

# Learning to Extract Motion from Videos in Convolutional Neural Networks

Damien Teney

Martial Hebert

The Robotics Institute  
Carnegie Mellon University  
Pittsburgh, PA, USA

## Abstract

*This paper shows how to extract dense optical flow from videos with a convolutional neural network (CNN). The proposed model constitutes a potential building block for deeper architectures to allow using motion without resorting to an external algorithm, e.g. for recognition in videos. We derive our network architecture from signal processing principles to provide desired invariances to image contrast, phase and texture. We constrain weights within the network to enforce strict rotation invariance and substantially reduce the number of parameters to learn. We demonstrate end-to-end training on only 8 sequences of the Middlebury dataset, orders of magnitude less than competing CNN-based motion estimation methods, and obtain comparable performance to classical methods on the Middlebury benchmark. Importantly, our method outputs a distributed representation of motion that allows representing multiple, transparent motions, and dynamic textures. Our contributions on network design and rotation invariance offer insights nonspecific to motion estimation.*

## 1. Introduction

The success of convolutional neural networks (CNNs) on image-based tasks, from object recognition to semantic segmentation or geometry prediction, has inspired similar developments with videos. Example applications include activity recognition [30, 40, 41], scene classification [37], or semantic segmentation of scenes with dynamic textures [35]. The appeal of CNNs is to be trainable end-to-end, *i.e.* taking raw pixel values as input, and learning their mapping to the output of choice, identifying appropriate intermediate representations in the layers of the network. The natural application of this paradigm to videos involves a 3D volume of pixels as input, made of stacked consecutive frames. The direct application of existing architectures on such inputs has shown mixed results. An alternative is to first extract optical flow or dense trajectories with an external algorithm [30, 41, 40] and feed the CNN with this information in addition to the pixel values. The success of

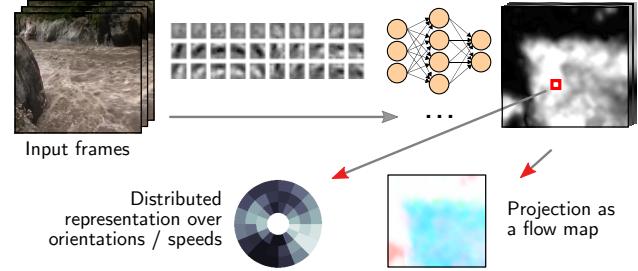


Figure 1. The proposed CNN takes raw pixels as input and produces features representing evidence for motion at various speeds and orientations. These can be projected as a traditional optical flow map. First-layer kernels (pictured) typically identify translating patterns in the image.

this approach can be explained by the intrinsically different nature of spatial and temporal components, now separated during a preprocessing. The extraction of motion regardless of image contents is not trivial, and has been addressed by the long-standing line of successful optical flow algorithms. Conversely, the end-to-end training of CNNs on videos for high-level tasks has shown limited capability for identifying intermediate representations of motion. In this paper, we show that specifically training a CNN to extract optical flow can be achieved once some key principles are taken into account.

We leverage signal processing principles and how motion manifests itself in the frequency domain of a spatiotemporal signal (Section 3) to derive convolutions, pooling and non-linear operations able to map input pixels to a representation of motion (Section 4). The resulting network is designed as a building block for deeper architectures addressing higher-level tasks. The current practice of extracting optical flow as an independent preprocessing might be suboptimal, as the assumptions of an optical flow algorithm may not hold for a particular end application. The proposed approach would potentially allow to fine-tune the motion representation with the whole model in a deep learning setting.

The proposed network outputs a distributed representation of motion. The penultimate layer comprises, for each

pixel, a population of neurons selective to various orientations and speeds. These can represent a multimodal distribution of activity at a single spatial location, and represent non-rigid, overlapping, and transparent motions that traditional optical flow usually cannot. The distributed representation can be used as a high-dimensional feature by subsequent applications, or decoded into a traditional map of the dominant flow. The latter allows training and evaluation with existing optical flow datasets.

The motivation for the proposed approach is not direct competition with existing optical flow algorithms. The aim is to enable using CNNs with videos in a more principled way, using pixel input for both spatial and temporal components. Instead of considering the flow of a complete scene as the end-goal, we rather wish to identify local motion cues without committing to an early scene interpretation. This contrasts with modern optical flow approaches, which often perform implicit or explicit tracking and/or segmentation. In particular, we avoid motion smoothness and rigidness priors, and spatial scene-level reasoning. This makes the features produced by the network suitable to characterize situations that break such assumptions, *e.g.* with transparent phenomena and dynamic textures [8, 35]. As a downside, our evaluation on the Sintel benchmark shows inferior performance to state-of-the-art techniques. This confirms that a *scene-level* interpretation of motion requires such priors and higher-level reasoning. In particular, we do not perform explicit feature tracking and long-range matching, which are the highlights of the best-performing methods on this dataset (*e.g.* [39, 4]).

The contributions of this paper are fourfold. (1) We derive, from signal processing principles, a CNN able to learn mapping pixels to optical flow. (2) We train this CNN end-to-end on videos with ground truth flow, then demonstrate performance on the Middlebury benchmarks comparable to classical methods. (3) We show how to enforce strict rotation invariance within a CNN through weight-sharing, and demonstrate significant benefit for training on a small dataset with no need for data augmentation. (4) We show that the distributed representation of motion produced within our network is more versatile than traditional flow maps, able to represent phenomena such as dynamic textures and multiple, overlapping motions. Our pretrained network is available as a building block for deeper architectures addressing higher-level tasks. Its training is significantly less complex than competing methods [12] while providing similar or superior accuracy on the Middlebury benchmark.

