Deep Discrete Flow

Fatma Güney[1] and Andreas Geiger[1,2]

[1]Autonomous Vision Group, MPI for Intelligent Systems, Tübingen
[2]Computer Vision and Geometry Group, ETH Zürich

Abstract. Motivated by the success of deep learning techniques in matching problems, we present a method for learning context-aware features for solving optical flow using discrete optimization. Towards this goal, we present an efficient way of training a context network with a large receptive field size on top of a local network using dilated convolutions on patches. We perform feature matching by comparing each pixel in the reference image to every pixel in the target image, utilizing fast GPU matrix multiplication. The matching cost volume from the network's output forms the data term for discrete MAP inference in a pairwise Markov random field. We provide an extensive empirical investigation of network architectures and model parameters. At the time of submission, our method ranks second on the challenging MPI Sintel test set.

1 Introduction

Despite large progress, optical flow is still an unsolved problem in computer vision. Challenges provided by autonomous driving applications [1–3] or current benchmarks like KITTI [4, 5] and MPI Sintel [6] include large motions, appearance changes, as well as uniform image regions. While the predominant paradigm for estimating optical flow is based on continuous optimization [7–9] with coarse-to-fine warping [10], recent approaches leverage discrete optimization strategies [11–14] in order to overcome local minima and to gain robustness.

While these approaches have shown promising results, their performance still falls considerably behind the state-of-the-art in stereo matching [15, 16]. While 2D flow estimation is an inherently more difficult problem than 1D matching along the epipolar line, most existing works on discrete optical flow optimization exploit hand-crafted features for calculating the matching costs. In contrast, the most successful approaches in stereo matching exploit a combination of learned local feature representations and global discrete optimization [16–18].

In this paper, we investigate the utility of feature learning for discrete optical flow, see Fig. 1 for an illustration. In particular, we modify the "DiscreteFlow" framework of Menze et al. [11] by replacing their hand-crafted descriptors with learned feature representations. In particular, we investigate two types of networks: a local network with a small receptive field consisting of 3x3 convolutions followed by non-linearities as well as a subsequent context network that aggregates information over larger image regions using dilated convolutions [19]. As naïve *patch-based* training with dilated convolutions is computationally very

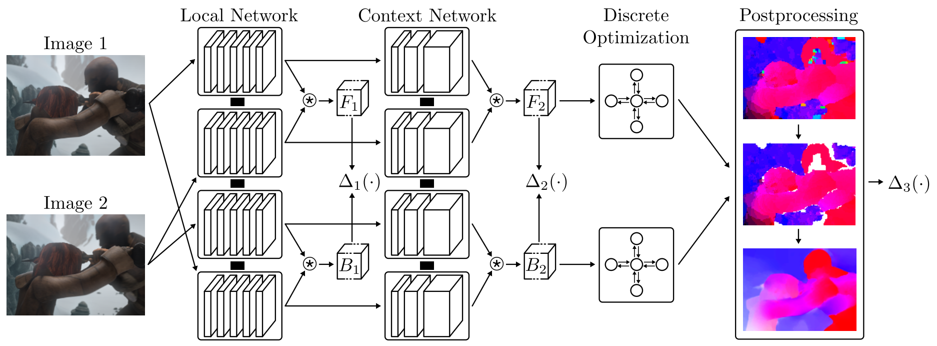

Fig. 1: **Deep Discrete Flow.** The input images are processed in forward order (top stream) and backward order (bottom stream) using local and context Siamese convolutional neural networks, yielding per-pixel descriptors. We then match points on a regular grid in the reference image to every pixel in the other image, yielding a large tensor of forward matching costs (F_1/F_2) and backward matching costs (B_1/B_2). Matching costs are smoothed using discrete MAP inference in a pairwise Markov random field. Finally, a forward-backward consistency check removes outliers and sub-pixel accuracy is attained using the EpicFlow interpolator [20]. We train the model in a piece-wise fashion via the loss functions.

expensive, we propose an efficient implementation based on regular strided convolutions. For efficient learning of the whole pipeline, we specify auxiliary loss functions akin to [16] and train the model in a piece-wise fashion.

We provide a detailed empirical analysis of the impact of each of the components of the pipeline. More specifically, we compare a large number of different local and context architectures with respect to each other and to traditional hand-crafted features. Further, we compare the results of the best-performing systems after discrete optimization and sub-pixel interpolation, and qualitatively visualize the results with their corresponding error images at every stage.

2 Related Work

In this section, we survey the most important related works. We first provide an overview of related optical flow approaches with a particular focus on recent discrete and mixed discrete/continuous approaches that attain state-of-the-art performance on current benchmarks. In the second part, we review current feature learning approaches for correspondence estimation.

Optical Flow: The classical formulation for estimating optical flow [7, 9] involves solving a continuous variational optimization problem. To cope with displacements larger than a few pixels, coarse-to-fine estimation heuristics are commonly employed [10, 21–27]. Unfortunately, coarse-to-fine schemes often lead to blurred object boundaries on current benchmarks [4, 6, 28] due to their susceptibility to local minima in the energy function.

Thus, discrete formulations have recently gained popularity. One line of work incorporates pre-estimated sparse feature correspondences into the optimization process [20,29–32]. To allow for more robust estimates, a second line of work directly formulates optical flow estimation as a discrete optimization problem [33], e.g., in terms of MAP inference in a Markov random field under appropriate flow priors. These approaches can be further categorized into epipolar constrained methods [34, 35], methods which estimate the most likely flow field based on a small set of dense flow field proposals [13,36–38] and methods that estimate flow directly at the pixel level [11, 12, 14, 39, 40].

More specifically, Menze et al. [11] establish a sparse set of 500 flow proposals by matching Daisy descriptors [41] using fast approximate nearest neighbor techniques. Exploiting the truncation property of their pairwise potentials, they efficiently approximate the MAP solution using belief propagation. Chen et al. [12] extend the efficient min-convolution algorithm to 2D flow fields and optimize a discretized version of the classical variational objective using normalized cross-correlation as data term. For sub-pixel accuracy and to extrapolate into occluded regions, both approaches exploit an additional extrapolation and variational post-processing step [20].

