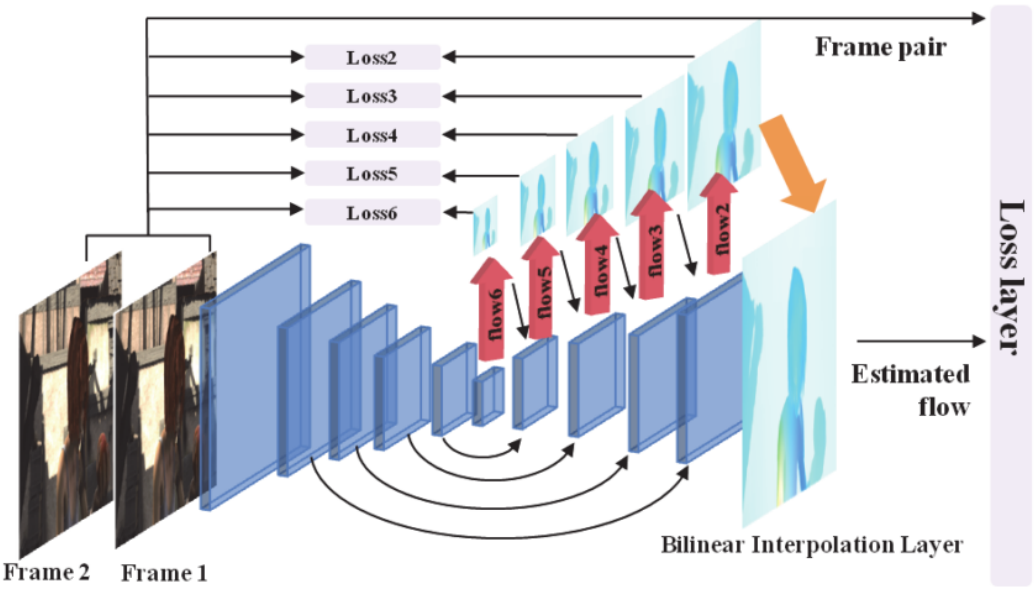

Figure 9: DSTFlow Architecture [15]

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 [16].
- Loss Layer: For large displacements, they decided to use the global estimation method of Brox and Malik, which is an refinement of the traditional method of Horn and Schunk.

3.3.1 Localization Network

The localization layer is the adopted FlowNetS architecture, which does basically the same as the original. It gets as input a pair of consecutive images and outputs the flow fields. The images are stacked together in the z-axis [16]. That means that the depth of the input has $3 + 3 = 6$ [16] (compare fig. 9). 3 stands for the depth of each color channel RGB (Red, Green, Blue). After some convolution layers are following some deconvolution layers [16], 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 [16].

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). While (x_i^s,y_i^s) defines the spatial location. V is the output.

The forward-path can be achieved through [16]:

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

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 [16] [17], what is essential that the DCNN learns. The $max()$ function makes the derivation easy as it is known from the ReLU activation function.

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

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

3.3.3 Loss Layer

In the paper of Z. Ren et al [16], they stated to use traditional loss function used in learning-free variational methods, e.g. by Brox [4] based on the formulation of Horn and Schunck [3] (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.

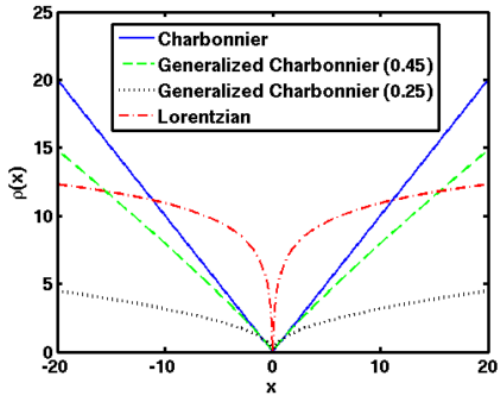

Figure 10: "Penalty functions for the spatial terms: Charbonnier ($\epsilon = 0.001$), generalized Charbonnier ($a = 0.45$ and $a = 0.25$)." [19]

In fig. 10 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 [19].

To define the dense spatial transform (DST) loss, they used the Charbonnier penalty [6].

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

DST is composed of two parts. The dense part 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.

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

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.

$$l_S = \int_\Omega \Psi(|\nabla u(x)|^2 + |\nabla v(x)|^2)dx$$

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

$$l_{DST} = l_D + \alpha l_S$$

3.3.4 Multi-scale loss accumulation

This layer is for improving the accuracy of the predicted flow. As in fig. 9 shown, there 6 loss layers, which are used at different scales. This procedure is similar to FlowNet (see fig. 7). At this point, the error (EPE) must be minimized. Hence, the parameters (weights, bias) are updated during back propagation.