## 2. Related work

**Learning spatiotemporal features** Several recent works have used CNNs for classification and recognition in videos. Karpathy *et al.* [19] consider the large-scale clas-

sification of videos, but obtain only a modest improvement over single-frame input. Simonyan and Zisserman [30] propose a CNN for activity recognition in which appearance and motion are processed in two separate streams. The temporal stream is fed with optical flow computed by a separate algorithm. The advantage of using separate processing of spatial and temporal information was further examined in [40, 41]. We propose to integrate the identification of motion into such networks, eliminating the need for a separate algorithm, and potentially allowing to fine-tune the representation of motion. Tran *et al.* [37] proposed an architecture to extract general-purpose features from videos, which can be used for classification and recognition. Their deep network captures high-level concepts that integrate both motion and appearance. In comparison, our work focuses on the extraction of motion alone, *i.e.* independently of appearance, as motivated by the two-stream approach [30]. Learning spatiotemporal features outside of CNN architectures was considered earlier. Le *et al.* [24] used independent subspace analysis to identify filter-based features. Konda *et al.* [20] learned motion filters together with depth from videos. Their model is based on the classical energy-based motion model, similarly to ours. Taylor *et al.* [33] used restricted Boltzmann machines to learn unsupervised motion features. Their representation can be used to derive the latent optical flow, but was shown to capture richer information, useful *e.g.* for activity recognition. The high-dimensional features produced by our network bear similar benefits. Earlier work by Olshausen *et al.* [27] learned sparse decompositions of video signals, identifying features resembling the filters learned in our approach. In comparison to all works mentioned above, we focus on the extraction of motion independently of spatial appearance, whereas decompositions such as in [27] result in representations that confound appearance and temporal information.

**Extraction of optical flow** The estimation of optical flow has been studied for several decades. The basis for many of today’s methods dates back to the seminal work of Horn and Schunk [16]. The flow is computed as the minimizer of data and smoothness terms. The former relies on the conservation of a measureable image feature (typically corresponding to the assumption of brightness constancy) and the latter models priors such as motion smoothness. Many works proposed improvements to these two terms (see [14] for a recent survey). Heeger *et al.* [15] proposed a completely different approach, applying spatiotemporal filters to the input frames to sample their frequency contents. This method naturally applies to sequences of more than two frames, and relies on a bank of hand-designed filters (typically, Gaussian derivatives or spatiotemporal Gabor filters [1]). Subsequent improvements [29, 31, 38] focused on the design of those filters. They must balance the sampling of narrow regions of the frequency spectrum, *i.e.*, to accurately estimate mo-

tion speed and orientation, while retaining the ability to precisely locate the stimuli in image. A practical consequence of this tradeoff is the typically blurry flow maps produced by the method. This historically played in favor of the more popular approach of Horn and Schunk. Another downside of the filter-based approach was the computational expense of convolutions. Our work revisits Heeger’s approach, motivated by two key points. On the one hand, applying spatiotemporal filters naturally falls in the paradigm of convolutional neural networks, which are currently of particular interest for analyzing videos. On the other hand, modern advances can overcome the two initial burdens of the filter-based approach by (1) learning the filters using backpropagation, and (2) leveraging GPU implementations of convolutions.

Concurrently with our own developments, Fisher *et al.* proposed another CNN-based method (Flownet [12]). They obtain very good results on optical flow benchmarks. In comparison to our work, they train a much deeper network that requires tens of thousands of training images. Our architecture is derived from signal processing principles, which contains fewer weights by several orders of magnitude. This allows training on much smaller datasets. The final results in [12] also include a variational refinement, essentially using the CNN to initialize a traditional flow estimation. Our procedure is entirely formulated as a CNN. This potentially allows fine-tuning when integrated into a deeper architecture.

A number of recent works [8, 7, 35, 36] studied the use of spatiotemporal filters to characterize motion in *e.g.* transparent and semi-transparent phenomena, and dynamic textures such as a swirling cloud of smoke, reflections on water, or swaying vegetation. These works highlighted the potential of filter-based features, and the need for motion representations – such as those produced by our approach – that go beyond displacement (flow) maps.

**Invariances in CNNs** One of our technical contributions is to enforce rotation invariance by tying groups of weights together. In contrast to weight *sharing* which involves weights at different layers, this applies to weights of a same layer. Encouraging or enforcing invariances in neural networks has been approached in several ways. The convolutional paradigm ensure translation invariance by reusing weights between spatial locations. Other schemes of weight sharing were proposed in a simple model in [11], and later in [23] with a method to learn which weights to share. No published work discussed the implementation of strict rotation invariance in CNN to our knowledge. In [9, 22], schemes akin to ensemble methods were proposed for invariance to geometric transformations. In [28] and more recently in [17], a network first predicts a parametric transformation, used to rotate or warp the image to a canonical state before further processing. Our approach, in addition to the actual invariance, has the benefit of reducing the number

of weights to learn and facilitates the training. Soft invariances, *e.g.* to contrast can be encouraged by specific operations, such as local response normalization (LRN, *e.g.* in [21]). We also use a number of such operations. Note that this paper abuses of the term “invariance” for cases more accurately involving *equivariance* or *covariance* [18].

### 3. Filter-based motion estimation

Our rationale for estimating motion with a convolutional architecture is based on the motion energy model [1]. Classical implementations [15, 26, 38] are based on convolutions with hardcoded spatiotemporal filters (*e.g.* 3D Gabors or Gaussian derivatives). The convolution of a signal with a kernel in the spatiotemporal domain corresponds to a multiplication of their spectra in the frequency domain. Convolutions with a bank of bandpass filters produce measurements of energy in these bands, which are then suitable for frequency analysis of the signal. A pattern moving in a video with a constant speed and orientation manifests itself as a plane in the frequency domain [1, 13], and the signal energy entirely lies within this plane. It passes through the origin, and its tilt corresponds to the motion orientation and speed in the image domain. Classical implementations have used various schemes to identify the best-fitting plane to the energy measurements. In our model, we learn the spatiotemporal filters together with additional layers to decode their responses into the optical flow. Importantly, transparent patterns in an image moving with different directions or speeds correspond to distinct planes in the frequency domain. The same principle can thus identify multiple, overlapping motions.