While all aforementioned methods focus on the optimization of an energy function based on hand-crafted local feature descriptors, in this work, we investigate the benefits of learning feature representations for discrete optical flow estimation. In particular, we leverage the framework of [11] and replace their features using non-local pixel representations trained for predicting optical flow.

Feature Learning for Correspondence Estimation: Motivated by the success of deep learning in image classification and object recognition [42], a number of papers have tackled the problem of correspondence estimation by learning deep convolutional representations.

Recently, Fischer, Mayer et al. [43, 44] have demonstrated dense end-to-end flow prediction using a deep convolutional neural network which takes as input two images and directly outputs a flow map. While impressive performance has been demonstrated, the method does not attain state-of-the-art performance on current leaderboards. One difficulty is the model's high capacity and the associated large amount of data required to train it.

An alternative approach, which we follow in this paper, is to learn per-pixel feature representations using Siamese networks which can be fed into a winner-takes-all selection scheme or - as in our case - into a discrete optimization algorithm. While the learned representations tend to be more local, they are also less prone to overfitting. Importantly, even small datasets such as KITTI [4] or MPI Sintel [6] provide millions of training points as every pixel provides a training example. This is in contrast to dense approaches [43] where hundreds of thousands images with ground truth flow maps are required for obtaining reliable representations.

A number of approaches [45, 46] aim for descriptor learning for sparse feature matching. Due to the relatively small number of interest points per image, metric learning networks can be exploited for this task. However, sparse feature matching approaches do not benefit from spatial smoothness priors which we incorporate into dense correspondence estimation via discrete optimization.

For the problem of binocular stereo matching, Zbontar et al. [16], Chen et al. [17] and Luo et al. [18] have demonstrated state-of-the-art performance by combining deep feature representations with discrete optimization. In similar spirit, Zagoruyko and Komodakis [47] learn Siamese matching networks for wide-baseline stereo matching. Motivated by this success, here we leverage feature learning and discrete optimization to tackle the more challenging problem of unconstrained 2D flow estimation.

Very recently, Bai et al. [48] have extended the approach of [18] to segment-wise epipolar flow where motion stereo has been estimated separately for each independently moving vehicle in the KITTI dataset [4]. In contrast, in this paper we neither assume rigidly moving objects nor the availability of highly accurate semantic instances. Thus, our method is also applicable to more general scenes as occuring, e.g., in the MPI Sintel optical flow challenge [6].

In [49], Siamese networks for optical flow computation have been combined with winner-takes-all matching and smoothing of the resulting correspondence field. While they use patch-wise max pooling operations to increase the size of the receptive field, we exploit computationally efficient dilated convolutions for this purpose. Furthermore, we investigate the usefulness of spatial priors and present a detailed empirical analysis of network architectures and settings.

3 Deep Discrete Flow

Menze et al. [11] formulate optical flow estimation as discrete MAP inference in a Markov random field with pairwise smoothness priors, followed by sub-pixel interpolation [20]. We follow their framework, but replace their hand-crafted Daisy features [41] with learned local and non-local representations to investigate the effect of feature learning on this framework as illustrated in Fig. 1. In Section 3.1, we first describe our local and context network architecture and provide details about training and inference. For completeness, we briefly review the discrete optimization framework [11] in Section 3.2.

3.1 Feature Learning using Dilated Convolutions

The classical approach to establish correspondences between two images is to search for the most similar patch in the target image, given a particular patch in the reference image, assuming that corresponding regions appear more similar than non-corresponding regions. Popular similarity measures for optical flow include brightness and gradient constancy [24], normalized cross-correlation [12], SIFT [30], Daisy [11] and hierarchical histograms of oriented gradients [32].

Following recent trends in computer vision [16,18,45–49], we use deep convolutional neural networks in order to learn better representations tailored for the task. In particular, we use Siamese architectures to process a pair of patches and produce a matching score as an indication of their level of similarity. In addition to traditional local 3×3 convolutional layers, we integrate context information by adopting dilated convolutions [18], which have recently demonstrated great performance in semantic segmentation. Compared to increasing the receptive field size using max-pooling operations, dilated convolutions have the advantage of not decreasing the image resolution, thus allowing for efficient dense inference with reuse of computation. In addition, patch-based dilated convolution networks can be efficiently trained as we demonstrate in this section.

For efficiency and due to the difficulty of training CNN-CRF models jointly, we train our model in a piece-wise fashion using auxiliary loss functions. That is, as illustrated in Fig. 1, we first train the local architecture using Δ_1, followed by the context architecture using Δ_2, and finally the CRF as well as hyperparameters of the post-processing stage using Δ_3. We also tried joint training on top of the pre-trained local and context networks, but observed no significant improvements. This agrees with the observations reported in [19].

Network Architecture: We use a Siamese network architecture composed of two shared-weight branches, one for the reference patch and one for the target patch. As we are also interested in calculating the backward flow, we have an additional backward Siamese network which shares weights with the forward network as illustrated in Fig. 1. Each of the branches consists of several building blocks where each block is defined as convolution, Batch Normalization, and ReLU for non-linearity except the last one which contains only a convolutional layer. The unit-length normalized output of the last layer is used as a feature vector of the patch. The similarity s between image pixels is calculated as the dot product between the respective feature vectors. As opposed to current trends in feature learning for stereo matching [16], we do not exploit fully connected layers for score computation as the large set of potential correspondences renders this computationally intractable (i.e., one network evaluation for each pixel pair).

Local Network: Our local network leverages 3×3 convolution kernels. The hyper-parameters of the network are the number of layers and the number of feature maps in each layer as specified in our experimental evaluation. We call this network local, because the size of each feature's receptive field is relatively small (i.e., $2n + 1$ where n denotes the number of blocks).