3.4 Latest Release: UnFlow

During I have been writing this work, another unsupervised approach is published by Meister and Roth. It is another end-to-end network, that works with the principle of auto-encoding (AE). Their 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 estimation works in 3 steps:

1. Design of bidirectional (forward/backward) optical flow estimation [12]
2. Train of FlowNetC with unsupervised loss [12]
3. Iterative refinment: Stacking several CNNs like FlowNet2.0 [10] [12]

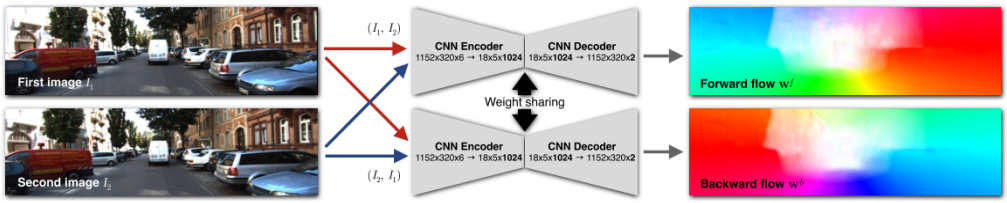

Figure 11: Schematic bidirectional [9] [12] training of FlowNet [10]

In fig. 11 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 [12]. In the loss function occlusions is taken into account by penalize them with the Charbonnier penalty [12] (see section 3.3.3). Occluded pixels will be penalized constant [12] to avoid that all pixels are occluded.

Moreover, Meister et al decided do not rely 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 [12], which "can compensate multiplicative and additive illumination changes" [12].

In fig. 12 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

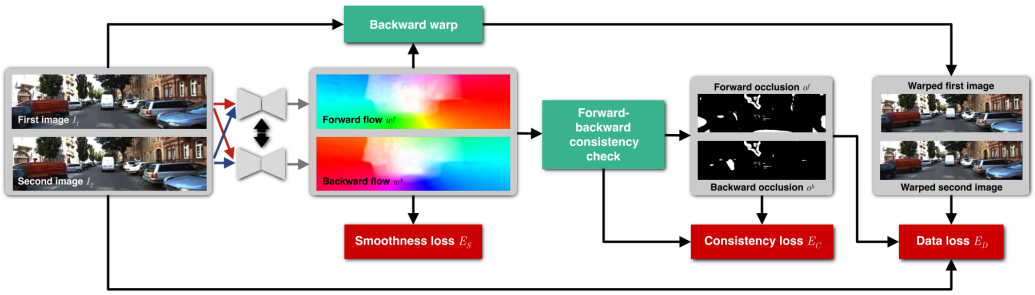

Figure 12: Schematic Architecture of UnFlow [12]

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.

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, Middlebury, 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

FlowNet is an overall term for basically two different network types, the FlowNetS (simple) and the FlowNetC (correlation), which can be seen in fig. 4.

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 [8]

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 [8].

In fig. 13 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.

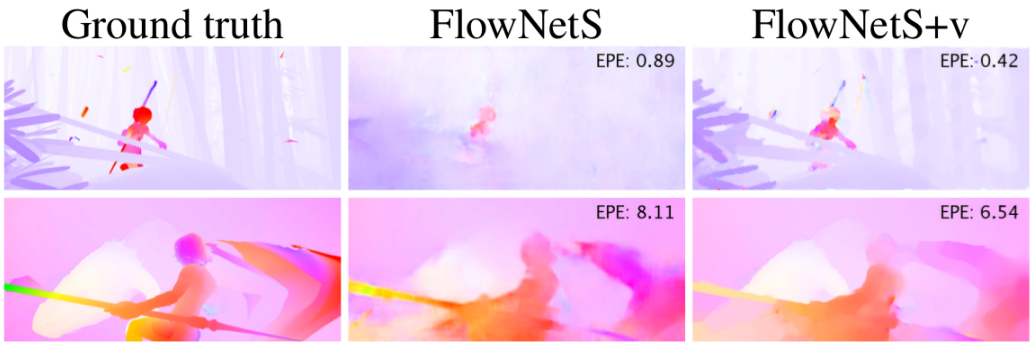

Figure 13: Results presented by Dosovitskiy et al [8] on the dataset Sintel

	Method	Sintel *clean* AEE		Sintel *final* AEE		KITTI 2012 AEE		KITTI 2015 AEE	KITTI 2015 Fl-all	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 [10] (FlowNetS and FlowNetC are meant to be FlowNetS-ft and FlowNetC+ft)

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. Ilg et al also mentioned that FlowNetC outperforms FlowNetS, because the of modified dataset and training schedules [10]. 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 [10] after some fine-tuning, which is not clearly described. On Middlebury, the training error is quite good, while the test error 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.