### 4. Proposed network architecture

Our network is fully convolutional, *i.e.*, it does not contain any fully connected layers. Each location in the output flow map is linked to a spatially-limited receptive field in the input, and each pixel of a training sequence can thus be seen as a unique training point. We describe below each layers of our network in their feedforward order. We use  $x_{ijk}^\ell$  to refer to the scalar value at coordinates  $(i, j, k)$  in the 3D tensor obtained by evaluating the  $\ell$ th layer. Indices  $i$  and  $j$  refer to spatial dimensions,  $k$  to feature channels. We use a colon (:) to refer to all elements along a dimension. We denote with  $*_{2D}$  and  $*$  convolutions in two and three dimensions, respectively. In contrast to CNNs used for image recognition, the desired output here is dense, *i.e.* a 2D flow vector for every pixel. To achieve this, all convolutions use a  $1px$  stride and the pooling (Eq. 5) a  $2px$  stride, all with appropriate padding. The output is thus at half the resolution of the input. Our experiments use bilinear upsampling (except otherwise noted) to obtain flow fields at the original resolution.

## 4.1. Network input

The input to the network is the  $H \times W \times F$  volume of pixels made by stacking  $F$  successive grayscale frames of a video (typically,  $F=3$ ). The desired network output is the flow between frames  $[F/2]$  and  $[F/2]+1$ .

$$x_{::k}^1 = \text{kth frame of the video } \forall k=1\dots F. \quad (1)$$

## 4.2. Invariance to brightness and contrast

The estimated motion should be insensitive to additive changes of brightness of the input. Since the subsequent processing will be local, instead of subtracting the average brightness over the whole image, we subtract the local low-frequency component:

$$x_{::k}^{\ell+1} = x_{::k}^\ell - (x_{::k}^\ell *_{2D} \mathcal{H}^0) \quad \forall k, \quad (2)$$

where  $\mathcal{H}^0$  denotes a fixed, 2D Gaussian kernel of standard deviation  $w/3$ . Note that this operation could also be written as a convolution with a center-surround filter. We then ensure invariance to local contrast changes with a normalization using the standard deviation in local neighbourhoods:

$$x_{ijk}^{\ell+1} = x_{ijk}^\ell / \text{std}_{i',j' \in \Omega(i,j)}(x_{i'j'k}^\ell) \quad \forall i, j, k, \quad (3)$$

where  $\Omega(i, j)$  refers to the square region of length  $w$  centered on  $(i, j)$ .

The two above operations proved essential to learn subsequent filters with little training data. They can be seen as a local equivalent to a typical image-wide whitening [25]. This formulation is better suited to the subsequent local processing of a fully convolutional network.

## 4.3. Motion detection

This key operation convolves the volume of pixels with learned 3D kernels. They can be interpreted as spatiotemporal filters that respond to various patterns moving at different speeds.

$$x_{::k}^{\ell+1} = x_{::k}^\ell * \mathcal{H}_k^1 + b_k^1 \quad \forall k=1\dots MO, \quad (4)$$

where  $\mathcal{H}_k^1$  are  $MO$  learned convolution kernels of size  $w \times w \times F$ , and  $b_k^1$  the associated biases. The constants  $M$  and  $O$  respectively fix the number of independent kernels and the number of orientations explicitly represented within the network (Section 4.7).

## 4.4. Invariance to local image phase

The learned kernels used as motion detectors above typically respond to lines and edges in the image. The estimated motion should however be independent of such image structure. Classical models [15, 31] account for this using pairs of quadrature filters, though this is not trivial to enforce with our learned filters. Instead, we approximate a phase-invariant response as follows. (1) The response of the convolution is rectified by pointwise squaring. Responses

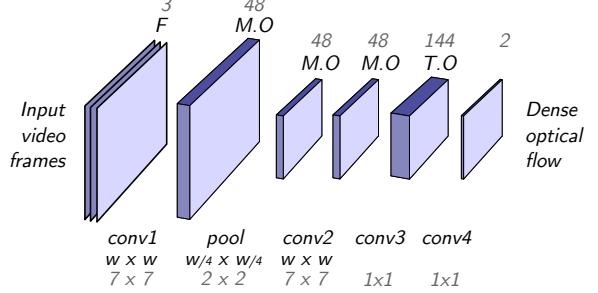


Figure 2. The proposed neural network takes raw pixels of  $F$  video frames as input, and outputs a dense optical flow at half its resolution. The network comprises 2 convolutional layers, 1 pooling layer, and 2 pixelwise weights ( $1 \times 1$  convolutions). Dimensions of receptive fields and feature maps are shown, with numbers chosen in our implementation in gray. Normalizations and non-linearities are not shown.

out of phase by  $180^\circ$  (e.g. dark-bright and bright-dark transitions) then give a same response. (2) We apply a spatial max-pooling. The wavenumber of the pattern captured by our kernels of size  $w$  is at least  $2/w$  cycles/px. The worst-case phase shift of  $90^\circ$  then corresponds to a spatial shift of  $w/4$  px. We maxpool responses over windows of size  $w/4$  with a fixed stride of 2, and thus approximate a phase-invariant response at the price of a lowered resolution.

$$x_{ijk}^{\ell+1} = \max_{i',j' \in \Omega'(2i,2j)} (x_{i'j'k}^\ell)^2 \quad \forall i, j, k, \quad (5)$$

where  $\Omega'(2i, 2j)$  refers to the square region of length  $[w/4]$  centered on  $(2i, 2j)$ .

