Optical Flow Estimation using a Spatial Pyramid Network

Anurag Ranjan   Michael J. Black

Max Planck Institute for Intelligent Systems, Tübingen, Germany

{anurag.ranjan, black}@tuebingen.mpg.de

Abstract

We learn to compute optical flow by combining a classical spatial-pyramid formulation with deep learning. This estimates large motions in a coarse-to-fine approach by warping one image of a pair at each pyramid level by the current flow estimate and computing an update to the flow. Instead of the standard minimization of an objective function at each pyramid level, we train one deep network per level to compute the flow update. Unlike the recent FlowNet approach, the networks do not need to deal with large motions; these are dealt with by the pyramid. This has several advantages. First, our Spatial Pyramid Network (SPyNet) is much simpler and 96% smaller than FlowNet in terms of model parameters. This makes it more efficient and appropriate for embedded applications. Second, since the flow at each pyramid level is small ($<$ 1 pixel), a convolutional approach applied to pairs of warped images is appropriate. Third, unlike FlowNet, the learned convolution filters appear similar to classical spatio-temporal filters, giving insight into the method and how to improve it. Our results are more accurate than FlowNet on most standard benchmarks, suggesting a new direction of combining classical flow methods with deep learning.

1. Introduction

Recent years have seen significant progress on the problem of accurately estimating optical flow, as evidenced by improving performance on increasingly challenging benchmarks. Despite this, most flow methods are derived from a "classical formulation" that makes a variety of assumptions about the image, from brightness constancy to spatial smoothness. These assumptions are only coarse approximations to reality and this likely limits performance. The recent history of the field has focused on improving these assumptions or making them more robust to violations [7]. This has led to steady but incremental progress.

An alternative approach abandons the classical formulation altogether and starts over using recent neural network architectures [17, 42]. Such an approach takes a pair (or sequence) of images and learns to directly compute flow from them. Ideally such a network would learn to solve the correspondence problem (short and long range), learn filters relevant to the problem, learn what is constant in the sequence, and learn about the spatial structure of the flow and how it relates to the image structure. The first attempts are promising but are not yet as accurate as the classical methods.

We argue that there is an alternative approach that combines the best of both approaches. Decades of research on flow has produced well engineered systems and principles that are effective. But there are places where these methods make assumptions that limit their performance. Consequently, here we apply machine learning to address the weak points, while keeping the engineered architecture, with the goal of 1) improving performance over existing neural networks and the classical methods upon which our work is based; 2) achieving real-time flow estimates with accuracy better than the much slower classical methods; and 3) reducing memory requirements to make flow more practical for embedded, robotic, and mobile applications.

Computing flow requires the solution to two problems. One is to solve for long-range correlations while the other is to solve for detailed sub-pixel optical flow and precise motion boundaries. The previous neural network method, FlowNet [17], attempts to learn both of these at once. In contrast, we tackle the latter using deep learning and rely on existing methods to solve the former.

To deal with large motions we adopt a traditional coarse-to-fine approach [20] using a spatial pyramid[1]. At that top level of the pyramid, the assumption is that the motions between frames are smaller than a few pixels and that, consequently, the convolutional filters can learn meaningful temporal structure. At each level of the pyramid we solve for the flow using a convolutional network and up-sample the flow to the next pyramid level. As is standard, with classical formulations [38], we warp one image towards the other using the current flow, and repeat this process at each pyramid level. Instead of minimizing a classical objective function at each level, we learn a convolutional network to predict the *flow increment* at that level. We train the network from

---

[1]This, of course, has well-known limitations, which we discuss later.

coarse to fine to learn the flow correction at each level and add this to the flow output of the network above. The idea is that the displacements are then always less than one (or a few) pixels at each pyramid level.

We call the method *SPyNet*, for Spatial Pyramid Network, and train it using the same Flying Chairs data as FlowNet [17]. We report similar performance as FlowNet on Flying Chairs and Sintel [11] but are significantly more accurate than FlowNet on Middlebury [4] and KITTI [19] after fine tuning. The total size of SPyNet is 96% smaller than FlowNet, meaning that it runs faster, and uses much less memory. The expensive iterative propagation of classical methods is replaced by the non-iterative computation of the neural network.

We do not claim to solve the full optical flow problem with SPyNet; we address the same problem as traditional approaches and inherit some of their limitations. For example, it is well known that large motions of small or thin objects are difficult to capture with a pyramid representation. We see the large-motion problem as separate, requiring different solutions. Rather, what we show is that the traditional problem can be reformulated, portions of it can be learned, and performance improves in many scenarios.

Additionally, because our approach connects past methods with new tools, it provides insights into how to move forward. In particular, we find that SPyNet learns spatio-temporal convolutional filters that resemble traditional spatio-temporal derivative or Gabor filters [2, 24]. The learned filters resemble biological models of motion processing filters in cortical areas, MT and V1 [36]. The MT-V1 filters have also been shown useful for optical flow estimation [37, 14]. This is in contrast to the random-looking filters learned by FlowNet (see Supplemental Material). This suggests that it is time to reexamine older spatio-temporal filtering approaches with new tools.

In summary our contributions are: 1) the combination of traditional coarse-to-fine pyramid methods with deep learning for optical flow estimation; 2) a new SPyNet model that is 96% smaller and faster than FlowNet; 3) SPyNet achieves comparable or lower error than FlowNet on standard benchmarks – Sintel, KITTI and Middlebury; 4) the learned spatio-temporal filters provide insight about what filters are needed for flow estimation; 5) the trained network and related code are publicly available for research [2].

