Fast Optical Flow Using Dense Inverse Search

Till Kroeger[1](✉), Radu Timofte[1], Dengxin Dai[1], and Luc Van Gool[1,2]

[1] Computer Vision Laboratory, D-ITET, ETH Zurich, Zurich, Switzerland
{kroegert,timofter,dai,vangool}@vision.ee.ethz.ch
[2] VISICS/iMinds, ESAT, KU Leuven, Leuven, Belgium

Abstract. Most recent works in optical flow extraction focus on the accuracy and neglect the time complexity. However, in real-life visual applications, such as tracking, activity detection and recognition, the time complexity is critical. We propose a solution with very low time complexity and competitive accuracy for the computation of dense optical flow. It consists of three parts: (1) inverse search for patch correspondences; (2) dense displacement field creation through patch aggregation along multiple scales; (3) variational refinement. At the core of our *Dense Inverse Search*-based method (DIS) is the efficient search of correspondences inspired by the inverse compositional image alignment proposed by Baker and Matthews (2001, 2004). DIS is competitive on standard optical flow benchmarks. DIS runs at 300 Hz up to 600 Hz on a single CPU core (1024×436 resolution. 42 Hz/46 Hz when including preprocessing: disk access, image re-scaling, gradient computation. More details in Sect. 3.1.), reaching the temporal resolution of human's biological vision system. It is order(s) of magnitude faster than state-of-the-art methods in the same range of accuracy, making DIS ideal for real-time applications.

1 Introduction

Optical flow estimation is under constant pressure to increase both its quality and speed. Such progress allows for new applications. A higher speed enables its inclusion into larger systems with extensive subsequent processing (*e.g.* reliable features for motion segmentation, tracking or action/activity recognition) and its deployment in computationally constrained scenarios (*e.g.* embedded systems, autonomous robots, large-scale data processing).

A robust optical flow algorithm should cope with discontinuities (outliers, occlusions, motion discontinuities), appearance changes (illumination, chromaticity, blur, deformations), and large displacements. Decades after the pioneering research of Horn and Schunck [4] and Lucas and Kanade [5] we have solutions for the first two issues [6,7] and recent endeavors lead to significant progress in handling large displacements [8–21]. This came at the cost of high run-times usually not acceptable in computationally constrained scenarios such as real-time applications. Recently, only very few works aimed at balancing accuracy

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-46493-0_29) contains supplementary material, which is available to authorized users.

B. Leibe et al. (Eds.): ECCV 2016, Part IV, LNCS 9908, pp. 471–488, 2016.
DOI: 10.1007/978-3-319-46493-0_29

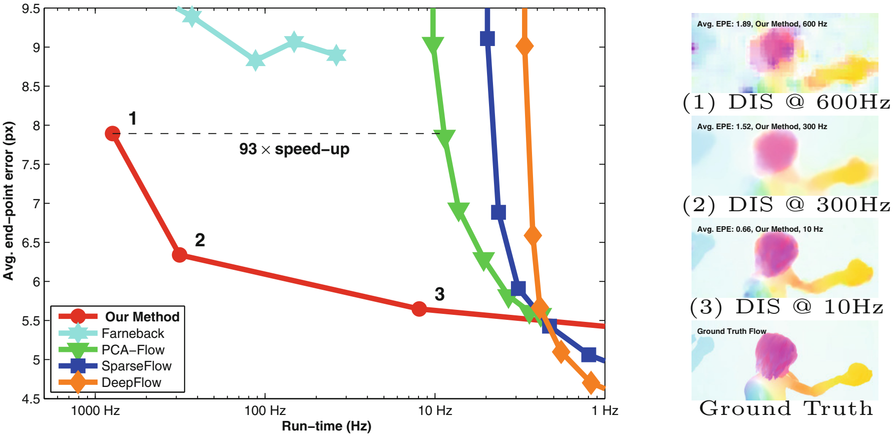


Fig. 1. Our DIS method runs at 10 Hz up to 600 Hz on a single core CPU for an average end-point pixel error smaller or similar to top optical flow methods at similar speed. This plot excludes preprocessing time for *all* methods. Details in Sects. 3.1 and 3.3.

and run-time in favor of efficiency [19,22,23], or employed massively parallelized dedicated hardware to achieve acceptable run-times [21,24,25]. In contrast to this, recently it has been noted for several computer vision tasks [3,26–29], that it is often desirable to trade-off powerful but complex algorithms for simple and efficients methods, and rely on high frame-rates and smaller search spaces for good accuracy. In this paper we focus on improving the speed of optical flow in general, non-domain-specific scenarios, while remaining close to the state-of-the-art flow quality. We propose two novel components with low time complexity, one using inverse search for fast patch correspondences, and one based on multi-scale aggregation for fast dense flow estimation. Additionally, a fast variational refinement step further improves the accuracy of our *dense inverse search*-based method. Altogether, we obtain speed-ups of 1–2 orders of magnitude over state-of-the-art methods at similar flow quality operating points (Fig. 1). The run-times are in the range of 10–600 Hz on 1024 × 436 resolution images, depending on the selected trade-off between run-time and accuracy, by using a single CPU core on a common desktop PC. The method reaches the temporal resolution of human's biological vision system [3]. To the best of our knowledge, this is the first time that optical flow at several hundred frames-per-second has been reached with such high flow quality on any hardware.