## 4.5. Invariance to local image structure

The estimated motion should be independent from the amount and type of texture in the image. To account for intensity differences of patterns at different orientations at any particular location (e.g. a grid pattern of horizontal lines crossing fainter vertical ones), we normalize the responses by their sum over all orientations [15]:

$$x_{ijk}^{\ell+1} = x_{ijk}^\ell / (\sum_{k'} x_{ijk'}^\ell + \epsilon) \quad \forall i, j, k, \quad (6)$$

where  $\epsilon$  is a small constant to avoid divisions by a small value in low-texture areas of the image. The sum is performed over feature channels  $k'$  that correspond to the  $O$  variants of  $k$  at all orientations (see Section 4.7).

To account for the aperture problem, we allow local interaction by introducing an additional convolutional layer with  $MO$  learned kernels  $\mathcal{H}^2$  of size  $w \times w \times MO$ .

$$x_{::k}^{\ell+1} = x_{::k}^\ell * \mathcal{H}_k^2 + b_k^2 \quad \forall k=1\dots MO, \quad (7)$$

$$x_{ijk}^{\ell+1} = \max(x_{ijk}^\ell, 0). \quad (8)$$

The classical hardcoded models typically use here 2D convolutions with Gaussian kernels. Our experiments showed that supervised training lead to similar kernels, although slightly non-isotropic, and modeling some cross-channel and center-surround interactions.

## 4.6. Decoding into flow vectors

The features maps at this point represent evidence for different types of motion at every pixel. This evidence is now decoded with a hidden layer, a softmax nonlinearity and a linear output layer:

$$x_{::k}^{\ell+1} = x_{::k}^\ell * \mathcal{H}_k^3 + b_k^3 \quad \forall k=1...TO \quad (9)$$

$$x_{ijk}^{\ell+1} = e^{x_{ijk}^\ell} / \sum_{k'} e^{x_{ijk'}^\ell} \quad \forall i, j, k=1...TO \quad (10)$$

$$x_{::k}^{\ell+1} = x_{::k}^\ell * \mathcal{H}_k^4 + b_k^4 \quad \forall k=\{1, 2\}, \quad (11)$$

where  $T$  is a constant that fixes the number of hidden units. The decoding is performed pixelwise, *i.e.*  $\mathcal{H}_k^3$  and  $\mathcal{H}_k^4$  are  $1 \times 1$ . Intuitively, the activations of the hidden layer (Eq. 9) represent scores for motions at  $S$  and  $O$  discrete speeds and orientations, of which the softmax picks out the highest. Assuming a unimodal distribution of scores (*i.e.* a single motion at any pixel), the output layer interpolates these scores and maps them to a 2D flow vector for every pixel (see Section 6).

## 4.7. Invariance to in-plane rotations

In our context, rotation invariance implies that a rotated input must produce a correspondingly rotated output. Note the contrast with image recognition where rotated inputs should give a *same* output. All of our learned weights (Eq. 4–9) are split into groups corresponding to discrete orientations. The key is to enforce these groups of weights to be equivalent, *i.e.* so that they make the same use of features from the preceding layer at the same *relative* orientations. In addition, convolutional kernels need to be 2D rotations of each other. These strict requirements allow us to maintain only a single version of the weights at a canonical orientation, and generate the others when evaluating the network (see Fig. 4). During training with backpropagation, the gradients are aligned with this canonical orientation and averaged to update the single version of the weights.

Formally, let us consider a convolutional layer<sup>1</sup>  $\ell+1$ . The feature maps  $x^\ell$  (respectively  $x^{\ell+1}$ ) are split into  $O$  ( $P$ ) groups of  $M$  ( $N$ ) channels. For example, in Eq. 7,  $O=P$  and  $M=N$ . The groups of channels correspond to regular orientations  $\theta_i^\ell$  ( $\theta_j^{\ell+1}$ ) in  $[0, 2\pi[$ . Considering the convolution weights  $\mathcal{H}$  and their slice  $h_{imjn}$  the 2D kernel acting on the input (respectively output) channel of orientation  $\theta_i^\ell$  ( $\theta_j^{\ell+1}$ ), we constrain the weights as follows:

$$h_{imjn} = \text{rotate2D}(h_{i'mj'n}) \quad (12)$$

$$\forall i, i', j, j', m, n \text{ s.t. } \cos(\theta_{j'}^{\ell+1} - \theta_i^\ell) = \cos(\theta_j^{\ell+1} - \theta_i^\ell) \quad (13)$$

Eq. 12 ensures that convolution kernels are rotated versions

<sup>1</sup>The general formulation applies to convolutional layers as well as to our pixelwise weights (Eq. 9, 11), in which case the 2D rotation of the kernel has no effect.

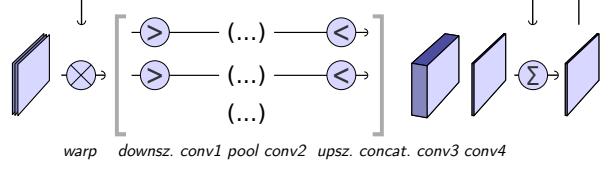


Figure 3. We bring two modifications to the basic network (Fig. 2) for multiscale processing. First, we apply the network on several downsampled versions of the input, concatenating the feature maps before the decoding stage. Second, we add a recurrent connection to warp the input frames according to the estimated flow, iterating its evaluation for a fixed number of steps. This is inspired by the classical “coarse-to-fine” strategy for optical flow, and designed to estimate motions larger than the receptive field of the network output units.

of each other (implemented with bilinear interpolation) and Eq. 13 ensures that the same weights are applied to input channels representing a same *relative* orientation with respect to a given output channel of the layer. In other words, the weights are shifted between each group so as to act similarly on channels representing the same relative orientations (see Fig. 4). In Eq. 7 and 9,  $N=N'$ . In Eq. 4,  $N=1$ . In Eq. 11,  $N=2$  with  $\theta_j^{\ell+1} = \{0, \pi/2\}$ .