2. Related Work

Our formulation effectively combines ideas from "classical" optical flow and recent deep learning methods. Our review focuses on the work most relevant to this.

Spatial pyramids and optical flow. The classical formulation of the optical flow problem involves optimizing

the sum of a data term based on brightness constancy and a spatial smoothness term [25]. Such methods suffer from the fact that they make assumptions about the image brightness change and the spatial structure of the flow that do not match reality. Many methods focus on improving robustness by changing the assumptions; for a review, see [38]. The key advantage of learning to compute flow is that we do not hand craft these assumptions. Rather, the variation in image brightness and spatial smoothness are embodied in the learned network.

The use of spatial pyramids for flow estimation has a long history [10, 20]. Typically Gaussian or Laplacian pyramids are used to deal with large motions. These methods are well known to have problems when small objects move quickly. Brox et al. [8] incorporate long range matching into the traditional optical flow objective function. This approach of combining image matching to capture large motions, with a variational [32] or discrete optimization [21] for fine motions, can produce accurate results.

Of course spatial pyramids are widely used in other areas of computer vision and have recently been used in deep neural networks [16] to learn generative image models.

Spatio-temporal filters. Burt and Adelson [2] lay out the theory of spatio-temporal models for motion estimation and Heeger [24] provides a computational embodiment. While inspired by human perception, such methods did not perform well at the time [6].

Various methods have shown that spatio-temporal filters emerge from learning, for example using independent component analysis [43], sparseness [31], and multi-layer models [12]. Memisevic and Hinton learn simple spatial transformations with a restricted Boltzmann machine [29], finding a variety of filters. Taylor et al. [41] use synthetic data to learn "flow like" features using a restricted Boltzmann machine but do not evaluate flow accuracy.

Dosovitskiy et al. [17] learn spatio-temporal filters for flow estimation using a deep network, yet these filters do not resemble classical filters inspired by neuroscience. By using a pyramid approach, here we learn filters that are visually similar to classical spatio-temporal filters, yet because they are learned from data, produce good flow estimates.

Learning to model and compute flow. Possibly the first attempt to learn a model to estimate optical flow is the work of Freeman et al. [18] using an MRF. They consider a simple synthetic world of uniform moving blobs with ground truth flow. The training data was not realistic and they did not apply the method to real image sequences.

Roth and Black [33] learn a field-of-experts (FoE) model to capture the spatial statistics of optical flow. The FoE can be viewed as a (shallow) convolutional neural network. The model is trained using flow fields generated from laser scans of real scenes and natural camera motions. They have no images of the scenes (only their flow) and consequently the

method only learns the spatial component.

Sun et al. [15] describe the first fully learned model that can be considered a (shallow) convolutional neural network. They formulate a classical flow problem with a data term and a spatial term. The spatial term uses the FoE model from [33], while the data term replaces traditional derivative filters with a set of learned convolutional image filters. With limited training data and a small set of filters, it did not fully show the full promise of learning flow.

Wulff and Black [46] learn the spatial statistics of optical flow by applying robust PCA [22] to real (noisy) optical flow computed from natural movies. While this produces a global flow basis and overly smooth flow, they use the model to compute reasonable flow relatively quickly.

Deep Learning. The above learning methods suffer from limited training data and the use of shallow models. In contrast, deep convolutional neural networks have emerged as a powerful class of models for solving recognition [23, 40] and dense estimation [13, 28] problems.

FlowNet [17] represents the first deep convolutional architecture for flow estimation that is trained end-to-end. The network shows promising results, despite being trained on an artificial dataset of chairs flying over randomly selected images. Despite promising results, the method lags behind the state of the art in terms of accuracy [17]. Deep matching methods [21, 32, 44] do not fully solve the problem, since they resort to classical methods to compute the final flow field. It remains an open question as to which architectures are most appropriate for the problem and how best to train these.

Tran et al. [42], use a traditional flow method to create "semi-truth" training data for a 3D convolutional network. The performance is below the state of the art and the method is not tested on the standard benchmarks. There have also been several attempts at estimating optical flow using unsupervised learning [3, 47]. However these methods have lower accuracy on standard benchmarks.

Fast flow. Several recent methods attempt to balance speed and accuracy, with the goal of real-time processing and reasonable (though not top) accuracy. GPU-flow [45] began this trend but several methods now outperform it. PCA-Flow [46] runs on a CPU, is slower than frame rate, and produces overly smooth flow fields. EPPM [5] achieves similar, middle-of-the-pack, performance on Sintel (test), with similar speed on a GPU. Most recently DIS-Fast [27] is a GPU method that is significantly faster than previous methods but is also significantly less accurate. Our method is also significantly faster than FlowNet, which has a runtime of 80ms/frame (FlowNetS).

Size is also important but has attracted less attention than speed. For optical flow on embedded processors, aerial vehicles, phones, etc., the algorithm needs a small memory footprint. Our network is 96% smaller than FlowNetS, us-

ing only 9.7 MB for the model parameters, making it easily small enough to fit on a mobile phone GPU.

3. Spatial Pyramid Network

Our approach uses the coarse-to-fine spatial pyramid structure of [16] to learn residual flow at each pyramid level. Here we describe the network and training procedure.