Context Network: Deeper architectures with more convolutional layers increase the receptive field size, possibly leading to improved performance. However, complex high capacity models are also hard to train and require a lot of data. Our context network increases the size of the receptive field with only modest increase in complexity by exploiting dilated convolutions [19]. In contrast to normal convolutions, dilated convolutions read the input feature maps at locations with a spatial stride larger than one. Thus, they take more contextual information into account while not increasing the number of parameters with respect to regular (i.e., 1-dilated) convolutions. In contrast to pooling operations, spatial information is not lost.

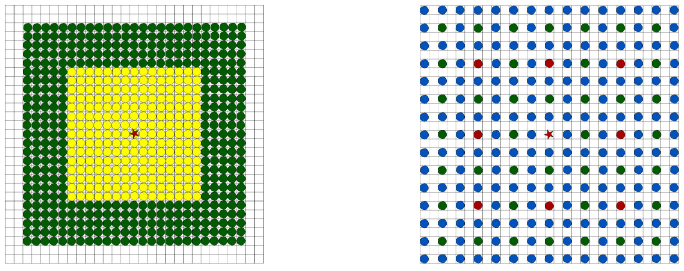
(a) Naïve (b) Proposed

Fig. 2: **Dilated Convolution Implementations** This figure shows the dilated convolution centers on a patch for the context network 2 and 12 with dilation factors 2, 4 and 8 as specified in Tab. 3. The center of the patch is marked with a red star and each color corresponds to a convolution center for a specific dilation factor, red for 4 dilations (shown in green), green for 2 dilations (shown in blue) and yellow for both. In other words, red dots show the convolution centers (outputs) for the 4-dilated convolutions which read their input values at the green points. Note that yellow points are only visible in (a) as red and green dots do not overlap in (b) due to the sparsity exploited by our approach.

Training: We consecutively train the local and the context network using the same auxiliary losses $\mathit{\Delta}_1 = \mathit{\Delta}_2$. As loss function, we leverage the hinge loss function which is defined for a positive-negative pair to penalize when the similarity score of the positive is not greater than the similarity score of the negative at least by margin: $\mathit{\Delta}_1(s_-, s_+) = \mathit{\Delta}_2(s_-, s_+) = \max(0, m + s_- - s_+)$. Here s_- denotes the the score of a wrong correspondence, s_+ denotes the score of a correct correspondence and m is the margin. We extract positive and negative patch pairs around points with valid ground-truth. Each positive is defined by the ground-truth flow with a perturbation of up to 1 pixel for robustness of the resulting feature representation. Unfortunately, the candidate set for the negative is the whole target image except the ground-truth matching point and thus intractable. Following [16], we sample negatives in a small circular region around each positive, keeping a minimum distance from the ground truth location. In particular, we use a threshold of 3 pixels for the minimum distance and a threshold of 6 pixels for the maximum distance to the ground truth flow. This ensures that the training set is composed of patches which are non-trivial to separate.

As illustrated in Fig. 2, a naïve implementation of dilated convolutions for training with patches would result in unnecessary computations. As we only need to forward/backward propagate information to/from the center of the patch, we can back-trace the source locations through the dilation hierarchy. Thus, we can implement the dilated convolution operation with sub-sampling and strides as a regular convolution as shown in Fig. 2.

Furthermore, we are able to exploit the fact that dilated convolutions on patches can be expressed as regular convolutions with strides as illustrated by our pseudo-code in Fig. 3. Our experiments show that this reduces computation

```
for $i = 1$ to $\#dilations$ do
    DilatedConvolution with $dilations[i]$
    if $i < \#dilations$ then
        Batch Normalization, ReLU
```

```
Sub-sampling with $dilations[1]$
for $i = 1$ to $\#dilations$ do
    if $i == \#dilations$ then
        $stride = 1$
    else
        $stride = \frac{dilations[i+1]}{dilations[i]}$
    Convolution with $stride$
    if $i < \#dilations$ then
        Batch Normalization, ReLU
```

Fig. 3: **Fast Patch-based Training of Dilated Convolutional Networks. Left:** A naïve implementation requires dilated convolution operations which are computationally less efficient than highly optimized `cudnn` convolutions without dilations. **Right:** The behavior of dilated convolutions can be replicated with regular convolutions by first sub-sampling the feature map and then applying 1-dilated convolutions with stride. Here *dilations* is denoting an array that specifies the dilation factor of the dilated convolution in each convolutional layer.

time as state-of-the-art implementations of regular convolutions (using `cudnn`) are significantly faster compared to dilated ones. This makes training with patch-based dilated convolutional networks much faster. At test time, we reuse the computations by dense convolutions over the image domain in traditional manner.

Inference: Differently from training with image patches, at test time the outputs of both branches of the network are computed for each point only once in a single forward pass of the full image, thereby reusing computations. The score computation between multiple reference points and every point on the target image can be performed efficiently as a single big matrix multiplication on the GPU. The first matrix is constructed by stacking reference feature descriptors as rows and the second matrix is built by stacking the target feature descriptors as columns. This waives the need for approximate search strategies as required in CPU-only model [11]. In our implementation, we handle the large GPU memory requirements by dividing the first matrix into individual chunks, balancing memory usage and computation time. We are able to further cut down inference time, as the post-processing stage which we use requires only every fourth pixel to be matched.

3.2 Discrete Optimization and Postprocessing

We follow the Discrete Flow approach [11] to aggregate information while respecting uncertainty in the matching. More specifically, we select the 300 best feature match hypotheses for each pixel on a regular 4-spaced grid, subject to mild non-maximum-suppression constraints (threshold 2 pixels), as input to a 4-connected MRF with pairwise smoothness constraints [11]. We find an approximate MAP solution using max-product belief propagation and the efficient robust pairwise potentials of [11].

