computer graphics & vision

Inffeldgasse 16/2, A-8010 Graz

www.icg.tu-graz.ac.at

TUG

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, October 28, 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 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 these are trained supervised or unsupervised, and analyze their architecture as well their results.

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

1 Estimation Motion in Images Sequences

Motion analysis is one of the main tasks of computer vision. From the user's viewpoint, there is information of the dynamic behavior of an object or by the camera(s) itself. There is a wide range of applications, such as tracking objects or quantifying deformations, but the most low-level property is the estimation of dense motion field, where the displacement of each pixel is called Optical Flow (OF). In the recent 30 years OF gave a huge number of works, originally initialized by [1] and since then progressively improved. New demanding of applications lead to apply the OF to more complex problems in terms of large displacements or motion changes. Handling those problems are still open issues. Meanwhile comprehensive surveys [2,3,4] and the most recent focuses on variational methods (VM) [5] are published.

Deep Learning has replaced previous methods in many computer vision tasks, because of its overwhelming results. OF is one task that is dominated by classical methods[1]. Due to the field of autonomous driving vehicles a better vision system is needed to achieve better results in image segmentation. For example, the segmentation, can tell where is the street or identifies upcoming cars. A scene can be decomposed by several independently moving objects and this helps for several tasks like collision detection. We want to list deep learning methods and how often parts of classical methods are used within. Thus, we present state-of-the-art deep learning architectures in chronological order and explain some basic assumptions that hold for classical as well deep learning methods. After decomposing the architectures and their connection to classical methods, we want to discuss the results in the end.

This paper is basically divided int three sections: The first section deals with the applications of OF and what is it used for. The second section explains the OF in detail and the algorithm of Horn and Schunck [1]. The third section is about deep convolutional neural networks (DCNN) architectures and starts with the explaination of the basic elements of a neural network, such as perceptron, 2D convolution, pooling layer. Then, state-of-the-art DCNN (FlowNet, FlowNet2.0, DSTFlow, Unflow) are analyzed. Finally, the different DCNN architectures are compared by accuracy and computation time.

[1]Classical methods are meant no deep learning method, such as variational methods.

1.1 Background and Motivation

Optical flow was originally introduced by Horn and Schunk [1] and builds the fundament for many ongoing research topics. Almost every classical computer vision method was replaced by DCNN, since concisely better results can be determined. In contrast, except for OF, there still is not a single deep learning architecture which is significantly better than classical methods. A supervised learning [6] and [7] are considered as the best deep learning methods, but variational methods still yield better results. 3 datasets, Middlebury, Sintel and KITTI are used to compare results. They provide OF ground truth and hence can be used for

Ranking in Middlebury, Sintel and KITTI Dataset

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 methods.

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 [3].

The Middlebury dataset is small and both FlowNets yield bad results. Results of the other DCNNs have not been published by [3]. Only 8 frames are provided per scene and the frame sizes differ per scene.

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	3D Scene Flow

Table 2: Selected Algorithms from Sintel Dataset [8]

On a larger dataset, as given by Sintel in Table 2, the FlowNet architecture gives a lot better results than on the small dataset is 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	PSPO	6.15	Unknown
2	ISF	6.22	convolutional neural networks (CNN) + 3D Scene Flow
3	PRSM (TU Graz)	6.68	3D Scene Flow

Table 3: Selected Algorithms from KITTI2015 dataset [9].

The KITTI dataset is the most difficult dataset of those 3 selected data sets. It consists of 200 training scenes and 200 test scenes. The frames are taken, while the car is driving on the street as stereo images.

1.1.1 Applications

OF is a the low-level computer vision problem and therefore has a number of different application areas:

- Computer Vision:

  Video segmentation [10]: is very challenging, when objects move fast through the scenery. OF can help to identify moving objects.

  Object tracking [11]: OF can be used to track an object through several frames. For example, by use of a motion vector estimation can provide an estimation of object position.

  Structure from motion: is a technique that tries to reconstruct the 3D structure o from a sequence of images. It can be used for obstacle detection as well as for path-planning and navigation.

  Visual simultaneous localization and mapping (SLAM) [3]: Online mapping enables tracking in new environments.
- Computer Graphics:

  Video synthesis: is meant to synthesize frames of an video. More frames can be added to video and can change the scene, e.g. of a waterfall.

  Fluid reconstruction: is a method for constructing the velocity field of a fluid-flow from image intensity data.
- HCI:

  Hand gestures: are obtained through intensity images before the time-of-flight camera is invited.

  Facial expressions: Human have the ability to read emotions from facial expressions. For machine learning there is differed between the whole face expression and micro expressions that are parts of a full expression. OF can help to find out the facial movement amplitude and/or texture changes [12].

Pose estimation: OF can be used to estimate a 2D body pose [13]. Problems appear when people do not move any more, so there is no motion estimation.

2 Optical Flow

OF can be described as the estimation of object motions in a scene, the structure of the vehicle is represented as a vector field. The direction of each vector describes in which direction an object is moving. For determining the OF, a sequence of (at least 2) images is needed because OF algorithms estimate motions of pixels in consecutive image frames. More details of the OF is given in Section 2.1.2. In classical computer vision approaches, the first image is the ground truth and pixel that must be moved is looked up in the second image in the local neighborhood. In today's machine learning approaches the networks are often trained in the same way.

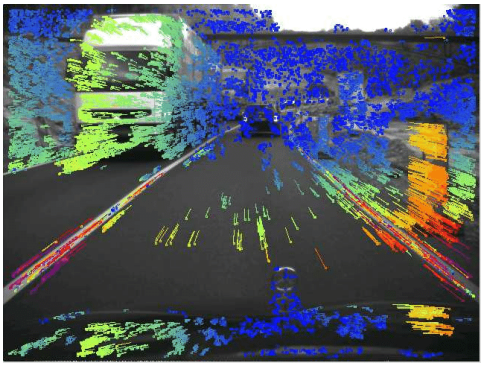


Figure 1: Optical flow vectors with oncoming traffic [14].

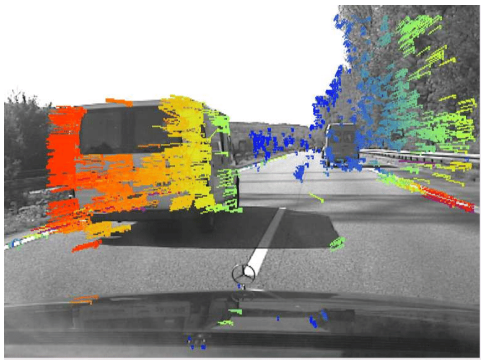


Figure 2: Optical flow vectors by following a car [14].

The best illustrating example is driving a car, where the flow vectors show the direction of the moving pixels. Corresponding flow field vectors are shown in Figure 1 and Figure 2. The colors denote the direction of the vector, which is explained in Section 6.2. The higher the intensity, the more longer the vector the more the pixel moved 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.

First, we will discuss some assumptions/constrain that relies on most OF algorithms. For example, the brightness of the pixel does not change

and pixel only makes a small movement from frame to frame. Second, we describe the method of Horn and Schunck in detail. We need to convert the continuous to a discrete model, so that we can write the energy minimization equation in matrix form. Last, we describe briefly the current challenges of OF.