It follows that the number of convolution kernels to explicitly maintain is reduced from  $OPMN$  to  $\lceil O/2 \rceil MN$ . In Eq. 7 for example, in our implementation with  $O=P=12$  orientations, this amounts to a decrease by a factor 24. It allows training on small amounts of data with lower risk of overfitting. This also negates the need to artificially augment the dataset with rotations and flips.

## 5. Multiscale processing

Equations 1–11 form a complete network that maps pixels to a dense flow field. However, the detection of large motion speeds is limited by the small effective receptive field of the output units, due to the limited number of convolutional layers. We remedy this in two ways (Fig. 3) without increasing the number of weights to train. First, we apply the network (Eq. 1–7) on multiple downsampled versions of the input frames. The feature maps are brought back to a common resolution by bilinear upsampling and concatenated before the decoding stage (Eq. 9–11).

Second, we add a recurrent connection to the network that warps the input frames according to the current estimate of the flow. The evaluation runs through the recurrent connection for a fixed number of steps. This is inspired by the classical coarse-to-fine warping strategy [10]. It allows the model to approximate the flow iteratively. Note that the recurrent connection to the warping layer is not a strictly linear operation contrary to typical recurrent neural networks. Training is performed by backpropagation through the unfolded recurrent iterations, but not through the recurrent connection itself. Since the result of each iter-

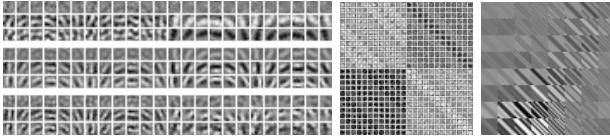


Figure 4. Convolutional kernels  $\mathcal{H}^1$ ,  $\mathcal{H}^2$ , and pixelwise weights  $\mathcal{H}^3$  learned on the Middlebury dataset (selection). The obvious structure is enforced by constraints that ensure rotation invariance of the overall network.

ation is summed to give the final output, we can append and sum the same loss (Section 6) at the end of each unfolded iteration.

## 6. Implementation and training

We train the weights ( $\mathcal{H}^1, \dots, \mathcal{H}^4$ ) and biases ( $b^1, \dots, b^4$ ) end-to-end with backpropagation. Even though the network ultimately performs a regression to flow vectors, we found more effective to train it first for *classification*. First, we pick a number  $TO$  of flow vectors uniformly in the distribution of training flows. We train the network for nearest-neighbour classification over these possible outputs, with a logarithmic loss over Eq. 10. Second, we add the linear output layer (Eq. 11) and initialize the rows of  $\mathcal{H}_k^4$  with the vectors used for classification. Second, the network is fine-tuned with a Euclidean (“end-point error”) loss over the decoded vectors. That fine-tuning has a marginal effect in practice. The softmax values are practically unimodal and sum to one by construction, hence they can directly interpolate the vectors used for classification with a linear operation. We believe the 2-step training is helpful because the Euclidean loss alone does not reflect well the quality (*e.g.* smoothness) of the flow. The classification loss guides the optimization towards a better optimum.

Considering the above, the softmax values (Eq. 10) constitute a distributed representation of motion, where each dimension corresponds to a different orientation and speed. Feature maps at this layer can encode multimodal distributions over this representation and represent patterns that optical flow cannot.

## 7. Experimental evaluation

We present three sets of experiments. (1) We evaluate the non-standard design choices of the proposed architecture through ablative analysis. (2) We compare our performance versus existing methods on the Middlebury and Sintel benchmarks. (3) We demonstrate applicability to dynamic textures and multiple transparent motions that goes beyond traditional optical flow. Code and trained models are available on the author’s website [34].

### 7.1. Ablative analysis

Our ablative analysis uses the public Middlebury dataset, with the sequences split into halves for training (Grove2,

	EPE (px)	AAE (°)
Full model, $F=3$ , $O=12$ , 10 scales, 1 rec.iter.	0.66	6.9
(1) Number of input frames $F=2/5$	0.67 0.90	6.8 8.7
(2) No center-surround filter	N.C.	N.C.
(3) No local normalization	0.67	6.9
(4) Hard-coded $\mathcal{H}_k^1$ : Gauss. deriv.	0.78	8.4
(5) No L1-normalization over orientations	N.C.	N.C.
(6) No pooling for phase invariance	0.93	11.2
(7) ReLU after conv1 (default: squaring)	N.C.	N.C.
(8) No constraints for rotation invariance	N.C.	N.C.
(9) Number of orientations $O=6/8/16$	1.65 0.72 0.65	26.6 8.1 6.6
(10) Loss: classification/regression	0.66/0.83	7.0 8.7
(default: 2-step, classification (logarithmic) then regression (Euclidean))		
(11) Number of scales: 4/8/16	0.69 0.66 0.67	6.9 6.8 6.9
(12) Recurrent iterations: 2/3/4/5	0.57 0.55 0.56 0.57	6.5 6.4 6.7 6.8

Table 1. We evaluate every non-standard design choice by retraining a modified network. N.C. denotes networks for which the training did not converge within a reasonable number of iterations. See discussion in Section 7.1.