3.1. Spatial Sampling

Let I be an $m \times n$ image with dimensions that are powers of 2. Let $d(.)$ be the downsampling function that decimates I to the corresponding image $d(I)$ of size $m/2 \times n/2$. Let $u(.)$ be the reverse operation that upsamples images by a factor of 2. These operators are bilinear and are also used for resampling the the optical flow field, V. We also define a warping operator $w(I, V)$ that warps the image, I according to the flow field, V, using bi-linear interpolation.

3.2. Inference

Let $\{G_0, ..., G_K\}$ denote a set of trained convolutional neural network (convnet) models, each of which computes residual flow, v_k

$$v_k = G_k(I_k^1, w(I_k^2, u(V_{k-1})), u(V_{k-1})) \qquad (1)$$

at the k-th pyramid level. The convnet G_k computes the residual flow v_k using the upsampled flow from the previous pyramid level, V_{k-1}, and the frames $\{I_k^1, I_k^2\}$ at level k. The second frame I_k^2 is warped using the flow as $w(I_k^2, u(V_{k-1}))$ before feeding it to the convnet G_k. The flow, V_k at the k-th pyramid level is then

$$V_k = u(V_{k-1}) + v_k. \qquad (2)$$

As shown in Fig. 1, we start with downsampled images $\{I_0^1, I_0^2\}$ and an initial flow estimate that is zero everywhere to compute the residual flow $v_0 = V_0$ at the top of the pyramid. We upsample the resulting flow, $u(V_0)$, and pass it to the network G_1 along with $\{I_1^1, w(I_1^2, u(V_0))\}$ to compute the residual flow v_1. At each pyramid level, we compute the flow V_k using Equation (2). The flow V_k is similarly propagated to higher resolution layers of the pyramid until we obtain the flow V_K at full resolution. Figure 1 shows the working of our approach using a 3-level pyramid. In practice, we use a 5-level pyramid ($K = 4$).

3.3. Training and Network Architecture

We train each of the convnets $\{G_0, ..., G_K\}$ independently and sequentially to compute the residual flow v_k given the inputs $\{I_k^1, w(I_k^2, u(V_{k-1})), u(V_{k-1})\}$. We compute target residual flows $\hat{v}_k$ as a difference of target flow $\hat{V}_k$ at the k-th pyramid level and the upsampled flow, $u(V_{k-1})$ obtained from the trained convnet of the previous level

$$\hat{v}_k = \hat{V}_k - u(V_{k-1}). \qquad (3)$$
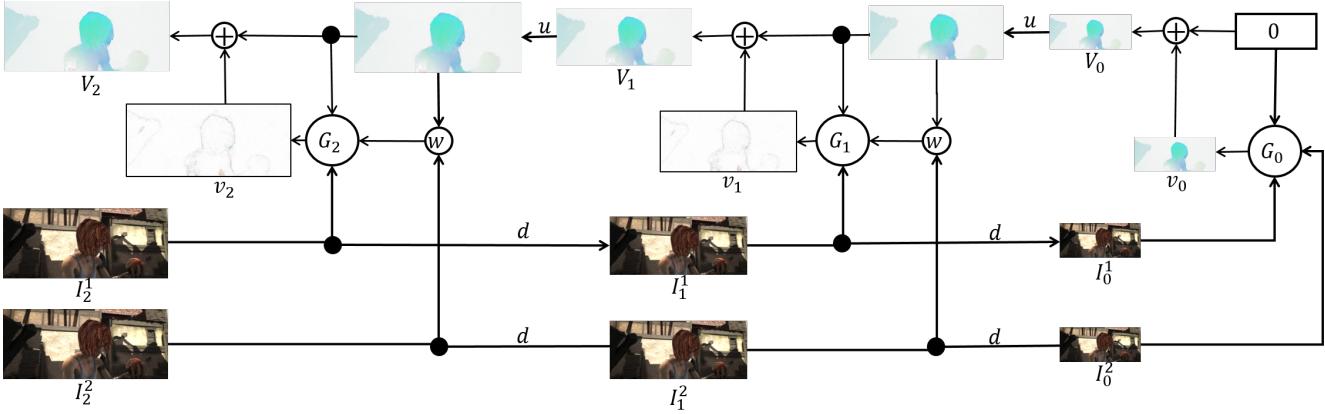
Figure 1. Inference in a 3-Level Pyramid Network [16]: The network G_0 computes the residual flow v_0 at the highest level of the pyramid (smallest image) using the low resolution images $\{I_0^1, I_0^2\}$. At each pyramid level, the network G_k computes a residual flow v_k which propagates to each of the next lower levels of the pyramid in turn, to finally obtain the flow V_2 at the highest resolution.
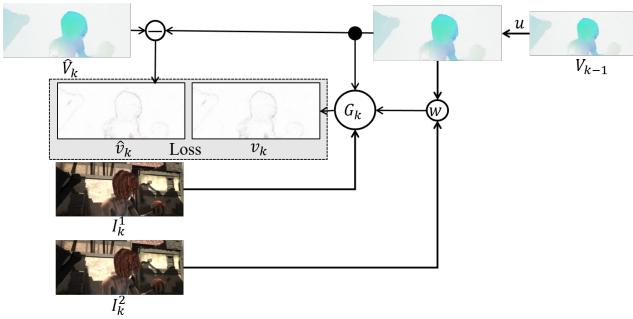

Figure 2. Training network G_k requires trained models $\{G_0...G_{k-1}\}$ to obtain the initial flow $u(V_{k-1})$. We obtain ground truth residual flows $\hat{v}_k$ by subtracting downsampled ground truth flow $\hat{V}_k$ and $u(V_{k-1})$ to train the network G_k using the EPE loss.