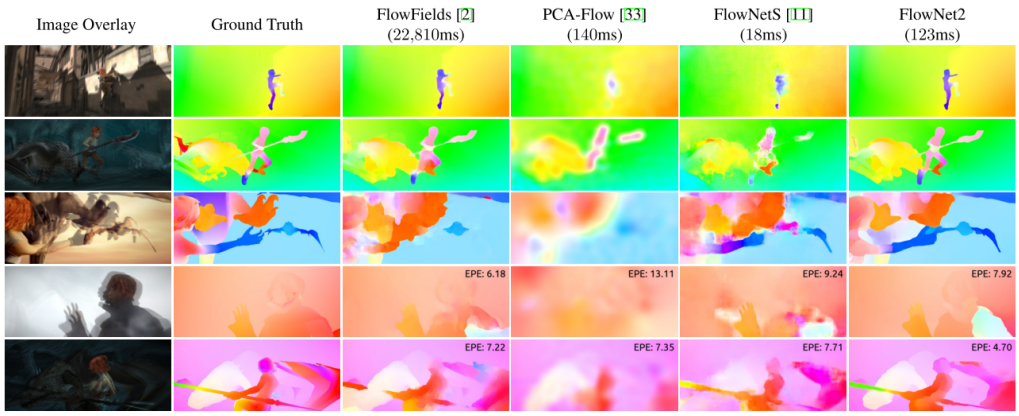

Figure 14: Results [10]

4.2 DSTFlow with FlowNet and FlowNet2

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

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 occluded (OCC) and non-occluded (NOC) pixels. Asterisks mean that they were trained on the own data (overfitting). [16]

In table 6 several training combinations are shown, i.e. DSTFlow(C+K) [16] means DSTFlow trained with the dataset Chairs and refined with the dataset KITTI or DSTFLOW(C+S) [16] 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 FlowNet fig. 9, 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.