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] are based on these assumptions/constraints. These assumptions/constraints still can be found in today's 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. We assume that we have a series of frames of a scene. So we have to define an discretized image

$$I : \{1, ..., M\} \times \{1, ..., N\} \tag{1}$$

where $M, N \in \mathcal{N}$ and for accessing the image frames in series, we define the parameter time t to distinguish each frame from another.

First of all, if we focus only on one scene point x, we assume that the brightness of this one scene point stays the same, just the location is different. Let u be the velocity vector $\binom{u_1}{u_2}$, that is added to the scene point x, so that the movement of the scene point x is determined.

Assumption 1-$\mathcal{F}$. *The brightness of a scene point in an image I does not change at time t:*

$$I(x, t) = Constant$$

where $x = \binom{x_1}{x_2}$ is the pixel position and $\triangle x = \binom{\triangle x_1}{\triangle x_2}$ the displacement vector, that shows a shift of a pixel.

Assumption 2-$\mathcal{F}$. *Assume the shift $\triangle x$ from one frame to another frame of one pixel to be small (less than a pixel, or smooth),*

$$I(x, t) \rightarrow I(x + \triangle x, t + \triangle t) \tag{2}$$

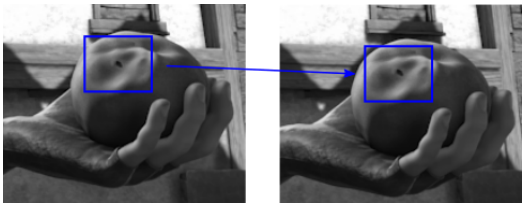


Figure 3: From frame to frame the hand is at a different position in the image and the BCA is valid, because of the constant brightness of the pixels in the blue rectangles.

then the intensity of that pixel does not change for a different t. Thus, we can use a 1^{st} order Taylor expansion and obtain to the linearized BCA, because the pixel movement from one frame to another frame is very small.

The brightness constancy equation can also be expressed

$$\frac{dI(x,t)}{dt} = \frac{\partial I}{dx}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0. \tag{3}$$

We can approximate the Equation 3 with the 1^{st}-order Taylor expansion:

$$I(x,y,t) \approx I(x_{t_0}, t_{t_0}) + \frac{\partial I}{dx}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} \tag{4}$$

Thus, the intensity of the scene does not change the term

$$I(x,t) = I(x_{t_0}, t_{t_0}) + 0 \tag{5}$$

and becomes equal. 2-$\mathcal{F}$ can be proved the 1^{st} order Taylor expansion by rearranging the Equation 5:

$$\begin{aligned}
& 0 = I(x+u, t+1) - I(x,t) && \text{Brightness Constancy Equation} \\
\Leftrightarrow\ & 0 \approx I(x, t+1) + I_x u - I(x,t) \\
\Leftrightarrow\ & 0 \approx [I(x,t+1) - I(x,t)] + I_x u = \frac{\partial I}{\partial x} \\
\Leftrightarrow\ & 0 \approx I_t + I_x u \\
\Leftrightarrow\ & 0 = I_t + \nabla_x I \cdot u && \square
\end{aligned} \tag{6}$$

The last line of Equation 6 yields basically the 1^{st}-order Taylor expansion, so that it is proved that the pixel movement can be linear approximated.

2.1.2 Optical Flow Constrained (OFC)

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 be 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 the flow field u_0 can be approximated via a warped image I_w and geometrically transform I_2 by u_0:

$$I_w(x) = I_2(x + u_0). \tag{7}$$

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

$$(\nabla_w(x))^T(u - u_0) + I_t(x) = 0. \tag{8}$$

The goal is to minimize Equation 8.

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. The question arises when does a neighboring pixel belong to one or to another flow field?

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. [15]) used for loss layers, see Section 4.4.3. There is an incorporate spatial coherence of the flow field $u = \begin{pmatrix} u_1(x) \\ u_2(x) \end{pmatrix}$ by minimizing a global model:

$$\min_{u_1,u_2} \int_\Omega |\nabla u_1(x)|^2 + |\nabla u_2(x)|^2 + \lambda |(\nabla I_w(x))^T(u - u_0) + I_t(x)|^2 dx. \tag{9}$$

Parameters: λ is a regularization constant; the larger λ, the smoother the flow.

The Equation 9 shows a continuous model of Horn and Schunck. In practice, it is better to discretize this model for calculation with matrices. For

the approximation we have two continuous images $I_1(x) = I(x,t)$ and $I_2(x) = I(x, t + \nabla_t)$. The time-derivative is approximated via forward differences:

$$I_t(x,t) \approx \frac{I(x,t+\nabla_t) - I(x,t)}{\nabla_t} = \frac{I_2(x) - I_1(x)}{\nabla_t}. \tag{10}$$

Moreover, our images are spatially discrete and hence the spatial derivatives are usually approximated via

$$\nabla I(x,t) \approx \frac{1}{2}\begin{pmatrix} \frac{I_1(x_1+\nabla_x,x_2-I_1(x_1+\nabla_x,x_2}{2\nabla_x} \\ \frac{I_1(x_1,x_2+\nabla_x,x_2-I_1(x_1,x_2+\nabla_x,x_2}{2\nabla_x} \end{pmatrix} + \frac{1}{2}\begin{pmatrix} \frac{I_2(x_1+\nabla_x,x_2-I_2(x_1+\nabla_x,x_2}{2\nabla_x} \\ \frac{I_2(x_1,x_2+\nabla_x,x_2-I_2(x_1,x_2+\nabla_x,x_2}{2\nabla_x} \end{pmatrix}. \tag{11}$$

$\nabla = \binom{\nabla_x}{\nabla_y} \in \mathbb{R}^{2MN \times MN}$ 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. We define the warped image $I_w = I(x + u_0)$, which is optained by geometrically transforming I by the velocity field u_0. $I_w^{\{x,y\}} = diag(D_{\{x,y\}} I_w)$ are the finite difference approximations 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:

$$\min_{U,V} \; ||\nabla U||^2 + ||\nabla V||^2 + \lambda ||I_w^x(U - U_0) + I_w^y(V - V_0) + I_t||^2. \tag{12}$$

Optimality conditions for U and V are given by:

$$\begin{aligned} \nabla^T \nabla U + \lambda I_w^x I_w^x (U - U_0) + I_w^y (V - V_0) + I_t = 0 \\ \nabla^T \nabla V + \lambda I_w^y I_w^x (U - U_0) + I_w^y (V - V_0) + I_t = 0. \end{aligned} \tag{13}$$

The optimality conditions can be rewritten into matrix notation:

$$\begin{pmatrix} \nabla^T \nabla + \lambda (I_w^x)^2 & \lambda I_w^x I_w^y \\ \lambda I_w^x I_w^y & \nabla^T \nabla + \lambda (I_w^y)^2. \end{pmatrix} \begin{pmatrix} U \\ V \end{pmatrix} = \lambda \begin{pmatrix} I_w^x (I_w^x U_0 + I_w^y) V_0 - I_t \\ I_w^y (I_w^x U_0 + I_w^y) V_0 - I_t \end{pmatrix}. \tag{14}$$

The system is large ($2MN \times 2MN$) 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 is there are no sharp edges between flow fields, because of the Euclidean distance and outliers can not be handled robustly.