As shown in Fig. 2, we train each of the networks, G_k, to minimize the average End Point Error (EPE) loss between the residual flow v_k and $\hat{v}_k$

$$\frac{1}{m_k n_k} \sum_{x,y} \sqrt{(v_k^x - \hat{v}_k^x)^2 + (v_k^y - \hat{v}_k^y)^2}$$

where the $m_k \times n_k$ is the image dimension at level k and the x and y superscripts indicate the horizontal and vertical components of the flow vector.

Each level in the pyramid has a simplified task relative to the full optical flow estimation problem; it only has to estimate a small-motion update to an existing flow field. Consequently each network can be simple. Here, each G_k has 5 convolutional layers; this gave the best combination of accuracy, size, and speed. We train five convnets $\{G_0, ..., G_4\}$ at different resolutions of the Flying Chairs dataset. The network G_0 is trained with 24x32 images. We double the resolution at each pyramid level and finally train the convnet, G_4 with a resolution of 384x512.

Each convolutional layer is followed by a Rectified Linear Unit (ReLU), except the last one. We use 7x7 convolutional kernels for each layer; these worked better than smaller filters. The number of feature maps in each convnet, G_k are $\{32, 64, 32, 16, 2\}$. The image I_k^1 and the warped image $w(I_k^2, u(V_{k-1}))$ have 3 channels each (RGB). The upsampled flow $u(V_{k-1})$ is 2 channel (horizontal and vertical). We stack image frames together with the upsampled flow to form an 8 channel input to each G_k. The output is 2 channel flow corresponding to velocity in x and y.

Using Torch7[3], we train five networks $\{G_0, ..., G_4\}$ such that each network G_k uses the previous network G_{k-1} as initialization. The networks are trained using Adam [26] optimization with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We use a batch size of 32 across all networks with 4000 iterations per epoch. We use a learning rate of 1e-4 for the first 60 epochs and decrease it to 1e-5 until the networks converge. We use We use the Flying Chairs [17] dataset and MPI Sintel [11] for training. We trained G_0 for three days and $\{G_1, G_2, G_3, G_4\}$ for one day each on a single Titan X.

We include various types of data augmentation during training. We randomly scale images by a factor of $[1, 2]$ and apply rotations at random within $[-17°, 17°]$. We then apply a random crop to match the resolution of the convnet, G_k being trained. We include additive white Gaussian noise sampled uniformly from $\mathcal{N}(0, 0.1)$. We apply color jitter with additive brightness, contrast and saturation sampled from a Gaussian, $\mathcal{N}(0, 0.4)$. We finally normalize the images using a mean and standard deviation computed from a large sample of Imagenet [34] data in [23].

4. Experiments

We evaluate our performance on standard optical flow benchmarks and compare with FlowNet [17] and Clas-

---

[3] http://torch.ch/

Method	Sintel Clean		Sintel Final		KITTI		Middlebury		Flying Chairs	Time (s)
	train	test	train	test	train	test	train	test	test	
Classic+NLP	4.13	6.73	5.90	8.29	-	-	0.22	0.32	3.93	102
FlowNetS	4.50	7.42	**5.45**	**8.43**	**8.26**	-	1.09	-	2.71	0.080
FlowNetC	4.31	7.28	5.87	8.81	9.35	-	1.15	-	**2.19**	0.150
SPyNet	**4.12**	**6.69**	5.57	**8.43**	9.12	-	**0.33**	**0.58**	2.63	**0.069**
FlowNetS+ft	3.66	6.96	4.44	**7.76**	7.52	9.1	0.98	-	3.04	0.080
FlowNetC+ft	3.78	6.85	5.28	8.51	8.79	-	0.93		**2.27**	0.150
SPyNet+ft	**3.17**	**6.64**	**4.32**	8.36	8.25	10.1	**0.33**	**0.58**	3.07	**0.069**
SPyNet+ft*	-	-	-	-	**3.36**	**4.1**	-	-	-	-

Table 1. Average end point errors (EPE). Results are divided into methods trained with (+ft) and without fine tuning. SPyNet+ft* uses additional training data compared to FlowNet+ft. Bold font indicates the most accurate results among the convnet methods. All run times are measured on Flying Chairs and exclude image loading time.

Method	Sintel Final						Sintel Clean					
	d_{0-10}	d_{10-60}	d_{60-140}	s_{0-10}	s_{10-40}	s_{40+}	d_{0-10}	d_{10-60}	d_{60-140}	s_{0-10}	s_{10-40}	s_{40+}
FlowNetS+ft	7.25	4.61	**2.99**	1.87	5.83	43.24	5.99	3.56	2.19	1.42	3.81	40.10
FlowNetC+ft	7.19	4.62	3.30	2.30	6.17	**40.78**	5.57	3.18	1.99	1.62	3.97	**33.37**
SpyNet+ft	**6.69**	**4.37**	3.29	**1.39**	**5.53**	49.71	**5.50**	**3.12**	**1.71**	**0.83**	**3.34**	43.44

Table 2. Comparison of FlowNet and SpyNet on the Sintel benchmark for different velocities, s, and distances, d, from motion boundaries.

sic+NLP [38], a traditional pyramid-based method. We compare performance using average end point errors in Table 1. We evaluate on all the standard benchmarks and find that SPyNet is the most accurate overall, with and without fine tuning (details below). Additionally SPyNet is faster than all other methods.