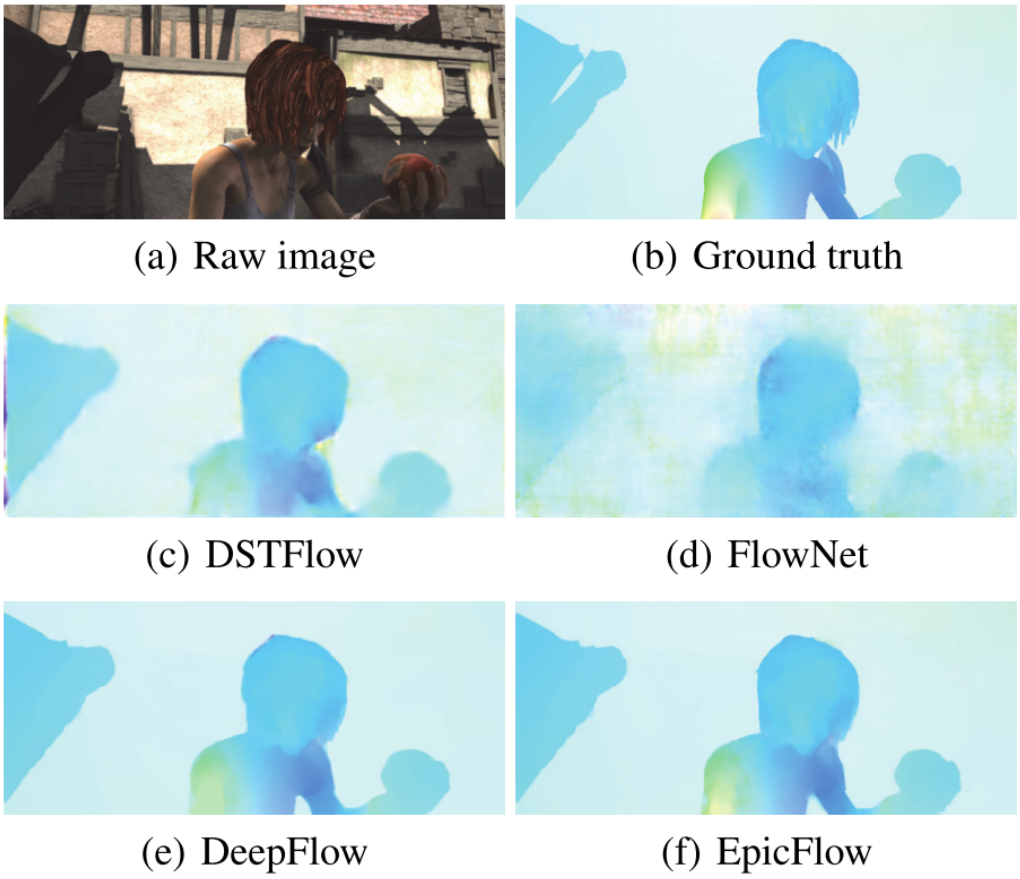

Figure 15: DSTFlow results on dataset Sintel. [16]

4.3 Unflow with FlowNet and DSTFlow

Unflow is another unsupervised approach and is the latest publication. The aim of this approach is to predict well on real data.

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 [12].

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 DSTFlow. Even Unflow-C-Cityscapes, which was not trained on the dataset KITTI15 (see fig. 16, outperforms DSTFlow [12]. 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.

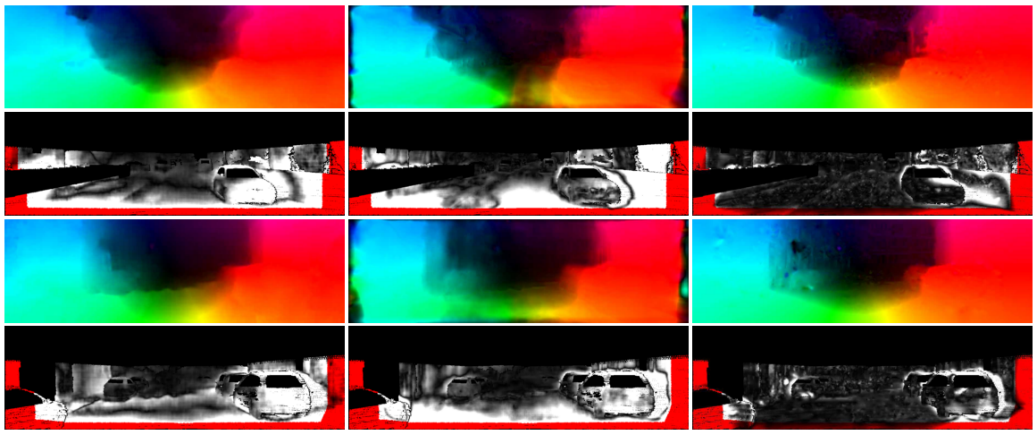

Figure 16: UnFlow Results [12]. From left to right: FlowNetS, UnsupFlowNet [20] and Unflow-CSS. Above flow, underneath flow error.

4.4 Conclusion

This work analyzed several DCNNs with their advantages and their drawbacks. FlowNet1.0 was the first break-through and the basic idea is used in all the other listed DCNNs, e.g. FlowNet2 stacks different FlowNet1.0 to increase accuracy. UnFlow is the latest published approach to improve the optical flow estimation. It is based on the idea of Auto Encoders, which results good at non-synthetic datasets related to the statements of the authors. No official results are online on different ranking lists. UnFlow took occlusions into account and differentiate between accuracy measure on occluded areas and non-occluded areas.

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 fig. 17).

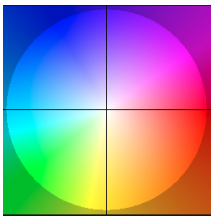

Figure 17: Color coding of optical flow images. No optical flow at all is indicated white and is in the middle of the circle. If the flow vector does look to the right, then the color is red. Cyan means to the right.

5.3 Endpoint Error (EPE)

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

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

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 by 3 pixels and you want to up-sample it to 9 by 9 pixels. Then, a filter (bilinear, linear, nearest-neighbor, bicubic, sinc and so on) can be applied to the 3x3 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.

$$\begin{aligned} f(x) &= 1 - |x| \quad for\ |x| < 1 \\ &= 0 \qquad\qquad\quad else \end{aligned} \tag{6}$$

In fig. 18, an example from avisynth [1] is shown, if the values of x are taken in the interval of {-1, 0, 1}.