Method	Middlebury		Sintel	private	Time	
	public	private	clean	final	(sec)	
FlowNetS [12]	1.09	13.28	–	4.44	7.76	0.08
FlowNetS + refinement [12]	0.33	3.87	0.47 4.58	4.07	7.22	1.05
DeepFlow [39]	0.21	3.24	0.42 4.22	4.56	7.21	17
LDOF [5]	0.45	4.97	0.56 4.55	6.42	9.12	2.5
Classic++ [32]	0.28	–	0.41 3.92	8.72	9.96	–
FFV1MT [31]	0.95	9.96	1.24 11.66	–	–	–
Proposed	(0.45)	(5.47)	0.70 6.41	9.36	10.04	6
Proposed + refinement as [12]	(0.35)	(4.10)	0.58 5.22	9.47	10.14	7

Table 2. Comparison with existing algorithms on the Middlebury and Sintel benchmarks. We report average end-point errors (EPE, in pixels) average angular errors (AAE, in gray, in degrees), and execution times per frame on Sintel. Numbers in parentheses correspond to the sets used for training the model.

RubberWhale, Urban3) and testing (Grove3, Dimetrodon, Hydrangea). For each run, we modify the network in one particular way, retrain it from scratch, and report its performance on the test set (Table 1). We observe that all preprocessing and normalization steps (2–3) have a positive impact, and some are even necessary for the training to converge at all (at least with the small dataset used in these experiments). Operations inspired by the classical implementations of the motion energy model, *i.e.*, (8) rectification by squaring of filter responses and (6) normalization across orientations, proved beneficial as well. This shows the benefit for our principled approach to network design. We perform a comparison to the classical, hardcoded filter-approach approach [15] by setting the first filters to Gaussian derivatives (4). Although learned filters are visually somewhat similar (Fig. 4), learned kernels clearly perform better. This confirms the general benefit of a data-driven approach to motion estimation.

### 7.2. Performance on optical flow benchmarks

We use the Middlebury and Sintel benchmarks for evaluation of networks trained from scratch on their respective training sets. A network trained on the smaller Middlebury dataset performs decently on Sintel sequences with small motions. However, most include much faster motions that had to be retrained for.

**Middlebury** Flow maps estimated by our method on the Middlebury dataset [2] are generally smooth and accurate (see Fig. 5). Most errors occur near boundaries of objects that become, or cause occlusions. Although our flow maps remain generally more blurry than those of state-of-the-art methods, some fine details are remarkably well preserved (*e.g.* Fig. 5, second row). This blurriness, or imprecision in the spatial localization of motions, is a well-known drawback of filter-based motion estimation. Convolutional kernels of smaller extent would be desirable to provide better localization, but the extent of its response in the frequency domain (Section 3) would correspondingly increase, which would imply a coarser sampling of the signal spectrum and lesser accuracy in motion direction and speed. Comparisons with existing methods show performance on the level of classical methods. We obtain much better performance than the recent implementation of Solari *et al.* [31] of a filter-based method with no learning.

**Sintel** The MPI Sintel dataset [6] contains computer-generated scenes of a movie provided in “clean” and “final” versions, the latter including atmospheric effects, reflections, and defocus/motion blur. Flow maps estimated by our method (Fig. 5) are often smooth. Flows in scenes with small motions are usually accurate, but they lack details at the objects’ borders and near small image details. Although this is partly alleviated by our recurrent iterative processing within the network, large errors remain in scenes with fast motions. This is reflected by a poor quantitative performance (Table 2). Additional insights can be gained by examining the flow maps (Fig. 5). Errors arise not on the estimates of large motions, but on their localization, in particular near zones of (dis)occlusions. This is obvious *e.g.* in Fig. 5, last row, with a thin wing flapping over a blank sky. Although the actual motion (in yellow) is detected, it spills on both sides of the thin wing structure. Since such occlusions are caused by large motions, they result in a large penalty in EPE. As argued before [6], good overall performance in such situations clearly require reasoning over larger spatial and/or temporal extent than the local motions cues that our method was designed around.

Interesting comparisons can be made with the competing approach Flownet [12]. It uses a more standard deep architecture with numerous convolutional and pooling layers. It also includes a variational refinement to improve the precision of motion estimates from the CNN. As discussed in [12], this refinement cannot correct large errors of correspondingly large motions. For comparison, we applied this same post-processing to our own results. Our results right off the CNN on Middlebury are already accurate, and the post-processing brings only marginal improvement (Table 2). The refinement on Flownet has a stronger effect: the output of their CNN is less precise, and it benefits more from this refinement. Looking at the Sintel dataset, the sit-

uation is very different. The main metric (the average EPE) is dominated by large motions, which are the weak point of the filter-based principles (Section 3) that we rely on. Flownet is particularly good at long-range matching thanks to its deep architecture, and this results in vastly superior performance. As stated above, the refinement is of little use with large motions, and brings minimal improvement to either method on Sintel. In conclusion, the different design choices in Flownet and our approach seem complimentary in different regimes. It would be interesting to investigate how to combine their strengths.