Note that the FlowNet results reported on the MPI-Sintel website are for a version that applies variational refinement ("+v") to the convnet results. Here we are not interested in the variational component and only compare the results of the convnet output.

Flying Chairs. SPyNet achieves better performance than FlowNetS [17] on the Flying Chairs dataset, however FlowNetC [17] performs better than ours. We show the qualitative results on Flying Chairs dataset in Fig. 3 and compare the performance in Table 1.

MPI-Sintel. The resolution of Sintel images is 436x1024. To use SPyNet, we scale the images to 448x1024, and use 6 pyramid levels to compute the optical flow. The networks used on each pyramid level are $\{G_0, G_1, G_2, G_3, G_4, G_4\}$. We repeat the network G_4 at the sixth level of pyramid for experiments on Sintel. Because Sintel has extremely large motions, we found that this gives better performance than using just five levels.

We evaluate the performance of our model on MPI-Sintel [11] in two ways. First, we directly use the model trained on Flying Chairs dataset and evaluate our performance on both

Frame 1   Frame 2   Ground Truth   SPyNet

Figure 3. Visualization of optical flow estimates using our model (SPyNet) and the corresponding ground truth flow fields on the Flying Chairs dataset.

the training and the test sets. Second, we extract a validation set from the Sintel training set, using the same partition as [17]. We fine tune our model independently on the Sintel Clean and Sintel Final split, and evaluate the EPE. The fine-tuned models are listed as "+ft" in Table 1. We show the qualitative results on MPI-Sintel in Fig. 4.

Table 2 compares our fine-tuned model with FlowNet [17] for different velocities and distances from motion boundaries. We observe that SPyNet is more accurate than

| Frames | Ground Truth | FlowNetS | FlowNetC | SPyNet |
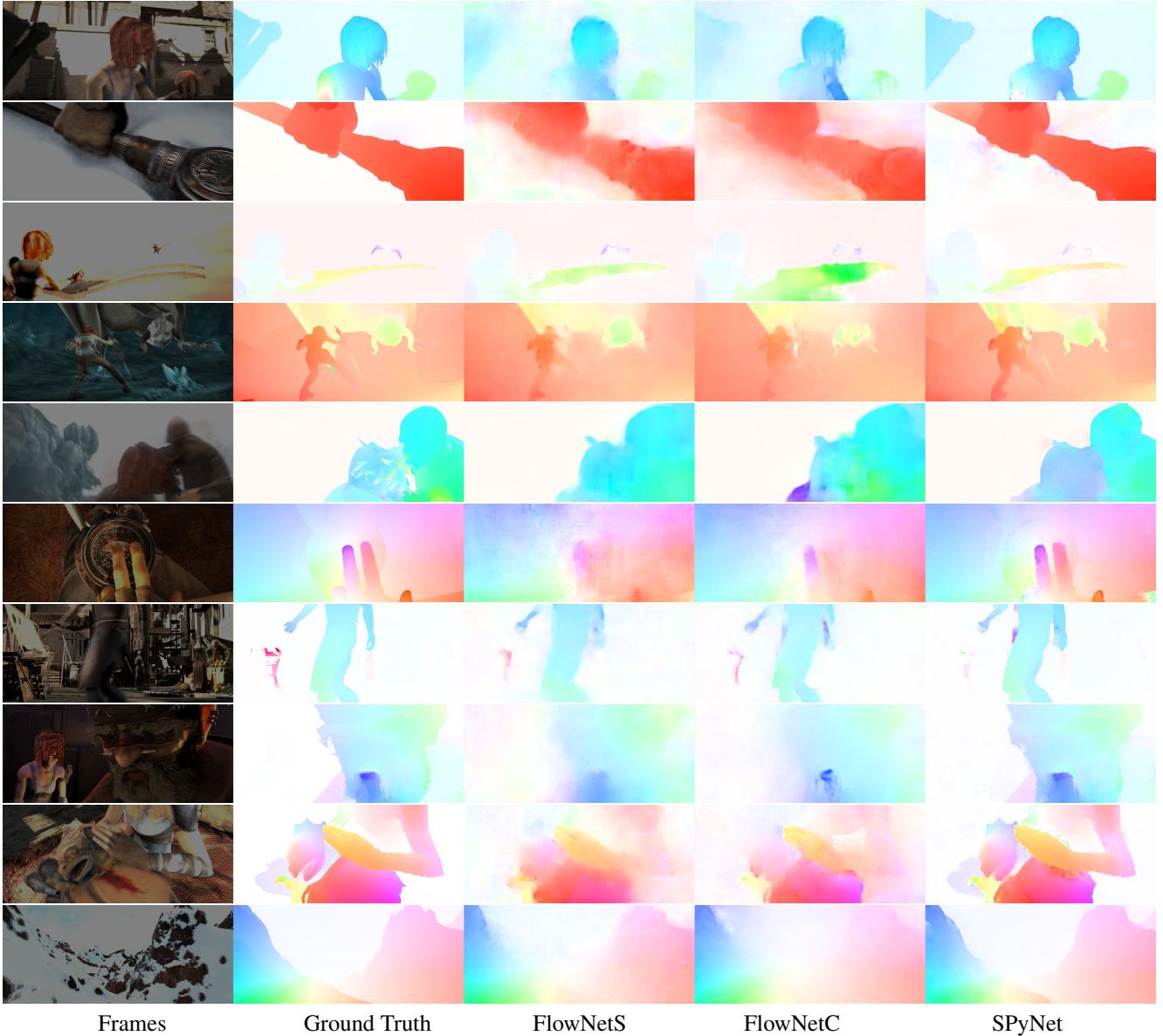
Figure 4. Visual comparison of optical flow estimates using our SPyNet model with FlowNet on the MPI Sintel dataset. The top five rows are from the Sintel Final set and the bottom five rows are from the Sintel Clean set. Each set of 5 rows is sorted in the order of increasing average displacements. SPyNet performs particularly well when the motions are relatively small.