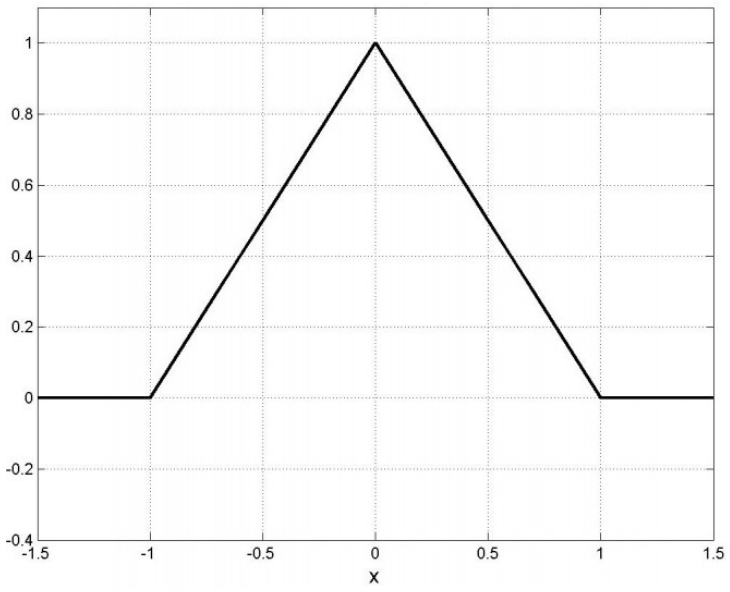

Figure 18: Bilinear interpolation for $x = \{-1, 0, 1\}$

Bibliography

[1] Avisynth. Resampling - Avisynth wiki. http://avisynth.nl/index.php/Resampling, 2015. 25

[2] Simon Baker, Daniel Scharstein, J P Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A Database and Evaluation Methodology for Optical Flow. *International Journal of Computer Vision*, 92(1):1–31, 2011. 2, 3

[3] Horn B.K.P. and Shunck B.G. Determining Optical Flow. *Artificial Intelligence*, 17:185–203, 1981. 2, 5, 7, 14

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

[5] Thomas Brox and Jitendra Malik. Large Displacement Optical Flow Descriptor Matching in Variational Motion Estimation.pdf. 33(3):500–513, 2011. 5

[6] Andrs Bruhn and Joachim Weickert. Towards ultimate motion estimation: Combining highest accuracy with real-time performance. *Proceedings of the IEEE International Conference on Computer Vision*, I:749–755, 2005. 15

[7] Yunjin Chen and Thomas Pock. Trainable Nonlinear Reaction Diffusion: A Flexible Framework for Fast and Effective Image Restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1256–1272, 2017. 6

[8] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning Optical Flow with Convolutional Networks. *IEEE International Conference on Computer Vision*, pages 2758–2766, 2015. 7, 8, 9, 10, 18, 19

[9] Junhwa Hur and Stefan Roth. MirrorFlow: Exploiting Symmetries in Joint Optical Flow and Occlusion Estimation. 16

[10] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. 2, 6, 7, 8, 9, 11, 12, 16, 19, 20

[11] Max Planck Institute for Intelligent Systems. MPI Sintel Dataset Results and Ranking. 3

[12] Simon Meister, Junhwa Hur, and Stefan Roth. UnFlow: Unsupervised Learning of Optical Flow with a Bidirectional Census Loss. 2017. 7, 16, 17, 22, 23

[13] Moritz Menze, Andreas Geiger, and Mpi Tübingen. Object Scene Flow for Autonomous Vehicles. 13

[14] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning Deconvolution Network for Semantic Segmentation. 1. 9

[15] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. pages 1–16, 2015. 13

[16] Zhe Ren, Junchi Yan, Bingbing Ni, Bin Liu, Xiaokang Yang, and Hongyuan Zha. Unsupervised Deep Learning for Optical Flow Estimation. *Proceedings of the 31th Conference on Artificial Intelligence (AAAI 2017)*, (Hollingworth 2004):1495–1501, 2017. 2, 12, 13, 14, 20, 21

[17] Sren Kaae Sønderby, Casper Kaae Sønderby, Lars Maaløe, and Ole Winther. Recurrent Spatial Transformer Networks. pages 1–15, 2015. 14

[18] Fridtjof Stein. Efficient Computation of Optical Flow Using the Census Transform. *Pattern Recognition*, 3175(August 2004):79–86, 2004. 4

[19] Deqing Sun. Secrets of Optical Flow Estimation and Their Principles Optical flow : motion of image pixels. page 904875, 2010. 15

[20] Jason J Yu, Adam W Harley, and Konstantinos G Derpanis. Back to Basics : Unsupervised Learning of Optical Flow via Brightness Constancy and Motion Smoothness. 23

[21] Jur Zbontar and Yann Lecun. Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. *Journal of Machine Learning Research*, 17:1–32, 2016. 7