As some pixels are occluded (i.e., not matchable) and due to the occurrence of outliers, we post-process our results using a forward-backward consistency check after discretely optimizing the forward and the backward flow. We further remove unlikely small segments from the solution using connected-component analysis. The resulting semi-dense flow map is fed into Epic Flow [20] for further refinement to sub-pixel accuracy. We optimize the parameters of the MRF and the post-processing stage using block coordinate descent with a 0/1 outlier loss $\Delta_3 = [\|\mathbf{f} - \hat{\mathbf{f}}\|_2 > 3\,\mathrm{Px}]$, averaged over all unoccluded pixels. Here, $\mathbf{f}$ is the ground truth flow vector, $\hat{\mathbf{f}}$ denotes its prediction and $[\cdot]$ is the Iverson bracket.

4 Experimental Results

We evaluate the performance of different local and context architectures, as well as the whole Deep Discrete Flow pipeline on MPI Sintel [6], KITTI 2012 [4] and KITTI 2015 [5]. Towards this goal, we trained separate networks for Sintel and KITTI, but merged the training sets of KITTI 2012 and KITTI 2015. For our internal evaluations, we follow the KITTI and MPI Sintel protocols: we split the training set into a training and a validation set using every fifth image for validation and the remaining images for training. While the images in MPI Sintel are temporally correlated, we found that Siamese networks without fully-connected layers do not suffer from over-fitting (i.e., the training and the validation errors behave similarly).

Note that the MPI Sintel and KITTI datasets leverage different evaluation metrics, average endpoint error (EPE) and outlier ratio, respectively. We follow each dataset's criteria for the final results, but report 3 pixel outlier ratios in all non-occluded regions for comparing the raw output of different network architectures, since the primary goal of our learned patch representations is to reduce the number of outliers.

Before passing them to the network, we normalize each image to zero mean and unit variance. Following common wisdom [50], we set the kernel size to 3 and use stride 1 convolutions unless otherwise specified. We start the training with standard uniform initialization in Torch and monitor the average outlier ratio in non-occluded regions on a subset of the validation set to stop training. We use stochastic gradient descent with momentum 0.9 for optimization, a batch size of 128, a hinge loss margin of 0.2 and a learning rate of 0.002 without any decay. We observe no over-fitting for neither our local nor our context networks. A detailed run-time analysis is provided in the supplementary material.

4.1 Baseline for Feature Matching

We leverage the winner-takes-all solution of Daisy [41] feature matching as pursued in Discrete Flow [11] as local baseline for our learned feature representations. For a fair comparison, we optimized the hyper-parameters of the Daisy

Arch.	Layers	Feature Maps									RFS
1	5	64	64	64	64	64					11
2	7	64	64	64	64	64	64	64			15
3	7	64	64	64	128	128	128	64			15
4	9	64	64	64	64	64	64	64	64	64	19
5	9	32	32	32	64	64	64	128	128	128	19

Table 1: **Local Architectures.** RFS denotes the receptive field size in pixels.

Arch.	Out-Noc
1	24.61 %
2	20.54 %
3	20.69 %
4	19.34 %
5	18.31 %

(a) **MPI Sintel**

Arch.	Out-Noc
1	34.60 %
2	29.71 %
3	29.89 %
4	30.37 %
5	27.36 %

(b) **KITTI**

	DF [11]	Optimized
MPI Sintel	29.97 %	19.16 %
KITTI	34.29 %	22.59 %

(c) **Daisy**

Table 2: **Comparison of Local Architectures**. Fig. (a)+(b) show the performance of different local architectures using winner-takes-all on the validation sets of MPI Sintel and KITTI, respectively. As baseline, Fig. (c) shows the performance of matching Daisy features on both datasets using the parameter setting of [11] in the first column and our re-optimized parameters in the second column. All numbers are percentages of non-occluded bad pixels as defined by the KITTI evaluation protocol.

feature descriptor [41] on a subset of the MPI Sintel training set using block coordinate descent to minimize the ratio of outliers. The results are shown in Table 2c. More details and the resulting combinations are provided in the supplementary material. Note that the optimized Daisy descriptor has 264 dimensions and a receptive field of approximately 40×40 pixels while Discrete flow [11] uses Daisy descriptors of length 68 with a receptive field of approximately 20×20 pixels. All numbers correspond to the WTA solution calculated over the full target image using exact matching.

4.2 Comparison of Local Architectures

We first compare five different local network architectures, including some of the architectures proposed in the literature for feature matching. Our starting point is the simple 5-layer architecture of [16] (architecture 1). We create additional architectures by changing the number of layers and feature maps in each layer as shown in Table 1. Architecture 5 corresponds to the recently proposed 9 layer network for stereo and flow in [18, 48]. Adding more layers changes network's receptive field size and has a clear effect on the performance as shown in Table 2. However, compared to our local architectures, the Daisy descriptor is fairly competitive. We attribute this to its larger receptive field size. In the next section, we explore context architectures to increase the receptive field size of our learned representations.

Arch.	Feature Maps	Arch.	Feature Maps	Dilations	RFS
1		11	64 128 256 512	2 4 8 16	+60
2		12	64 128 256	2 4 8	+28
3		13	64 128	2 4	+12
4		14	64 128 256	4 8 16	+56
5	all 64	15	64 128	8 16	+48
6		16	128 128	4 4	+16
7		17	64 64 128 128	2 2 4 4	+24
8		18	64 64 128 128 256 256	2 2 4 4 8 8	+28
9		19	128 128 256 256	8 8 16 16	+48

Table 3: **Context Architectures.** This table shows different context architectures and their receptive field sizes (RFS). We list the architectures that share the same set of dilations (and consequently RFS) in one row. Architectures in the same row differ solely by the number of feature maps in each layer. Receptive field sizes are added (+) to the RFS size of the respective local architecture.