2.3 Challenges of Optical Flow

One of the challenges of optical flow is to predict sharp edges as shown in Figure 4. Classical Computer Vision approaches as listed in the tables Table 1 and Table 2. In stereo images it is helpful to know occlusions, because these occur at the edges of objects, so more clear segmentation at edges can be learned.

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 in [16]. Pock formulated a reaction diffusion model [16], which can be applied to image denoising tasks.

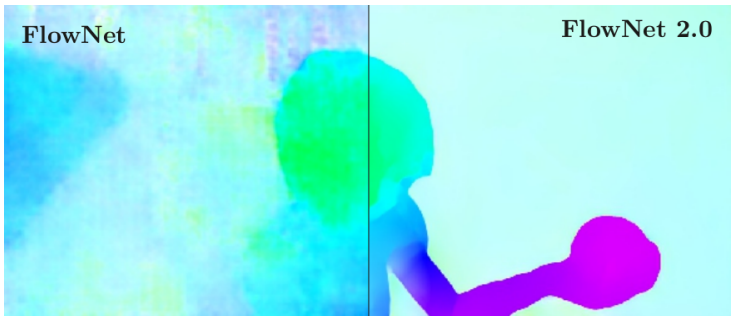


Figure 4: Comparison of FlowNet with FlowNet2 [6].

3 Deep Learning Architectures for Predicting Optical Flow

In this section, 4 DCNN approaches are chosen for predicting optical flow and are described in details. Before that we describe the basic structure of a CNN, such as multilayer perceptron and 2d convolution. Two of the chosen approaches are supervised learning, FlowNet and FlowNet2. 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 [17], 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.

4 Convolutional Neural Networks (CNN)

Neural Networks consists of an artificial neuron, also called a perceptron [18]. A perceptron gets some inputs, i.e. $x_1, x_2, x_3 \in \{0, 1\}$, that can be expressed as binary encoding. The input x_1 can be 1 and the others can be be zero. Each input is weighted by $w_1, w_2, w_3 \in \mathbb{R}$, that expresses the importance of the input.

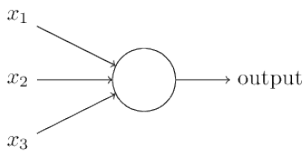


Figure 5: Perceptron with 3 inputs (x_1, x_2, x_3). [18]

The output of a neuron is expressed by the weighted sum $\sum_j x_j w_j = x \cdot w$ which is less or greater than a certain threshold value. In Equation 15 threshold is rewritten to $b \equiv -threshold$ and the cases can be algebraic expressed as two cases

$$output = \begin{cases} 0 & if \sum_j w_j x_j + b \leq 0 \\ 1 & if \sum_j w_j x_j + b > 0 \end{cases} \tag{15}$$

that can be 0 if $\leq$ a threshold or 1 if ¿ than a threshold and hence the neuron is activated. Usually there is a non-linearity ($\sigma = tanh, ReLU, sigmoid, \ldots$) warped around a neuron. Only non-linear activation functions allow a neural network to compute nontrivial problems by using a small number of nodes. The output of a perceptron

$$\sigma(w_j x_j + b) \tag{16}$$

is squased to a range, e.g. a sigmoid function maps the input between 0 and 1. If we want to solve more complicated problems, we need to put more perceptrons together. This is called a multi-layer perceptron (MLP). The network is splitted into 3 parts: input, hidden and output layer. The input layer can be pixels of images or speech recordings. The hidden layer(s) follows after the input layer and ends before the output layer. These layers are responsible for what the network is learning. Then, the output layer, in our case a single neuron. If we want to classify numbers as in MNIST, we have has input 64×64 grey scaled images from numbers 0..9. We would need

4096 neurons by this image size. The output layer, which is a single neuron, indicates if input image is not a 9 by less than 0.5 and values greater than 0.5 indicating the input image is a 9.

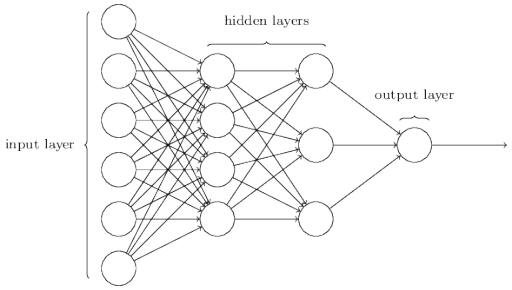


Figure 6: MLP with input, hidden and output layer(s) [18].

This kind of networks are called *feedforwward* neural networks, because the output of the previous layer is the input of the next layer. In other words, there are no loops in the network as in recursive neural networks (RNN). This thesis focuses on CNN, a specialization of neural networks. Up to now we explained fully-connected layers (see Figure 6), where each input neuron is connected with each output neuron. In CNN a convolution is calculated where a filter with the depth of the input layer is sliding over input. In literature this is often called *sliding dot product*. Different methods to calculate convolutions can be seen in Section 4.1.

4.1 2D Convolution

A 2D Convolution is a convolution in x and y direction. A kernel window, in Figure 7, is a 3×3 sized and is sliding from top left horizontally to the right, row-by-row. This is the case if *striding = 1*. If *striding* > 1, e.g. we take 2, then every 2^{nd} pixel is left out. The output y of the convolution can be produced by dot products of the filter kernel h with the input x, as described

$$y[m,n] = x[m,n] * h[m,n] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x[i,j] \cdot h[m-i, n-j]. \qquad (17)$$

4.2 Supervised Deep Learning Approach: FlowNet and FlowNet2

FlowNet [20] (see Figure 8) was the first approach, which showed "optical flow" as learning problem. FowNet2 [6] is the improved version and increases

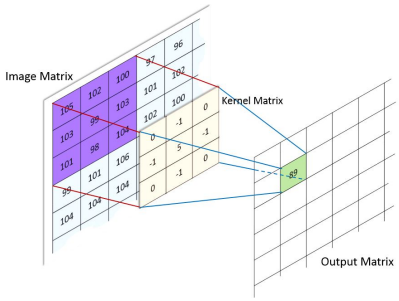


Figure 7: 2D Convolution with 3×3 filter [19].

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. [20] used the idea of Zuntar and LeCun [21] by training a Siamese architecture (two consecutive images given as input, mostly chosen from stereo cameras setup or from a scene or videos) to predict the similarities of images. Zuntar and LeCun [21] learned per patches and did spatial aggregation as post-processing, while in FlowNet1 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, as suffer mostly from noise and some post-processing is necessary, which are will be discussed in the following section.