FlowNet for all velocity ranges except the largest displacements (over 40 pixels/frame). SPyNet is also more accurate than FlowNet close to motion boundaries (see Figure 4), which is important for many problems.

KITTI and Middlebury. We evaluate KITTI [19] scenes using the base model SPyNet trained on Flying Chairs. Here we also fine-tune the model using Driving and Monkaa scenes from [30] and evaluate the fine-tuned model SPyNet+ft*. Fine tuning results in a significant improvement in accuracy by about 5 pixels. The large improvement suggests that better datasets are needed and that these could improve the accuracy of SPyNet further on general scenes.

For the Middlebury [4] dataset, we evaluate the sequences using the base model SPyNet as well as SPyNet+ft, which is fine-tuned on the Sintel-Final dataset; the Middlebury dataset itself is too small for fine-tuning. SPyNet is significantly more accurate on Middlebury, where FlowNet has trouble with the small motions. Both learned methods are less accurate than Classic+NL on Middlebury but both are also significantly faster.

Number of Model Parameters

FlowNetS	32,070,472
FlowNetC	32,561,032
SPyNet	1,200,250
G_k	240,050
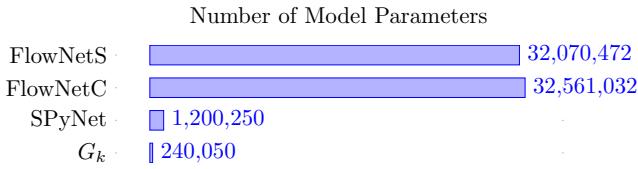
Figure 5. Model size of various methods. Our model is 96% smaller than FlowNet.

5. Analysis

Model Size. Combining spatial pyramids with convnets results in a huge reduction in model complexity. At each pyramid level, a network, G_k, has 240,050 learned parameters. The total number of parameters learned by the entire network is 1,200,250, with 5 spatial pyramid levels. In comparison, FlowNetS and FlowNetC [17] have 32,070,472 and 32,561,032 parameters respectively. SPyNet is about 96 % smaller than FlowNet (Fig. 5).

The spatial pyramid approach enables a significant reduction in model parameters without sacrificing accuracy. There are two reasons – the warping function and learning of residual flow. By using the warping function directly, the convnet does not need to learn it. More importantly, learning residual flows restricts the range of flow fields in the output space. Each network only has to model a smaller range of velocities at each level of the spatial pyramid.

SPyNet also has a small memory footprint. The disk space required to store all the model parameters is 9.7 MB. This simplifies deployment on mobile or embedded devices with GPU support; this is future work.

Visualization of Learned Filters. Figure 6(a) shows examples of filters learned by the first layer of the network, G_2. In each row, the first two columns show the spatial filters that operate on the RGB channels of the two input images respectively. The third column is the difference between the two spatial filters, hence representing the temporal features learned by our model. We observe that most of the spatio-temporal filters in Fig. 6(a) are equally sensitive to all color channels, and hence appear mostly grayscale. Note that the actual filters are 7×7 pixels and are upsampled for visualization.

We observe that many of the spatial filters appear to be similar to traditional Gaussian derivative filters used by classical methods. These classical filters are hand crafted and typically are applied in the horizontal and vertical direction. Here we observe a greater variety of derivative-like filters of varied scales and orientations. We also observe filters that spatially resemble second derivative or Gabor filters [2]. The temporal filters show a clear derivative-like structure in time. Note that these filters are very different
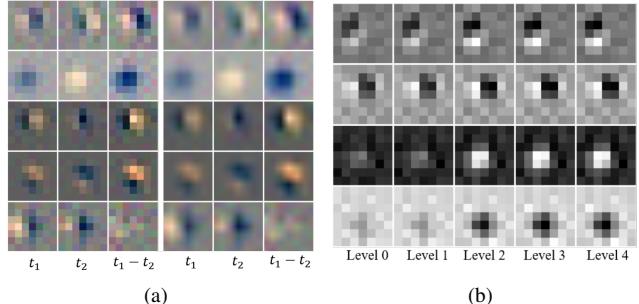
Figure 6. (a) Visualization of filter weights in the first layer of G_2 showing their spatiotemporal nature on RGB image pairs using nearest-neighbor (left) and bilinear (right) interpolation. (b) Evolution of filters across the pyramid levels (from low resolution (0) to high resolution (4))

from those reported in [17] (Sup. Mat.), which have a high-frequency structure, unlike classical filters.

Figure 6(b) illustrates how filters learned by the network at each level of the pyramid differ from each other. Recall that, during training, each network is initialized with the network before it in the pyramid. The filters, however, do not stay exactly the same with training. Most of the filters in our network look like rows 1 and 2, where the filters become sharper with increasing contrast as we progress towards the finer-resolution levels of the pyramid. However, there are some filters that are similar to rows 3 and 4. These filters become more defined at higher resolution levels of the pyramid.