Arch.	Out-Noc	Arch.	Out-Noc
1	16.32 %	11	11.92 %
2	14.51 %	12	12.19 %
3	15.65 %	13	14.32 %
4	15.27 %	14	12.53 %
5	15.66 %	15	13.34 %
6	15.10 %	16	13.59 %
7	15.68 %	17	13.69 %
8	15.50 %	18	13.01 %
9	20.04 %	19	14.55 %

(a) **MPI Sintel**

Arch.	Out-Noc	Arch.	Out-Noc
1	30.16 %	11	24.28 %
2	25.82 %	12	20.28 %
3	24.67 %	13	21.39 %
4	29.40 %	14	25.29 %
5	28.54 %	15	24.89 %
6	25.11 %	16	21.13 %
7	26.78 %	17	22.93 %
8	31.63 %	18	24.68 %
9	40.12 %	19	34.43 %

(b) **KITTI**

Table 4: **Comparison of Context Architectures.** This table shows the performance of different context architectures on top of local architecture 1 on the validation sets of MPI Sintel and KITTI. "Out-Noc" is defined as in Table 2.

4.3 Comparison of Context Architectures

Towards this goal, we fix the local architecture to the simplest one (architecture 1), and train different context architectures on top of this network. In a later section, we also show the performance of the best context architecture trained on top of the best local architecture. We create two types of context architectures, one by fixing the number of feature maps to 64, and one by changing the number of feature maps in each layer as summarized in Table 3.

Again, we compare the winner-takes-all performance in Table 4. We first note that the outlier ratio is significantly lower than the outlier ratio of the local architectures and the Daisy baseline shown in Table 2. Secondly, architectures which double the number of feature maps consistently outperform their respective constant counterparts. Finally, the 3-layer context architectures 2 and 12 are the best performing models in our set.

Local	Context	Winner-takes-All				Discrete Optimization				Full Pipeline			
		Noc		Occ		Noc		Occ		Noc		Occ	
		Out	EPE	Out	EPE	Out	EPE	Out	EPE	Out	EPE	Out	EPE
DF [11]		26.36 %	27.92	29.85 %	33.93	10.45 %	7.44	14.82 %	13.05	8.20 %	2.77	11.28 %	4.61
1	-	24.67 %	49.86	28.45 %	62.05	12.06 %	10.27	16.55 %	17.60	7.27 %	2.78	10.14 %	4.41
1	12	12.24 %	25.70	16.76 %	39.69	8.95 %	8.73	13.44 %	16.03	6.93 %	2.61	9.92 %	4.18
5	-	18.36 %	42.51	22.64 %	56.30	10.75 %	10.76	15.29 %	18.61	7.28 %	2.83	10.12 %	4.37
5	12	12.13 %	27.94	16.66 %	42.42	8.75 %	9.12	13.26 %	16.70	7.07 %	2.73	10.02 %	4.29

(a) **MPI Sintel**

Local	Context	Winner-takes-All				Discrete Optimization				Full Pipeline			
		Noc		Occ		Noc		Occ		Noc		Occ	
		Out	EPE	Out	EPE	Out	EPE	Out	EPE	Out	EPE	Out	EPE
DF [11]		33.01 %	30.21	40.99 %	49.16	10.84 %	3.73	21.81 %	22.38	8.55 %	1.76	18.43 %	4.49
1	-	34.38 %	69.45	42.00 %	99.55	13.38 %	8.75	24.00 %	29.92	8.35 %	2.14	16.73 %	4.44
1	12	20.00 %	41.67	29.33 %	77.21	13.26 %	9.90	23.54 %	31.00	9.75 %	2.57	18.27 %	5.35
5	-	27.18 %	58.58	35.69 %	92.92	13.07 %	9.63	23.63 %	31.70	8.74 %	2.38	17.12 %	4.77
5	12	22.09 %	55.46	31.10 %	92.85	14.01 %	12.78	24.16 %	34.72	10.62 %	2.99	19.10 %	5.95

(b) **KITTI**

Table 5: **Comparison of Model Components.** This table shows our results after winner-takes-all feature matching, after discrete optimization and the results of the full pipeline including post-processing and sub-pixel interpolation. We report end-point-errors (EPE) and outliers ratios (Out) both in non-occluded (Noc) and in all image regions (Occ) on the respective validation sets.

4.4 Evaluation of Model Components

Table 5 compares (from left to right) the results of winner-takes-all (WTA), discrete optimization and the full pipeline including post-processing and Epic Flow interpolation with respect to each other. For this experiment, we selected the simplest local architecture 1 as well as the top performing local architecture 5, both with and without context. For comparison, we also show the results of Discrete Flow with Daisy features as baseline ("DF [11]").

We first note that for WTA (first column), the context architectures improve the outlier ratio significantly with respect to local architectures for both datasets. However, this improvement is less visible after spatial smoothing (second and third column). We conclude that the gain of leveraging a larger receptive field can be partially compensated by using a spatial smoothing stage.

On the KITTI dataset, the improvements are less pronounced than on the MPI Sintel dataset. Here, our smallest local architecture (second row) outperforms Discrete Flow [11] slightly. Interestingly, the context architectures improve performance when considering the winner-takes-all (WTA) solution, but perform on par or even lead to degradation after spatial smoothing (second and third column). Our investigations revealed that the reason for this is the scale changes which are prominently present on KITTI (but less so on MPI Sintel) and which the networks have difficulty to cope with. We thus conclude that progress in invariant deep representations (in particular scale invariance) is necessary to address this issue.

4.5 Results on Test Set

We submitted our results to the MPI Sintel and KITTI 2012 and 2015 evaluation servers. We picked the best row for each dataset according to the results in Table 5, i.e., local model 1 in combination with context model 12 for the MPI Sintel and local model 1 alone for both KITTI datasets. In accordance with our results on the training/validation split, we obtain good results on MPI Sintel (best performing method amongst the published methods and second best performing method overall) while we are slightly better than Discrete Flow [11] on KITTI 2012 and KITTI 2015. We refer to the benchmark websites for details[1].