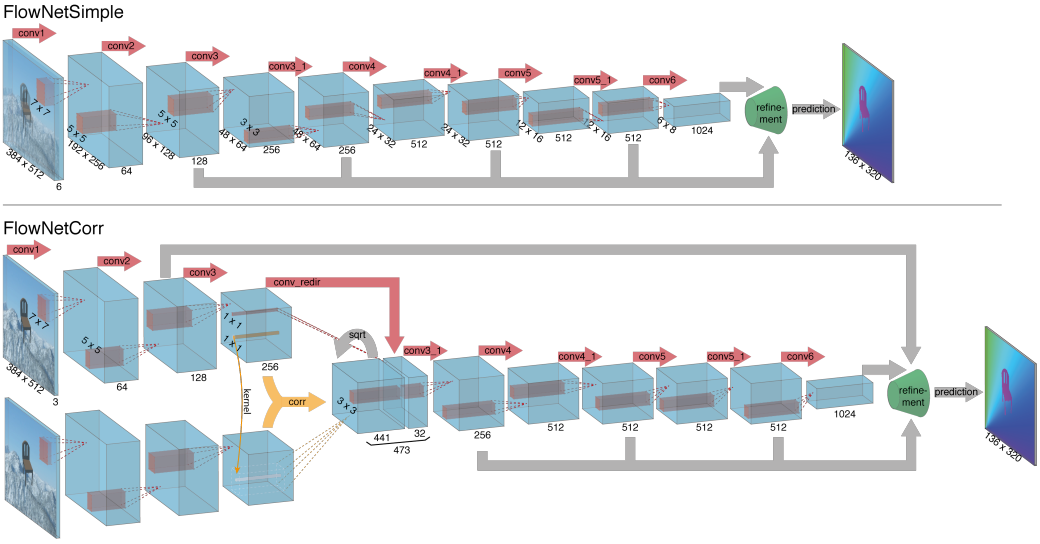


Figure 8: The network architectures: FlowNetSimple [20] and FlowNetCorr [20]n

4.2.1 Correlation Layer

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

$$f_1, f_2 : \mathcal{R}^2 \rightarrow \mathcal{R}^c, \tag{18}$$

where a patch f have a height and a width that is mapped to c output channels.

Figure 9 restricts to comparison of 2 patches, where x_1 and x_2 are the centers of the first and second patch. Dosovitskiy et al. [20] tries not to compare all patches $w^2 \cdot h^2$ and is computational expensive. Moreover, it describes 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 a similar pixel can be found. The assumption of small displacement holds as well the BCA (see Section 2.1.1). Therefore, a maximal displacement d [20] is introduced, which compares pixel in a maximum distance of d in the neighborhood (10 pixels) [20]. Then, the feature map is concatenated as one volume.

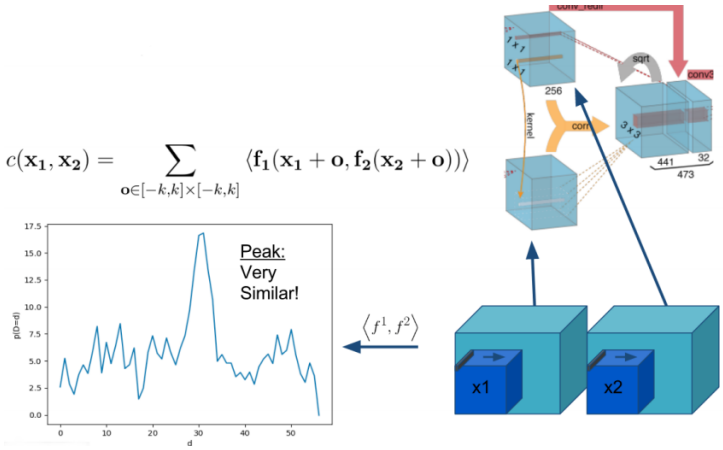


Figure 9: Correlation Layer [6] (right top) and comparison of the features.

4.2.2 Pooling and Refinement Layer

After the concatenation, we have a huge volume which contains the point correspondences. 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 8) and a refinement step can be applied. Pooling layers downgrade the number of parameters. It takes the maximum value of a two by two kernel window. So fewer parameters must be learned. A detailed view[2] of the original code is given. This helps to filter noisy activations [22] by using a single value of a receptive field. A side effect is that it retains stable activations in upper layers [22]. A drawback is the loss of spatial information within the receptive field and is unbecomingly for sharp edges that are required for semantic segmentation [22]. In order to keep the end-to-end structure, an up-sampling procedure must be introduced to obtain the same height and width as of the input images. The image size is growing during this process until to the original size. This bilinear (see section 6.3.1) up-sampling approach leads to a simple computational less expensive up-sampling method [6].

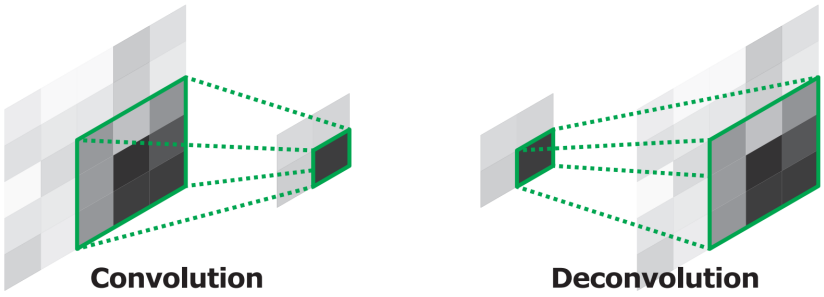


Figure 10: Example of convolution and deconvolution [22] .

4.3 Improved FlowNet: FlowNet2

In this section, we analyze 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 4.2). Ilg et al. [6] state a decrease of the endpoint error (EPE) so that the accuracy is four times higher [6], but it is a little bit slower than the FlowNet [6]. The new idea is 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 tries 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 2-$\mathcal{F}$ redundant. Ilg et al.

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

state that to deal with small displacements, they used a smaller striding parameter [6] 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.

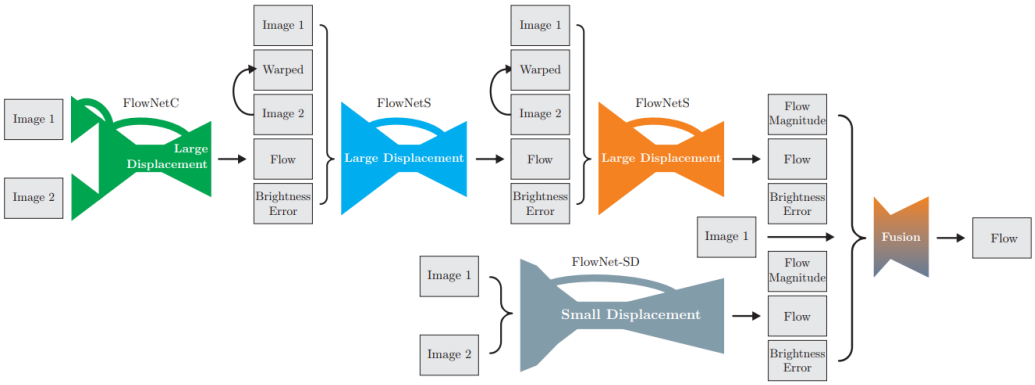


Figure 11: Schematic overview of FlowNet2 [6]. Large displacements are computed by combining mutliple FlowNets. The braces indicates the concatenation of several outputs. The brightness error is calculated by the difference of the 1st image and the 2nd image warped with the previously computed flow.