Speed. Optical flow estimation is traditionally viewed as an optimization problem involving some form of variational inference. This is computationally expensive, often taking seconds or minutes per frame. This has limited the application of optical flow in robotics, embedded systems, and video analysis. Using a GPU can speed up traditional methods [39, 45] but with reduced accuracy. Feed forward deep networks [17] leverage fast GPU convolutions and avoid iterative optimization.

Figure 7 shows the speed-accuracy comparisons of several well known methods. All times shown are measured with the images already loaded in the memory. The errors are computed as the average EPE of both the clean and final MPI-Sintel sequences. SPyNet offers a good balance between speed and accuracy; no faster method is as accurate.

6. Discussion and Future Work

Traditional flow methods linearize the brightness constancy equation resulting in an optical flow constraint equation implemented with spatial and temporal derivative filters. Sometimes methods adopt a more generic filter con-
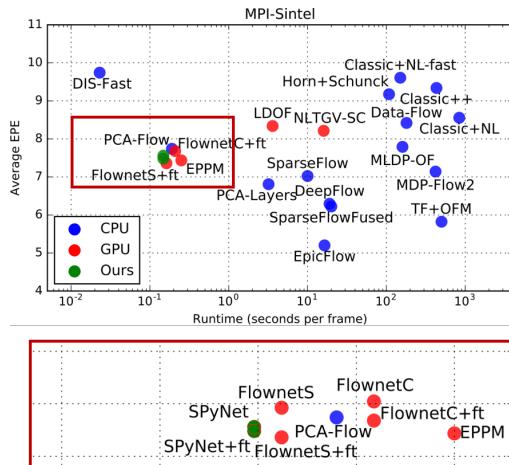
Figure 7. Average EPE vs. runtime on MPI-Sintel. Zoomed in version on the bottom shows the fastest methods. Times were measured by us. Adapted from [46].

stancy assumption [1, 9]. Our filters are somewhat different. The filters learned by SPyNet are used in the direct computation of the flow by the feed-forward network.

SPyNet is small compared with other recent optical flow networks. Examination of the filters, however, suggests that it might be possible to make it significantly smaller still. Many of the filters resemble derivative of Gaussian filters or Gabor filters at various scales, orientations, spatial frequencies, and spatial shifts. Given this, it may be possible to significantly compress the filter bank by using dimensionality reduction or by using a set of analytic spatio-temporal features. Some of the filters may also be separable.

Early methods for optical flow used analytic spatio-temporal features but, at the time, did not produce good results and the general line of spatio-temporal filtering decayed. The difference from early work is that our approach suggests the need for a large filter bank of varied filters. Note also that these approaches considered only the first convolutional layer of filters and did not seek a "deep" solution. This all suggests the possibility that a deep network of analytic filters could perform well. This could vastly reduce the size of the network and the number of parameters that need to be learned.

We observe that first layer filters of FlowNet [17] appear more random than the Gabor-like filters common to most neural networks or previous spatio-temporal filters (see Supplementary Material). We suspect this is because first layer windows do not overlap common image patches when the motion exceeds a few pixels. In contrast, our pyramid structure means that the motions are always small and the convolutional windows are always looking at related patches in two neighboring frames. We find that this leads to filters, even in the first layer, that resemble classical spatio-temporal filters.

Note that pyramids have well-known limitations for dealing with large motions [8, 35]. In particular, small or thin objects that move quickly effectively disappear at coarse pyramid levels, making it impossible to capture their motion. Recent approaches for dealing with such large motions use sparse matching to augment standard pyramids [8, 44]. Future work should explore adding long-range matches to SPyNet. Alternatively Sevilla et al. [35] define a channel constancy representation that preserves fine structures in a pyramid. The channels effectively correspond to filters that could be learned.

A spatial pyramid can be thought of as the simple application of a set of linear filters. Here we take a standard spatial pyramid but one could learn the filters for the pyramid itself. SPyNet also uses a standard warping function to align images using the flow computed from the previous pyramid level. This too could be learned.

An appealing feature of SPyNet is that it is small enough to fit on a mobile device. Future work will explore a mobile implementation and its applications. Additionally we will explore extending the method to use more frames (e.g. 3 or 4). Multiple frames could enable the network to reason more effectively about occlusion.

Finally, Flying Chairs is not representative of natural scene motions. We are exploring new training datasets to improve performance on common sequences.

7. Conclusions

In summary, we have described a new optical flow method that combines features of classical optical flow algorithms with deep learning. In a sense, there are two notions of "deepness" here. First we use a "deep" spatial pyramid to deal with large motions. Second we use deep neural networks at each level of the spatial pyramid and train them to estimate a flow *update* at each level. This approach means that each network has less work to do than a fully generic flow method that has to estimate arbitrarily large motions. At each pyramid level we assume that the motion is small (on the order of a pixel). This is borne out by the fact that the network learns spatial and temporal filters that resemble classical derivatives of Gaussians. Because each sub-task is so much simpler, our network needs many fewer parameters than previous methods like FlowNet. This results in a method with a small memory footprint that is faster than existing methods. At the same time, SPyNet achieves an accuracy comparable to FlowNet, surpassing it in several benchmarks. This opens up the promise of optical flow that is accurate, practical, and widely deployable.

Acknowledgement

References

[1] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden. Pyramid methods in image processing. *RCA engineer*, 29(6):33–41, 1984.

[2] E. H. Adelson and J. R. Bergen. Spatiotemporal energy models for the perception of motion. *J. Opt. Soc. Am. A*, 2(2):284–299, Feb. 1985.