4.6 Qualitative Results

Fig. 4 shows visualizations of the different stages of our approach for several selected images from both MPI Sintel (top) and KITTI (bottom). Some failure cases are shown in Fig. 5. Each sub-figure shows from top-to-bottom: the input image and the ground-truth flow, Discrete Flow with Daisy features, our local architecture 1, our architectures 1 + 12. For each sub-figure, the first double column shows the WTA result on the grid, the second the results of discrete optimization and the last double column shows the final result.

As evidenced from these results, the proposed feature learning approach handles object boundaries more precisely and in general leads to lower errors for all inliers. However, these advantages diminish after discrete optimization and in particular Epic Flow interpolation as non-matched regions are responsible for the largest portion of the remaining errors. From Fig. 5, it is clearly visible that the learned representations suffer from strong scale changes which need to be addressed to further improve performance.

5 Conclusion and Future Work

We presented an efficient way of learning features for optical flow in a discrete framework by showing that dilated convolutions can be implemented efficiently for patch-based training. Learning features with context networks improves feature matching performance with respect to local architectures and manually engineered features for both the MPI Sintel and KITTI datasets. Although our experiments demonstrated that learning features with context is crucial for reducing outliers in the WTA solution of the network, large gains mostly diminish in the later stages of our pipeline. We found that large changes in scale pose problems to current feature learning approaches, prompting for the development of inherently scale invariant deep features. Finally, we remark that our current model's performance is hampered by piece-wise training. We therefore plan to investigate end-to-end training by back-propagating errors through all stages of our pipeline.

[1] http://sintel.is.tue.mpg.de/ http://www.cvlibs.net/datasets/kitti/

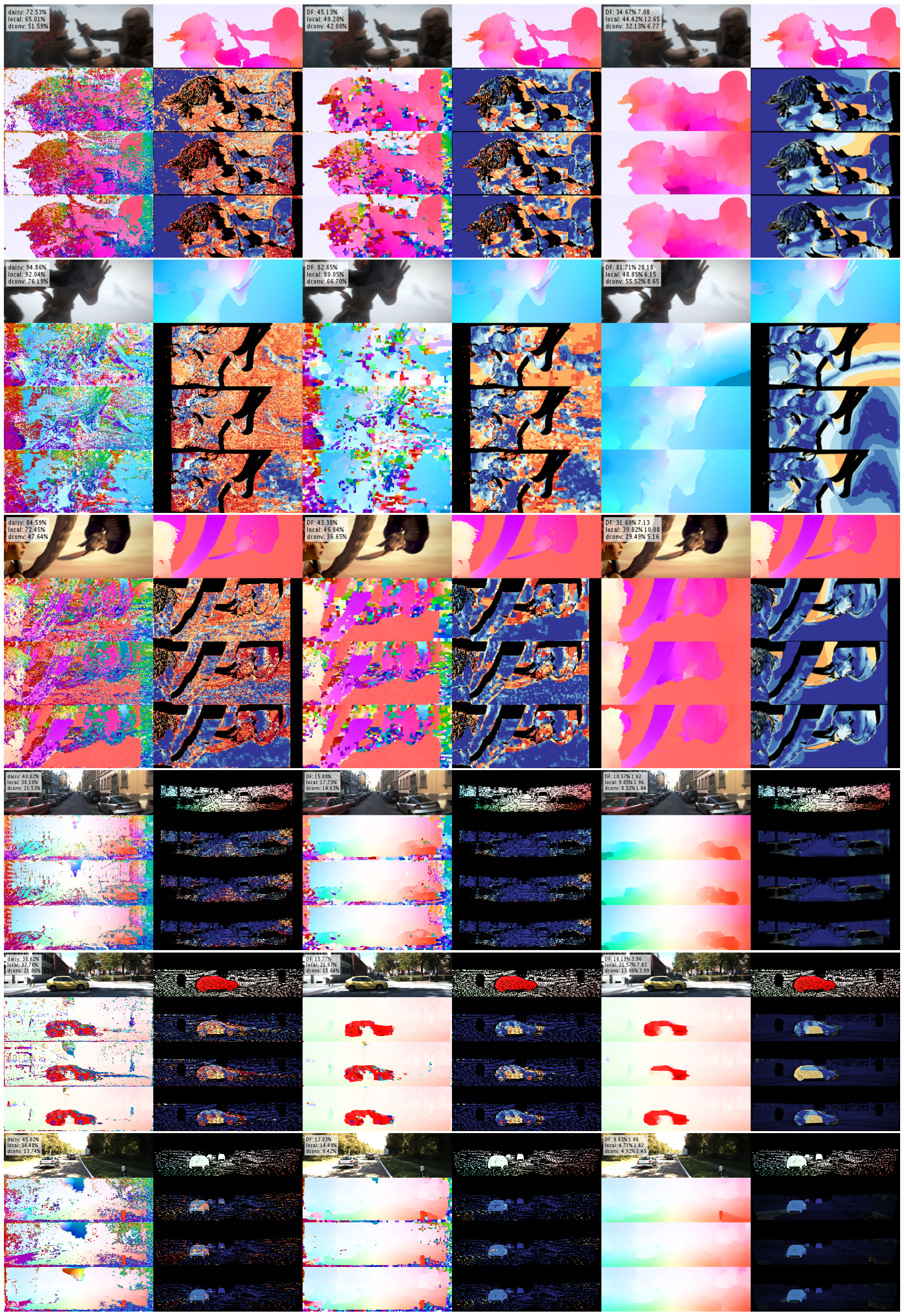

Fig. 4: **Qualitative Results.** See Section 4.6 for details.