In Figure 11, FlowNetC (FlowNet with correlation architecture, compare Figure 8), two FlowNetS (FlowNet without correlation architecture, see Figure 8) and one FlowNet-SD (FlowNet for learning small displacement).

End-to-end networks per pixel prediction often suffer to be very noisy. The use of refinement layers which relies on the formulation of (see Section 2.3), helps to improve the results. Chen and Pock [16] propose a refinement neural network for that.

4.3.1 Stacking CNN

To stack neural networks, we need to warp the output of one neural network with the input of another neural network. We take as an example two FlownetS as Net1 and Net2 that should be warped together. Different configuration can be applied such as pre-train the Net1 and randomly initialized Net2 or pre-train both. Some observations are listed:

1. Without warping: Increases performance on the dataset Flying Chairs[3]

[3]Synthetic dataset: https://lmb.informatik.uni-freiburg.de/resources/datasets/FlyingChairs.en.html

(20.000 images), but decreases on Sintel (1048 images) [6]. The dataset Flying Chairs is huge compared Sintel and that is why Sintel tends to overfit.

2. With warping: Stacking is helpful for better results [6].
3. Intermediate loss after Net1: This counteracts against the vanishing gradient problem, when the stacked network end-to-end is trained.
4. Best result on Sintel: Net1 has been trained, while Net2 istrained with warping [6].
5. Best result on Chairs: It is the last configuration in Table 4. The first network is trained and the second is trained after warping [6].

Stacking networks has overfitting as a drawback. The neural network is melting to one big network. Ilg et al. suggest training the networks one after another and warping helps refining the optical flow [6].

Stack architecture	Training enabled		Warping included	Warping gradient	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 4: FlowNet2 - comparison of two stacking FlowNetS [6]. 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.

4.4 Unsupervised deep learning approach: DSTFlow

Beside FlowNet2, dense spatial transform (DST) Flow [7], an end-to-end CNN, was published in 2017. This section will discuss a supervised DCNN approach that is working quite well. Unsupervised means that there are no labeled data. Applications for such CNNs are often meant to autonomous driving and video segmentation [23]. Unfortunately, no implementation is online available 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 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 [7].

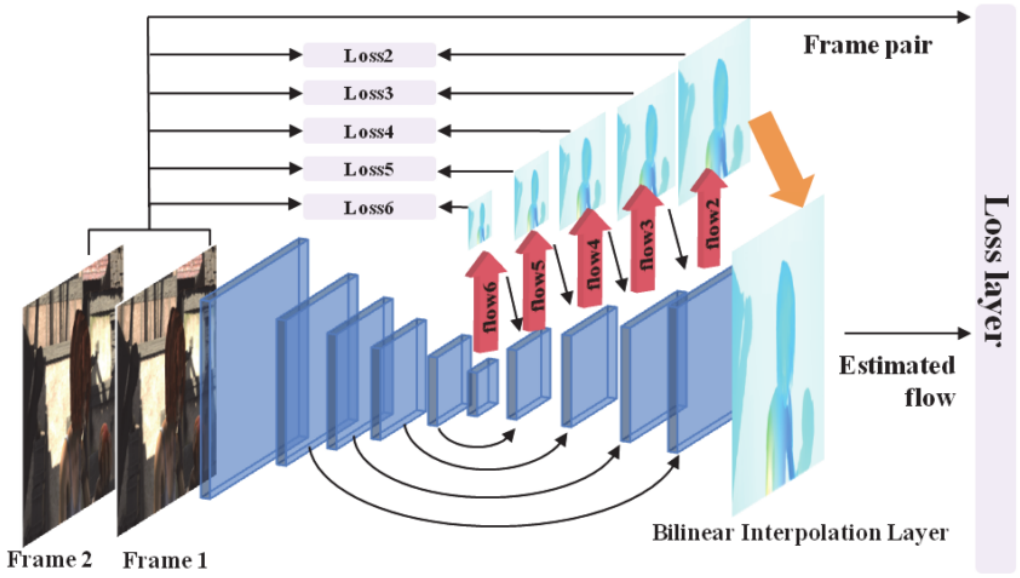


Figure 12: DSTFlow Architecture [24]. The convolutional hidden layers are akin to FlowNetS with convolution and deconvolution.

- Loss layer: For large displacements, they decided to use the global estimation method of Brox and Malik, which is a refinement of the traditional method of Horn and Schunk.

4.4.1 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 [7]. Hence the input is 3+3 = 6 depth [7] (compare Figure 12). 3 stands for the depth of each color channel RGB (Red, Green, Blue). After some convolution layers, there are some deconvolution layers [7], 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 [7].

4.4.2 Sampling Network

The sampling network gets as an input the output of the localization network. The sampling kernel (bilinear, see Section 6.3.1) is applied at the input features U. The indices m, n indicate the value of U at position (m,n) [7]. While (x_i^s,y_i^s) defines the spatial location. V is the output. The forward-path

can be achieved through:

$$V_i^c = \sum_{n=1}^{H} \sum_{m=1}^{W} U_{nm}^c max(0, 1 - |x_i^s - m|) max(0, 1 - |y_i^s - n|). \tag{19}$$

They 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. The $max()$ function makes the derivation easy as it is known from the ReLU activation function:

$$\frac{\partial V_i^s}{\partial U_{nm}^c} = \sum_{n=1}^{H} \sum_{m=1}^{W} max(0, 1 - |x_i^s - m|) max(0, 1 - |y_i^s - n|) \tag{20}$$

$$\frac{\partial V_i^s}{\partial x_i^s} = \sum_{n=1}^{H} \sum_{m=1}^{W} U_{nm}^c max(0, 1 - |y_i^s - n|) \begin{cases} 0 & \text{if} \quad |m - x_i^s| \geq 1 \\ 1 & \text{if} \quad m \geq x_i^s \\ -1 & \text{if} \quad m < x_i^s. \end{cases} \tag{21}$$

4.4.3 Loss Layer

In [7], a traditional loss function used in learning-free variational methods, such as by Brox [25] 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 (penalty, such as l_1-norm) is important for differencing between flow predictions in the neighborhood. The Charbonnier penalty Figure 13 (blue line) is close to the l_1-norm. The l_1-norm is the best result and is the most robust convex function [26].

To define the dense spatial transform DST loss, it is used the Charbonnier penalty [27]:

$$\Psi(s) = \sqrt{(s^2 + \beta^2)} \tag{22}$$

with $\beta = 0.001$. 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 an 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 be found in Section 2.1.1 at of BCA assumption:

$$l_D = \int_{\Omega} \Psi(|I_2(x + w) - I_1(x)|^2 + \gamma |\nabla I_2(x + w| - \nabla I_1(x)|^2) dx. \tag{23}$$

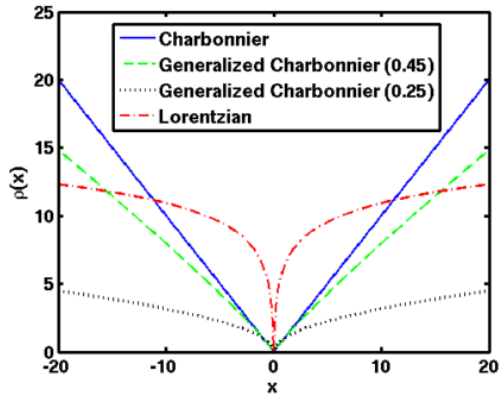


Figure 13: "Penalty functions for the spatial terms: Charbonnier ($\epsilon = 0.001$), generalized Charbonnier ($a = 0.45$ and $a = 0.25$)" [26].