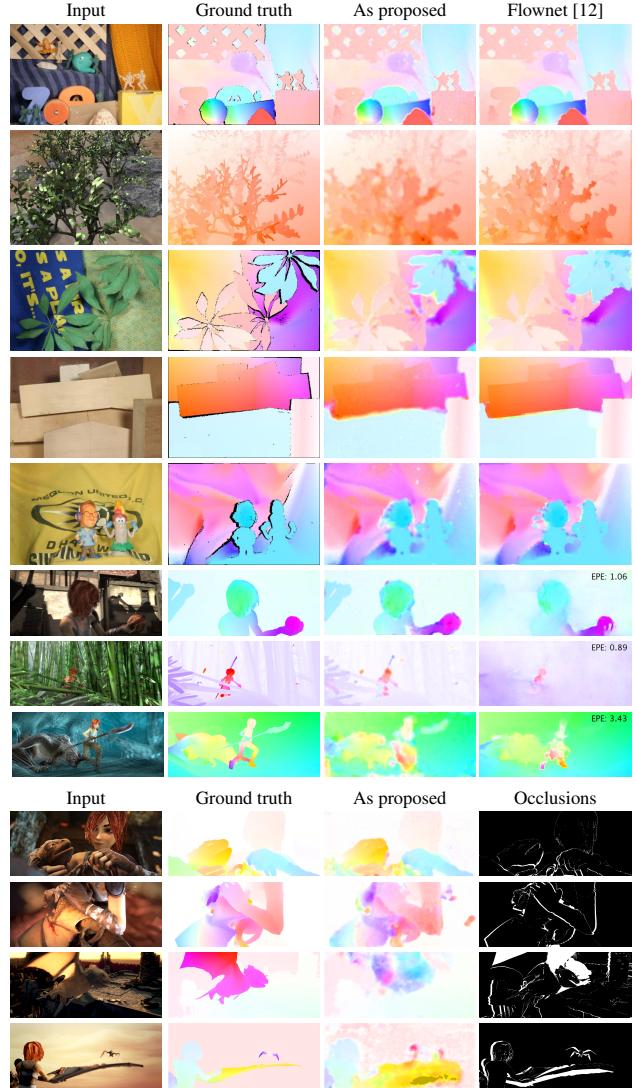


Figure 5. Estimated flow on sequences from the Middlebury (top five) and Sintel (others) datasets. Most failure cases (*e.g.* bottom two) occur near (dis)occlusions, which are common on the Sintel dataset due to large motions of small objects and the small relative camera field-of-view.

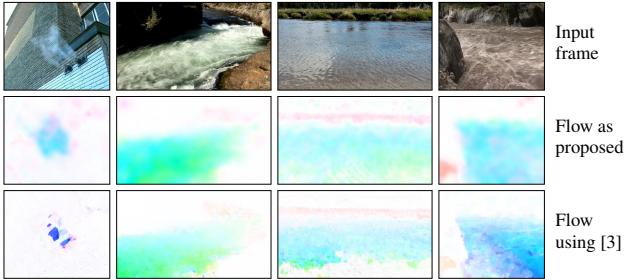


Figure 6. The extraction of optical flow on dynamic textures is challenging for traditional methods, as transparencies (*e.g.* with steam, left) or flicker (*e.g.* on water ripples, middle right) violate the typical assumption of brightness constancy. The core of our approach relies on the analysis of the frequency contents of the video, and produces more stable and reliable motion estimates.

### 7.3. Applicability to transparent motions and dynamic textures

We tested the applicability of our method on scenes that are challenging (dynamic textures) or impossible (transparencies) to handle with traditional optical flow methods. There are no established benchmarks related to motion estimation and dynamic textures. Recent works [8, 7, 35, 36] that highlighted the potential of filter-based motion features in such situation focused on applications such as segmentation [35, 36] or scene recognition [8, 7]. In Fig. 7, we show scenes containing dynamic textures (water, steam) from which we identified the dominant motion. The flow estimated by a typical method [3] is typically noisy and/or inaccurate, as the usual assumption of brightness constancy does not hold (*e.g.* flickering effect on the water surface, changes of brightness/transparency of the steam, etc.). Although no ground truth is available for these scenes, the flow estimated by our methods is more reliable in comparison. We then demonstrate the ability of our distributed representation to capture multiple motions at a single location (transparencies and semi-transparencies), thus going beyond the optical flow representation of pixelwise displacements. We show features in Fig. 7 from three sequences. The first depicts two alpha-blended (in equal proportions) textures moving in opposite directions, thus simulating transparency. The other two depict persons moving behind a fence in directions different than the fence itself [7]. Feature vectors from different locations in the image are visualized as radial bins (orientations) of concentric rings (speeds). Larger values (brighter bins) indicate motion evidence. As expected, areas with simple translations produce one major peak, whereas areas with transparencies produce correspondingly more complex, multimodal distributions<sup>2</sup>.

<sup>2</sup>These experiments used a model trained on the Middlebury dataset.

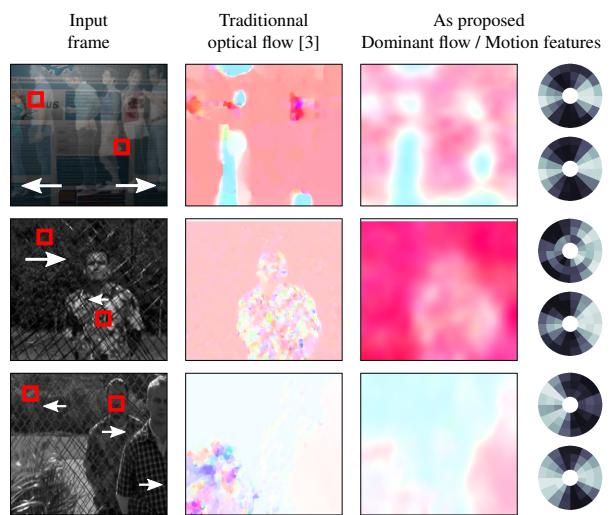


Figure 7. In scenes with multiple, transparent motions, traditional optical flow methods fail and typically produce incoherent results. Our method identifies a more stable dominant motion. More importantly, our higher-dimensional motion descriptor can capture multiple motions at single locations (red squares; white arrows indicate approximate ground truth direction of motion; see text for details).

## 8. Conclusions and future work

We showed how to identify optical flow entirely within a convolutional neural network. By reasoning about required invariances and by using signal processing principles, we designed a simple architecture that can be trained end-to-end, from pixels to dense flow fields. We also showed how to enforce strict rotation invariance by constraining the weights, thus reducing the number of parameters and enabling training on small datasets. The resulting network performs on the Middlebury benchmark with performance comparable to classical methods, but inferior to the best engineered methods.