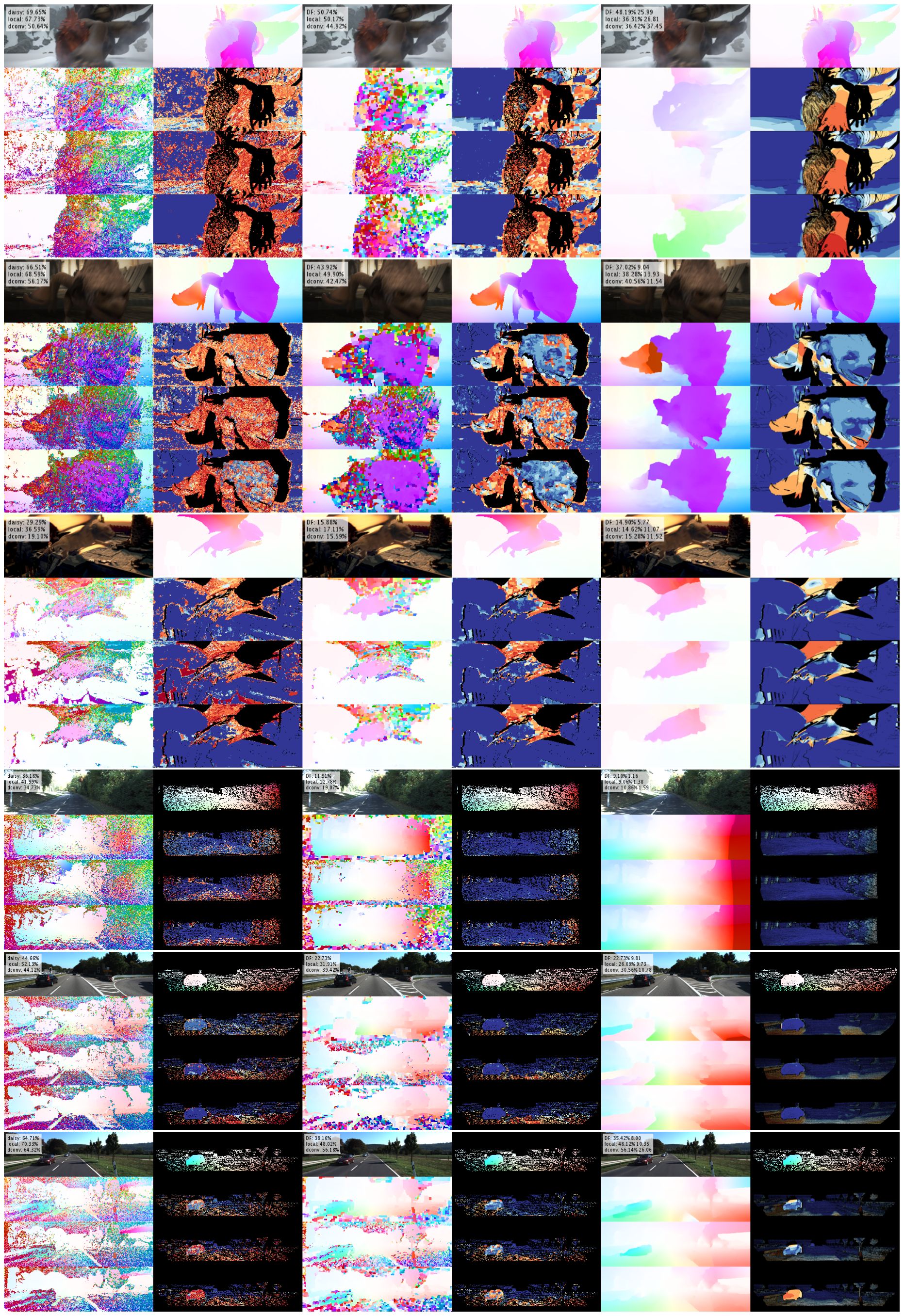

Fig. 5: **Qualitiative Results.** See Section 4.6 for details.

References

1. Zhang, H., Geiger, A., Urtasun, R.: Understanding high-level semantics by modeling traffic patterns. In: Proc. of the IEEE International Conf. on Computer Vision (ICCV). (2013)
2. Schönbein, M., Geiger, A.: Omnidirectional 3d reconstruction in augmented manhattan worlds. In: Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS). (2014)
3. Geiger, A., Lauer, M., Wojek, C., Stiller, C., Urtasun, R.: 3D traffic scene understanding from movable platforms. IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) **36** (2014) 1012–1025
4. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? The KITTI vision benchmark suite. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). (2012)
5. Menze, M., Geiger, A.: Object scene flow for autonomous vehicles. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). (2015)
6. Butler, D.J., Wulff, J., Stanley, G.B., Black, M.J.: A naturalistic open source movie for optical flow evaluation. In: Proc. of the European Conf. on Computer Vision (ECCV). (2012)
7. Horn, B.K.P., Schunck, B.G.: Determining optical flow. Artificial Intelligence (AI) **17** (1981) 185–203
8. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proc. of the International Joint Conf. on Artificial Intelligence (IJCAI). (1981)
9. Black, M.J., Anandan, P.: A framework for the robust estimation of optical flow. In: Proc. of the IEEE International Conf. on Computer Vision (ICCV). (1993)
10. Brox, T., Bruhn, A., Papenberg, N., Weickert, J.: High accuracy optical flow estimation based on a theory for warping. In: Proc. of the European Conf. on Computer Vision (ECCV). (2004)
11. Menze, M., Heipke, C., Geiger, A.: Discrete optimization for optical flow. In: Proc. of the German Conference on Pattern Recognition (GCPR). (2015)
12. Chen, Q., Koltun, V.: Full flow: Optical flow estimation by global optimization over regular grids. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). (2016)
13. Wulff, J., Black, M.J.: Efficient sparse-to-dense optical flow estimation using a learned basis and layers. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). (2015)
14. Hornacek, M., Besse, F., Kautz, J., Fitzgibbon, A.W., Rother, C.: Highly overparameterized optical flow using PatchMatch Belief Propagation. In: Proc. of the European Conf. on Computer Vision (ECCV). (2014)
15. Güney, F., Geiger, A.: Displets: Resolving stereo ambiguities using object knowledge. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). (2015)
16. Žbontar, J., LeCun, Y.: Stereo matching by training a convolutional neural network to compare image patches. Journal of Machine Learning Research (JMLR) **17** (2016) 1–32
17. Chen, Z., Sun, X., Wang, L., Yu, Y., Huang, C.: A deep visual correspondence embedding model for stereo matching costs. In: Proc. of the IEEE International Conf. on Computer Vision (ICCV). (2015)