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:

$$l_S = \int_\Omega \Psi(|\nabla u(x)|^2 + |\nabla v(x)|^2) dx. \tag{24}$$

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

$$l_{DST} = l_D + \alpha l_S. \tag{25}$$

4.4.4 Multi-scale loss accumulation

This layer is for improving the accuracy of the predicted flow. As shown in Figure 12, there are 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.

4.5 UnFlow

UnFlow is another end-to-end network, that is based on the principle of auto-encoding (AE). The DCNN is aimed at 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 [17].

2. Train of FlowNetC with unsupervised loss [17].
3. Iterative refinement: Stacking several CNNs like FlowNet2.0 [6, 17].

In Figure 14 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 needed for occlusion detection [17]. In the loss function occlusions are taken into account by penalizing them with the Charbonnier penalty [17] (see Section 4.4.3). Occluded pixels will be penalized constantly [17] to avoid that all pixels are occluded.

Moreover, Meister et al. [17] decided not rely on the BCA 1-$\mathcal{F}$, because it is not robust due to illumination changes, what is suitable for realistic environments. It is better to use the census transform [17], which "can compensate multiplicative and additive illumination changes" [17].

In Figure 15 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.

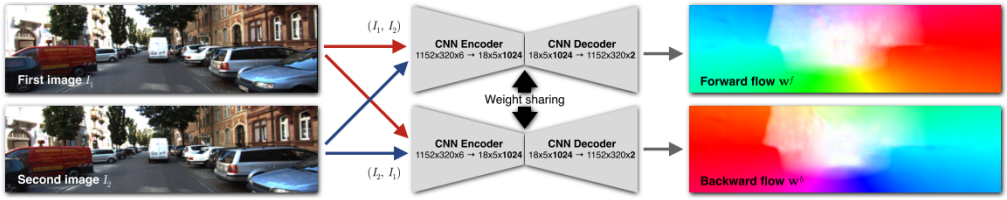


Figure 14: Schematic bidirectional [17, 28] training of FlowNet [6].

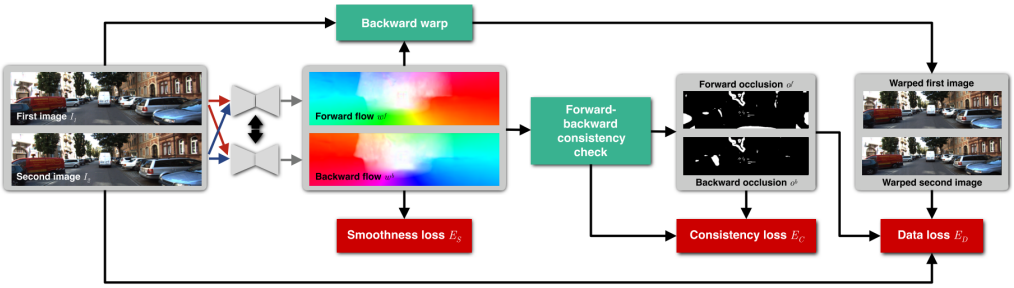


Figure 15: Schematic Architecture of UnFlow [17].

5 Comparison of Results

Several DCNNs have been described in the previous sections. The results of each architecture differ, depending on the used dataset (Sintel, Middlebury, Chairs, Kitti). However, as we learned from the previous descriptions, FlowNet is often used as the 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.

5.1 FlowNet and FlowNet2

Method	Sintel Clean		Sintel Final		KITTI		Middleb. train		Middleb. test		Chairs	Time (sec)	
	train	test	train	test	train	test	AEE	AAE	AEE	AAE	test	CPU	GPU
EpicFlow	2.40	4.12	3.70	6.29	3.47	3.8	0.31	3.24	0.39	3.55	2.94	16	-
DeepFlow	3.31	5.38	4.56	7.21	4.58	5.8	0.21	3.04	0.42	4.22	3.53	17	-
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	-	-	2.62	-	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 5: FlowNet results presented by Dosovitskiy et al. [20].

FlowNet is an overall term for basically two different network types, the FlowNetS (simple) and the FlowNetC (correlation), which can be seen in Figure 8.

Table 5 lists the EPE (see Section 6.3) 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) result the FlowNetC is 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 synthetic 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 [20].

In Table 6 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 5. 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 **??** the results comparing FlowNet2 to FlowNet and state-of-the-art algorithms are shown. FlowNet2 outperforms FlowNet by its segmentation

Method	Sintel Clean AEE		Sintel Final AEE		KITTI2015 AEE	KITTI2015 fl-all		Middlebury AEE		Time (sec)	
	train	test	train	test	train	train	test	train	test	CPU	GPU
EpicFlow	2.27	4.12	3.56	6.29	9.27	27.18%	**27.10%**	0.31	0.39	4.2	-
DeepFlow	2.66	5.38	3.57	7.21	10.63	26.52%	29.18%	**0.25**	0.42	5.1	-
FlowNetS	4.50	6.96	5.45	7.52	-	-	-	1.09	-	-	1.8
FlowNetC	4.31	6.85	5.87	8.51	-	-	-	1.15	-	-	3.2
FlowNet2-s	4.55	-	5.21	-	16.42	56.18%	-	1.27	-	-	7
FlowNet2-ss	3.22	-	3.85	-	12.84	41.03%	-	0.68	-	-	14
FlowNet2-css	2.51	-	3.54	-	11.01	35.19%	-	0.54	-	-	31
FlowNet2-css-ft-sd	2.50	-	3.50	-	11.18	34.10%	-	0.43	-	-	31
FlowNet2-CSS	2.10	-	3.23	-	**8.94**	29.77%	-	0.44	-	-	69
FlowNet2-CSS-ft-sd	2.08	-	3.17	-	10.07	30.73%	-	0.38	-	-	69
FlowNet2	**2.02**	**3.96**	3.14	6.02	10.06	30.37%	-	0.35	0.52	-	123
FlowNet2-ft-sintel	(1.45)	4.16	(2.01)	5.74	9.84	28.20%	-	0.35	-	-	123
FlowNet2-ft-kitti	3.43	-	4.66	-	(2.30)	(8.61%)	11.48%	0.56	-	-	123

Table 6: Results presented by Dosovitskiy et al. [20] on the Sintel dataset. 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.

performance and it can even compete with state-of-the-art algorithms. Ilg et al. also mentioned that FlowNetC outperforms FlowNetS, because of the modified dataset and training schedules [6]. 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 [6] after some fine-tuning, which is not clearly described. On Middlebury, the training error is quite good, while the test error shows signs of overfitting.

In FlowNet2, the 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.