We believe the approach presented here bears two major advantages over existing optical flow algorithms. First, building upon the classical motion energy model, our approach is able to produce high-dimensional features that can capture non-rigid, transparent, or superimposed motions, which traditional optical flow cannot represent. Second, it constitutes a method for motion estimation formulated entirely as a shallow, easily-trainable CNN, without requiring any post-processing. Its potential is to be used as a building block in deeper architectures (*e.g.* for activity or object recognition in videos) offering the possibility for fine-tuning the representation of motion. The potential of these two aspects will deserve further exploration and should be addressed in future work.

## References

- [1] E. H. Adelson and J. Bergen. Spatiotemporal energy models for the perception of motion. *J. Opt. Soc. Am. A.*, 2, 284–299, 1985.
- [2] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and S. R. A database and evaluation methodology for optical flow. In *International Conference on Computer Vision (ICCV)*, 2007.
- [3] T. Brox, A. Bruhn, N. Papenberg, and W. J. High accuracy optical flow estimation based on a theory for warping. In *European Conference on Computer Vision (ECCV)*, 2004.
- [4] T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 33(3):500–513, 2011.
- [5] T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2011.
- [6] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision (ECCV)*, 2012.
- [7] K. G. Derpanis and R. P. Wildes. The structure of multiplicative motions in natural imagery. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(7):1310–1316, 2010.
- [8] K. G. Derpanis and R. P. Wildes. Spacetime texture representation and recognition based on a spatiotemporal orientation analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34(6):1193–1205, 2012.
- [9] S. Dieleman, K. W. Willett, and J. Dambre. Rotation-invariant convolutional neural networks for galaxy morphology prediction. *CoRR*, abs/1503.07077, 2015.
- [10] M. E. and P. P. A multigrid approach for hierarchical motion estimation. In *IEEE International Conference on Computer Vision (ICCV)*, 1998.
- [11] B. Fasel and D. Gatica-Perez. Rotation-invariant neoperceptron. In *International Conference on Pattern Recognition (ICPR)*, 2006.
- [12] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. *CoRR*, abs/1504.06852, 2015.
- [13] D. Fleet and A. Jepson. Computation of component image velocity from local phase information. *International Journal of Computer Vision (IJCV)*, 5(1):77–104, 1990.
- [14] D. Fortun, P. Bouhoumy, and C. Kervrann. Optical flow modeling and computation: a survey. *Computer Vision and Image Understanding (CVIU)*, 393, 2015.
- [15] D. J. Heeger. Model for the extraction of image flow. *J. Opt. Soc. Am. A*, 1987.
- [16] B. Horn and S. B. Determining optical flow. *Artificial Intelligence*, 11, 185–203, 1981.
- [17] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.
- [18] D. Jayaraman and K. Grauman. Learning image representations equivariant to ego-motion. *CoRR*, abs/1505.02206, 2015.
- [19] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [20] K. R. Konda and R. Memisevic. Unsupervised learning of depth and motion. *CoRR*, abs/1312.3429, 2013.
- [21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.
- [22] D. Laptev and J. M. Buhmann. Transformation-invariant convolutional jungles. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [23] Q. V. Le, J. Ngiam, Z. Chen, D. Chia, P. W. Koh, and A. Y. Ng. Tiled convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.
- [24] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.
- [25] Y. LeCun, L. Bottou, G. Orr, and K. Muller. Efficient backprop. In G. Orr and M. K., editors, *Neural Networks: Tricks of the trade*. Springer, 1998.
- [26] S. A. Niyogi. Fitting models to distributed representations of vision. In *International Joint Conference on Artificial Intelligence*, pages 3–9, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.
- [27] B. Olshausen. Learning sparse, overcomplete representations of time-varying natural images. In *ICIP*, volume 1, pages I–41–4 vol.1, 2003.
- [28] H. Rowley, S. Baluja, and T. Kanade. Rotation invariant neural network-based face detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1998.
- [29] N. C. Rust, V. Mante, E. P. Simoncelli, and J. A. Movshon. How mt cells analyze the motion of visual patterns. *Nature Neuroscience*, 9, 2006.
- [30] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. *CoRR (NIPS Spotlight Session)*, abs/1406.2199, 2014.
- [31] F. Solari, M. Chessa, N. Medathati, and P. Kornprobst. What can we expect from a V1-MT feedforward architecture for optical flow estimation? *Signal Processing: Image Communication*, 2015.
- [32] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2432–2439. IEEE, June 2010.
- [33] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler. Convolutional learning of spatio-temporal features. In *European Conference on Computer Vision (ECCV)*, 2010.
- [34] D. Teney. Personal website. <http://damienteney.info/cnnFlow.htm>.

- [35] D. Teney and M. Brown. Segmentation of dynamic scenes with distributions of spatiotemporally oriented energies. In *British Machine Vision Conference (BMVC)*, 9 2014.
- [36] D. Teney, M. Brown, D. Kit, and P. Hall. Learning similarity metrics for dynamic scene segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [37] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. C3D: generic features for video analysis. *CoRR*, abs/1412.0767, 2014.
- [38] V. Ulman. Improving accuracy of optical flow of heegers original method on biomedical images. In *Image Analysis and Recognition*, volume 6111 of *Lecture Notes in Computer Science*, pages 263–273. Springer Berlin Heidelberg, 2010.
- [39] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. DeepFlow: Large displacement optical flow with deep matching. In *International Conference on Computer Vision (ICCV)*, 2013.
- [40] Z. Wu, X. Wang, Y.-G. Jiang, H. Ye, and X. Xue. Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. In *ACM Multimedia Conference*, 2015.
- [41] H. Ye, Z. Wu, R.-W. Zhao, X. Wang, Y.-G. Jiang, and X. Xue. Evaluating two-stream CNN for video classification. In *ACM on International Conference on Multimedia Retrieval (ICMR)*, 2015.