18. Luo, W., Schwing, A., Urtasun, R.: Efficient deep learning for stereo matching. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). (2016)
19. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. In: Proc. of the International Conf. on Learning Representations (ICLR). (2016)
20. Revaud, J., Weinzaepfel, P., Harchaoui, Z., Schmid, C.: EpicFlow: Edge-preserving interpolation of correspondences for optical flow. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). (2015)
21. Bruhn, A., Weickert, J., Schnörr, C.: Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. International Journal of Computer Vision (IJCV) **61** (2005) 211–231
22. Demetz, O., Stoll, M., Volz, S., Weickert, J., Bruhn, A.: Learning brightness transfer functions for the joint recovery of illumination changes and optical flow. In: Proc. of the European Conf. on Computer Vision (ECCV). (2014)
23. Ranftl, R., Bredies, K., Pock, T.: Non-local total generalized variation for optical flow estimation. In: Proc. of the European Conf. on Computer Vision (ECCV). (2014)
24. Sun, D., Roth, S., Black, M.J.: A quantitative analysis of current practices in optical flow estimation and the principles behind them. International Journal of Computer Vision (IJCV) **106** (2013) 115–137
25. Werlberger, M., Trobin, W., Pock, T., Wedel, A., Cremers, D., Bischof, H.: Anisotropic Huber-L1 optical flow. In: Proc. of the British Machine Vision Conf. (BMVC). (2009)
26. Zach, C., Pock, T., Bischof, H.: A duality based approach for realtime TV-L1 optical flow. In: Pattern Recognition Letters. Springer Berlin Heidelberg (2007) 214–223
27. Zimmer, H., Bruhn, A., Weickert, J.: Optic flow in harmony. International Journal of Computer Vision (IJCV) **93** (2011) 368–388
28. Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M., Szeliski, R.: A database and evaluation methodology for optical flow. International Journal of Computer Vision (IJCV) **92** (2011) 1–31
29. Braux-Zin, J., Dupont, R., Bartoli, A.: A general dense image matching framework combining direct and feature-based costs. In: Proc. of the IEEE International Conf. on Computer Vision (ICCV). (2013)
30. Brox, T., Malik, J.: Large displacement optical flow: Descriptor matching in variational motion estimation. IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) **33** (2011) 500–513
31. Timofte, R., Gool, L.V.: Sparse flow: Sparse matching for small to large displacement optical flow. In: Proc. of the IEEE Winter Conference on Applications of Computer Vision (WACV). (2015)
32. Weinzaepfel, P., Revaud, J., Harchaoui, Z., Schmid, C.: DeepFlow: Large displacement optical flow with deep matching. In: Proc. of the IEEE International Conf. on Computer Vision (ICCV). (2013)
33. Steinbrücker, F., Pock, T., Cremers, D.: Large displacement optical flow computation without warping. In: Proc. of the IEEE International Conf. on Computer Vision (ICCV). (2009) 1609–1614
34. Yamaguchi, K., McAllester, D., Urtasun, R.: Robust monocular epipolar flow estimation. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). (2013)
35. Yamaguchi, K., McAllester, D., Urtasun, R.: Efficient joint segmentation, occlusion labeling, stereo and flow estimation. In: Proc. of the European Conf. on Computer Vision (ECCV). (2014)

36. Lempitsky, V.S., Roth, S., Rother, C.: Fusionflow: Discrete-continuous optimization for optical flow estimation. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). (2008)
37. Chen, Z., Jin, H., Lin, Z., Cohen, S., Wu, Y.: Large displacement optical flow from nearest neighbor fields. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). (2013)
38. Yang, J., Li, H.: Dense, accurate optical flow estimation with piecewise parametric model. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). (2015)
39. Mozerov, M.: Constrained optical flow estimation as a matching problem. IEEE Trans. on Image Processing (TIP) **22** (2013) 2044–2055
40. Besse, F., Rother, C., Fitzgibbon, A., Kautz, J.: PMBP: PatchMatch Belief Propagation for correspondence field estimation. International Journal of Computer Vision (IJCV) **110** (2014) 2–13
41. Tola, E., Lepetit, V., Fua, P.: Daisy: An efficient dense descriptor applied to wide baseline stereo. IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) **32** (2010) 815–830
42. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems (NIPS). (2012)
43. Fischer, P., Dosovitskiy, A., Ilg, E., Häusser, P., Hazirbas, C., Smagt, V.G.P., Cremers, D., Brox, T.: FlowNet: Learning optical flow with convolutional networks. arXiv.org **1504.06852** (2015)
44. Mayer, N., Ilg, E., Haeusser, P., Fischer, P., Cremers, D., Dosovitskiy, A., Brox, T.: A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: CVPR. (2016)
45. Han, X., Leung, T., Jia, Y., Sukthankar, R., Berg, A.C.: Matchnet: Unifying feature and metric learning for patch-based matching. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). (2015)
46. Simo-Serra, E., Trulls, E., Ferraz, L., Kokkinos, I., Fua, P., Moreno-Noguer, F.: Discriminative learning of deep convolutional feature point descriptors. In: Proc. of the IEEE International Conf. on Computer Vision (ICCV). (2015)
47. Zagoruyko, S., Komodakis, N.: Learning to compare image patches via convolutional neural networks. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). (2015)
48. Bai, M., Luo, W., Kundu, K., Urtasun, R.: Deep semantic matching for optical flow. arXiv.org **1604.01827** (2016)
49. Gadot, D., Wolf, L.: Patchbatch: a batch augmented loss for optical flow. Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2016)
50. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. Proc. of the International Conf. on Learning Representations (ICLR) (2015)