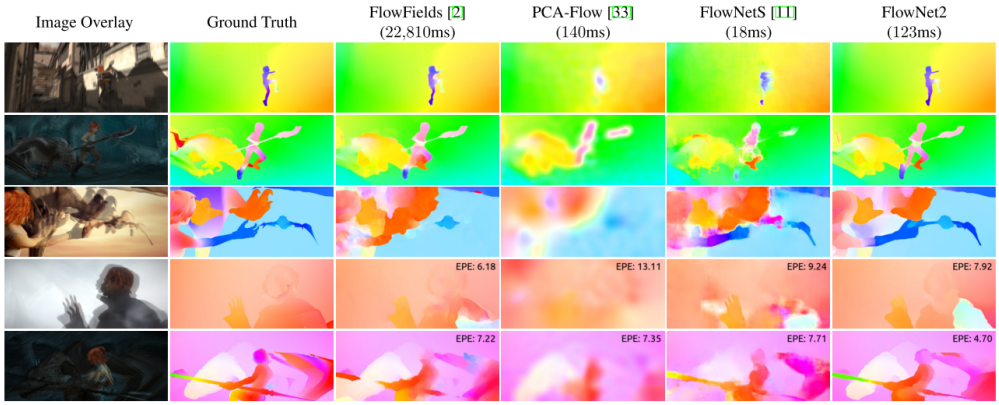


Figure 16: Results [6] of FlowNet2, which results close to FlowFields (best approach). FlowNet2 is very slow compared to FlowNetS and PCA-Flow.

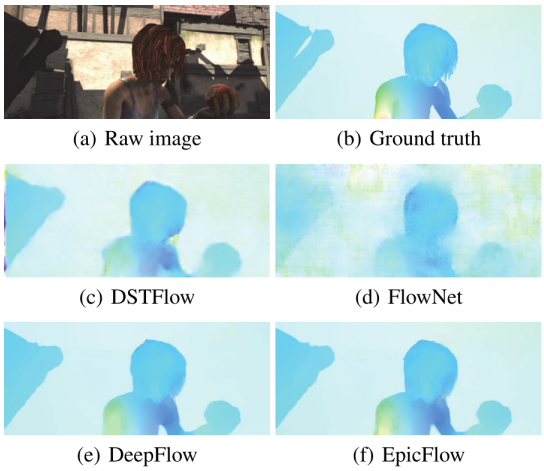
5.2 DSTFlow with FlowNet and FlowNet2

Method	Sintel Clean				Sintel Final				KITTI2015				Chairs	Time (sec)	
	train		test		train		test		train		test		test	CPU	GPU
	occ	noc	occ	noc	occ	noc	occ	noc	occ	noc	fl-all	fl-all			
EpicFlow	2.40	1.23	4.12	1.36	3.70	2.63	6.29	3.06	9.57	4.45	0.28	0.27	2.94	16	-
DeepFlow	3.31	1.78	5.38	1.77	4.56	3.07	7.21	3.34	13.89	6.75	0.31	0.30	3.53	17	-
FlowNetS(C+S)	*3.66	*2.82	6.96	-	*4.44	*3.99	7.76	-	14.19	8.12	0.51	-	3.04	-	0.2
DSTFlow(Chairs)	6.93	5.05	10.40	5.20	7.82	5.97	11.11	5.92	24.30	14.23	0.52	-	5.11	65	2.5
DSTFlow(KITTI)	7.10	5.26	10.95	5.87	7.95	6.16	11.80	6.70	16.79	6.69	0.36	0.39	6.86	-	0.08
DSTFlow(Sintel)	*6.16	*4.17	10.41	5.30	*7.38	*5.45	11.28	6.16	23.79	13.88	0.55	-	5.68	-	0.08
DSTFlow(C+K)	7.51	5.74	-	-	8.29	6.55	-	-	22.93	12.81	0.48	-	5.86	-	0.08
DSTFlow(C+S)	*6.47	*4.61	10.84	5.62	*6.81	*4.91	11.27	6.02	25.98	15.89	0.53		5.92	-	0.08

Table 7: DSTFLow results with EPE on OCC and NOC pixels. Asterisks mean that they were trained on the own data (overfitting). [7]

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

In Table 7 several training combinations are shown, i.e. DSTFlow(C+K) [7] means DSTFlow trained with the dataset Chairs and refined with the dataset KITTI or DSTFLOW(C+S) [7] 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 Figure 12, 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.



(a) Raw image (b) Ground truth

(c) DSTFlow (d) FlowNet

(e) DeepFlow (f) EpicFlow

Figure 17: DSTFlow results on the Sintel dataset [7].

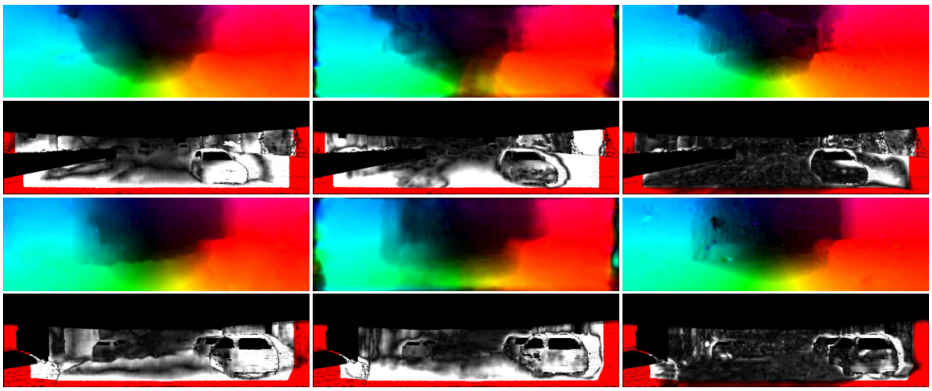


Figure 18: UnFlow Results [17]. From left to right: FlowNetS, UnsupFlowNet [29] and Unflow-CSS. Above flow, underneath flow error. KITTI2012 maps between 0 (black) and white if ≥ 5 pixels.

5.3 Unflow with FlowNet and DSTFlow

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 Figure 18, outperforms DSTFlow [17]. On Middlebury, Unflow C Cityshapes outperforms FlowNetS. This is an evidence that the network can generalize the learned data. In contrast, on Sintel, the Unflow CNN cannot outperform FlowNetS+ft nor FlowNet2, but it is better than DSTflow.

Method	Sintel Final AEE		KITTI2015 AEE	KITTI2015 fl-all		Middlb. AEE	
	train	test	train	train	test	train	test
FlowNetS+ft	(4.44)	7.76	-	-	-	0.98	-
DSTFlow (KITTI)	7.95	11.80	16.79	36%	39%	-	-
FlowNet2-C	-	-	-	-	-	-	-
FlowNet2-CSS	3.23	-	8.94	29.77%	-	0.44	-
FlowNet2-ft-kitti	4.66	-	(2.30)	(8.61%)	10.41%	0.56	-
Unflow-C-Cityscapes	8.23	-	10.78	33.89%	-	0.85	-
Unflow-C	8.64	-	8.80	28.94%	-	0.88	-
Unflow-CS	7.92	-	8.14	23.94%	-	0.65	-
Unflow-CSS	7.91	10.22	8.10	23.27%	-	0.65	-
Unflow-CS-ft	11.99	-	(2.25)	(9.24%)	12.55%	0.64	-
Unflow-CSS-ft	13.65	-	(1.86)	(7.40%)	11.11%	0.64	0.76

Table 8: UnFlow Results [17]. The accuracy is compared on public benchmarks. AEE is the average endpoint error that is similar to EPE. Parentheses indicate that the network is trained on the same dataset as it is tested on. The cross are results of Ilg et al.