[3] A. Ahmadi and I. Patras. Unsupervised convolutional neural networks for motion estimation. *arXiv preprint arXiv:1601.06087*, 2016.

[4] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.

[5] L. Bao, Q. Yang, and H. Jin. Fast edge-preserving Patch-Match for large displacement optical flow. *Image Processing, IEEE Transactions on*, 23(12):4996–5006, Dec 2014.

[6] J. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *Int. J. Comp. Vis. (IJCV)*, 12(1):43–77, 1994.

[7] M. J. Black and P. Anandan. A framework for the robust estimation of optical flow. In *Computer Vision, 1993. Proceedings., Fourth International Conference on*, pages 231–236. IEEE, 1993.

[8] T. Brox, C. Bregler, and J. Malik. Large displacement optical flow. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 41–48. IEEE, 2009.

[9] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *European conference on computer vision*, pages 25–36. Springer, 2004.

[10] P. J. Burt and E. H. Adelson. The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, COM-34(4):532–540, 1983.

[11] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, Oct. 2012.

[12] C. Cadieu and B. A. Olshausen. Learning transformational invariants from natural movies. In *Advances in neural information processing systems*, pages 209–216, 2008.

[13] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[14] M. Chessa, N. K. Medathati, G. S. Masson, F. Solari, and P. Kornprobst. Decoding mt motion response for optical flow estimation: An experimental evaluation. In *Signal Processing Conference (EUSIPCO), 2015 23rd European*, pages 2241–2245. IEEE, 2015.

[15] S. D., S. Roth, J. Lewis, and M. Black. Learning optical flow. In *ECCV*, pages 83–97, 2008.

[16] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015.

[17] A. Dosovitskiy, P. Fischery, E. Ilg, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, T. Brox, et al. Flownet: Learning optical flow with convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2758–2766. IEEE, 2015.

[18] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael. Learning low-level vision. *International Journal of Computer Vision*, 40(1):25–47, 2000.

[19] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[20] F. Glazer. *Hierarchical motion detection*. PhD thesis, University of Massachusetts, Amherst, MA, 1987. COINS TR 87–02.

[21] F. Güney and A. Geiger. Deep discrete flow. In *Asian Conference on Computer Vision (ACCV)*, 2016.

[22] S. Hauberg, A. Feragen, R. Enficiaud, and M. Black. Scalable robust principal component analysis using Grassmann averages. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, Dec. 2015.

[23] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[24] D. J. Heeger. Model for the extraction of image flow. *J. Opt. Soc. Am*, 4(8):1455–1471, Aug. 1987.

[25] B. K. Horn and B. G. Schunck. Determining optical flow. *1981 Technical Symposium East*, pages 319–331, 1981.

[26] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[27] T. Kroeger, R. Timofte, D. Dai, and L. V. Gool. Fast optical flow using dense inverse search. In *Computer Vision – ECCV*, 2016.

[28] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[29] R. Memisevic and G. E. Hinton. Learning to represent spatial transformations with factored higher-order boltzmann machines. *Neural Computation*, 22(6):1473–1492, 2010.

[30] N.Mayer, E.Ilg, P.Häusser, P.Fischer, D.Cremers, A.Dosovitskiy, and T.Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. arXiv:1512.02134.

[31] B. A. Olshausen. Learning sparse, overcomplete representations of time-varying natural images. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 1, pages I–41. IEEE, 2003.

[32] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow. In *Computer Vision and Pattern Recognition*, 2015.

[33] S. Roth and M. J. Black. Fields of experts. *International Journal of Computer Vision*, 82(2):205–229, 2009.

[34] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[35] L. Sevilla-Lara, D. Sun, E. G. Learned-Miller, and M. J. Black. Optical flow estimation with channel constancy. In *Computer Vision – ECCV 2014*, volume 8689 of *Lecture Notes in Computer Science*, pages 423–438. Springer International Publishing, Sept. 2014.

[36] E. P. Simoncelli and D. J. Heeger. A model of neuronal responses in visual area MT. *Vision Res.*, 38(5):743–761, 1998.

[37] F. Solari, M. Chessa, N. K. Medathati, and P. Kornprobst. What can we expect from a v1-mt feedforward architecture for optical flow estimation? *Signal Processing: Image Communication*, 39:342–354, 2015.

[38] D. Sun, S. Roth, and M. J. Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision*, 106(2):115–137, 2014.

[39] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *European conference on computer vision*, pages 438–451. Springer, 2010.

[40] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[41] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler. Convolutional learning of spatio-temporal features. In *European conference on computer vision*, pages 140–153. Springer, 2010.

[42] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Deep End2End Voxel2Voxel prediction. In *The 3rd Workshop on Deep Learning in Computer Vision*, 2016.

[43] L. van Hateren and J. Ruderman. Independent component analysis of natural image sequences yields spatio-temporal filters similar to simple cells in primary visual cortex. *Proceedings: Biological Sciences*, 265(1412):23152320, 1998.

[44] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1385–1392, 2013.

[45] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof. Anisotropic Huber-L1 optical flow. In *BMVC*, London, UK, September 2009.

[46] J. Wulff and M. J. Black. Efficient sparse-to-dense optical flow estimation using a learned basis and layers. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 120–130. IEEE, 2015.

[47] J. J. Yu, A. W. Harley, and K. G. Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. *arXiv preprint arXiv:1608.05842*, 2016.