5.4 Conclusion

This work analyzed several DCNNs with their advantages and their drawbacks. FlowNet1 was the first break-through and the basic idea is used in all the other listed DCNNs, e.g. FlowNet2 stacks different FlowNet1 to increase accuracy. UnFlow is the latest published approach to improve the optical flow estimation. It is based on the idea of æae, 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.

6 Appendix

6.1 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

6.2 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 19).

6.3 EPE

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

$$L_{epe} = \frac{1}{N} \sum \sqrt{(U - U')^2 + (V - V')^2}, \tag{26}$$

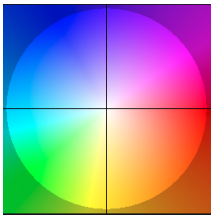


Figure 19: Color coding of optical flow images. No optical flow at all is indicated white and is in the middle of the circle. If the flow vector does look to the right, then the color is red. Cyan means to the right.

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.

6.3.1 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:

$$f(x) = \begin{cases} 1 - |x|, & \text{if } |x| < 1 \\ 0, & \text{otherwise.} \end{cases} \tag{27}$$

In Figure 20, an example from avisynth[4] is shown, if the values of x are taken in the interval of {-1, 0, 1}.

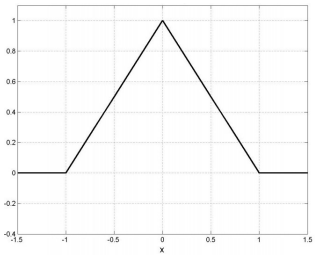


Figure 20: Bilinear interpolation for $x = \{-1, 0, 1\}$. The resampling kernel 27 is applied on x.

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

Bibliography

[1] H. Berthold and S. Brian, "Determining Optical Flow," *Proc. Conference on Artif. Intell.*, vol. 17, no. 1-3, pp. 185–203, 1981. 1, 2, 5, 7, 12, 18

[2] C. Stiller and J. Konrad, "Estimating motion in image sequences," *IEEE Signal Processing Magazine*, vol. 16, no. 4, pp. 70–91, 1999. 1

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

[4] M. Amar and B. Patrick, "Computation and analysis of image motion: A synopsis of current problems and methods," *Intern. Journal of Computer Vision*, vol. 19, pp. 29–55, 1996. 1

[5] J. Weickert, A. Bruhn, T. Brox, and N. Papenberg, *A Survey on Variational Optic Flow Methods for Small Displacements*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. 1

[6] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks," *Computer Society Conference on Computer Vision and Pattern Recognition*, 2017. 2, 9, 11, 13, 14, 15, 16, 20, 22

[7] Z. Ren, J. Yan, B. Ni, B. Liu, X. Yang, and H. Zha, "Unsupervised Deep Learning for Optical Flow Estimation," *Proc. Conference on Artif. Intell.*, 2017. 2, 16, 17, 18, 23

[8] D. Butler, J. Wulff, G. Stanley, and M. Black, "A naturalistic open-source Movie for Optical Flow Evaluation," *European Conference on Computer Vision*, pp. 611–625, 2012. 2

[9] M. Menze, C. Heipke, and A. Geiger, "Joint 3D Estimation of Vehicles and Scene Flow," in *Intern. Society for Photogrammetry and Remote Sensing.*, 2015. 3

[10] Y. H. Tsai, M. H. Yang, and M. J. Black, "Video Segmentation via Object Flow," in *Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3899–3908, June 2016. 3

[11] K. Kale, S. Pawar, and P. Dhulekar, "Moving Object tracking using Optical Flow and Motion Vector Estimation," in *Intern. Conf. on Reliability, Infocom Technologies and Optimization*, pp. 1–6, Sept 2015. 3

[12] B. Allaert, I. M. Bilasco, and C. Djeraba, "Consistent Optical Flow Maps for full and micro facial expression recognition," in *VISAPP*, vol. 5 of *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, (Porto, Portugal), pp. 235–242, Feb. 2017. 3

[13] J. Romero, M. Loper, and M. J. Black, "FlowCap: 2D Human Pose from Optical Flow," in *Pattern Recognition, Proc. 37th German Conference on Pattern Recognition (GCPR)*, vol. LNCS 9358, pp. 412–423, Springer, 2015. 4

[14] F. Stein, "Efficient Computation of Optical Flow Using the Census Transform," *Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 3175, pp. 79–86, 2004. 4

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

[16] Y. Chen and T. Pock, "Trainable Nonlinear Reaction Diffusion: A Flexible Framework for Fast and Effective Image Restoration," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1256–1272, 2017. 9, 15

[17] S. Meister, J. Hur, and S. Roth, "UnFlow: Unsupervised Learning of Optical Flow with a Bidirectional Census Loss," *CoRR*, vol. abs/1711.07837, 2017. 9, 19, 20, 24

[18] F. Rosenblatt, *Principles of neurodynamics: perceptrons and the theory of brain mechanisms*. Report (Cornell Aeronautical Laboratory), Spartan Books, 1962. 10, 11

[19] H. Kazemi, *A comprehensive Tutorial towards 2D Convolution and Image Filtering.*, 2017. http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html. 12

[20] 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," *Intern. Conf. on Computer Vision*, pp. 2758–2766, 2015. 11, 12, 13, 21, 22

[21] 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. 12

[22] H. Noh, S. Hong, and B. Han, "Learning Deconvolution Network for Semantic Segmentation," *Intern. Conf. on Computer Vision*, pp. 1520–1528, 2015. 14

[23] M. Menze and A. Geiger, "Object Scene Flow for Autonomous Vehicles," *Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3061–3070, 2015. 16

[24] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," *International Conference on Learning Representations*, vol. abs/1511.06434, pp. 1–16, 2015. 17

[25] T. Brox and J. Malik, "Large Displacement Optical Flow: Descriptor Matching in Variational Motion Estimation," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 3, pp. 500–513, 2011. 18

[26] D. Sun, S. Roth, and M. Black, "Secrets of Optical Flow Estimation and Their Principles," *Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2432–2439, 2010. 18, 19

[27] A. Bruhn and J. Weickert, "Towards ultimate Motion Estimation: Combining highest Accuracy with real-time Performance," *Intern. Conf. on Computer Vision*, vol. I, pp. 749–755, 2005. 18

[28] J. Hur and S. Roth, "MirrorFlow: Exploiting Symmetries in Joint Optical Flow and Occlusion Estimation," in *Intern. Conf. on Computer Vision*, pp. 312–321, IEEE Computer Society, 2017. 20

[29] J. Yu, A. Harley, and K. Derpanis, "Back to Basics: Unsupervised Learning of Optical Flow via Brightness Constancy and Motion Smoothness," in *European Conference on Computer Vision*, pp. 3–10, Springer International Publishing, 2016. 24