1.1 Related Work

Providing an exhaustive overview [30] of optical flow estimation is beyond the scope of this paper. Most of the work on improving the time complexity (without trading-off quality) combines some of the following ideas:

While, initially, the **feature descriptors** of choice were extracted sparsely, invariant under scaling or affine transformations [31], the recent trend in optical flow estimation is to densely extract rigid (square) descriptors from local frames [9,32,33]. HOG [34], SIFT [35], and SURF [36] are among the most popular square patch support descriptors. In the context of scene correspondence, the SIFT-flow [33] and PatchMatch [37] algorithms use descriptors or small patches. The descriptors are invariant only to similarities which may be insufficient especially for large displacements and challenging deformations [9]. Godot *et al.* [38] learn descriptors appropriate for optical flow using siamese CNNs.

The **feature matching** usually employs a (reciprocal) nearest neighbor operation [9,35,37,38]. Important exceptions are the recent works of Weinzaepfel *et al.* [17] (non-rigid matching inspired by deep convolutional nets), of Leordeanu *et al.* [11] (enforcing affine constraints), and of Timofte *et al.* [12] (robust matching inspired by compressed sensing). They follow Brox and Malik [9] and guide a variational optical flow estimation through (sparse) correspondences from the descriptor matcher and can thus handle arbitrarily large displacements. Xu *et al.* [20] combine SIFT [35] and PatchMatch [37] matching for refined flow level initialization at the expense of computational costs.

An **optimization** problem is often at the core of the flow extraction methods. The flow is estimated by minimizing an energy that sums up matching errors and smoothness constraints. While Horn and Schunck [4] proposed a variational approach to globally optimize the flow, Lucas and Kanade [5] solve the correspondence problem locally and independently for image patches. Local [5,23,39] methods are usually faster but less accurate than the global ones. Given location and smoothness priors over the image, MRF formulations are used [40,41]. Recently full optimization over discrete grids has been successfully applied [14,42].

Parallel computation is a natural way of improving the run-time of the optical flow methods by (re)designing them for parallelization. The industry historically favored specialized hardware such as FPGAs [43], while the recent years brought the advance of GPUs [21,24,25,44]. Yet, multi-core design on the same machine is the most common parallelization. However, many complex flow methods are difficult to adapt for parallel processing.

Learning. Most of the optical flow methods exploit training images for parameter tuning. However, this is only a rough embedding of prior knowledge. Only recently methods were proposed that successfully learn specific models from such training material. Wulff *et al.* [19] assume that any flow field can be approximated by a decomposition over a learned basis of flow fields. Fischer *et al.* [21] construct Convolutional Neural Networks (CNNs) to solve the optical flow estimation. Gadot *et al.* [38] learn patch similarities using siamese CNNs.

Coarse-to-fine optimizations have been applied frequently to flow estimation [9,45,46] to avoid poor local minima, especially for large motions, and thus to improve the performance and to speed up the convergence.

Branch and bound and **priority queues** have been used to find smart strategies to first explore the flow in the most favorable image regions and gradually refine it for the more ambiguous regions. This often leads to a reduction in computational costs. The PatchMatch methods [37,38,46] follow a branch and bound strategy, gradually fixing the most promising correspondences. Bao *et al.* [24] propose an edge-preserving extension (EPPM) based on PatchMatch.

Dynamic Vision Sensors [47], asynchronously capturing illumination changes at microsecond latency, have been used to compute optical flow. Benosman [3] and Barranco [29] note that realistic motion estimation, even with large displacements, becomes simple when capturing image evidence in the kilohertz-range.

1.2 Contributions

We present a novel optical flow method based on dense inverse search (DIS), which we demonstrate to provide high quality flow estimation at 10–600 Hz on a single CPU core. This method is 1–2 orders of magnitude times faster than previous results [12,17,19] on the Sintel [48] and KITTI [49] datasets when considering all methods at similar flow quality operating points. At the same time it is significantly more accurate compared to existing methods running at *equal speed* [5,22]. This result is based on two main contributions:

Fast inverse search for correspondences. Inspired by the inverse compositional image alignment of [1,2] we devise our inverse search procedure (explained in Sect. 2.1) for fast mining of a grid of patch-based correspondences between two input images. While usually less robust than exhaustive feature matching, we can extract a uniform grid of correspondences in microseconds.

Fast optical flow with multi-scale reasoning. Many methods assume sparse and outlier-free correspondences, and rely heavily on variational refinement to extract pixel-wise flow [12,17]. This helps to smooth-out small errors, and cover regions with flat and ambigious textures, where exhaustive feature matching fails. Other methods rely directly on pixel-wise refinement [24,25]. We chose a middle ground and propose a very fast and robust patch-averaging-scheme, performed only once per scale, after grid-based correspondences have been extracted. This step gains robustness against outlier correspondences, and initializes a pixel-wise variational refinement, performed once per scale. We reach an optimal trade-off between accuracy and speed at 300 Hz on a single CPU core, and reach 600 Hz without variational refinement at the cost of accuracy. Both operating points are marked as **(2)** and **(1)** in Figs. 1, 4 and 5.

Related to our approach is [25]. Here, the inverse image warping idea [2] is used on *all* the pixels, while our method optimizes patches independently. In contrast to our densification, done once per scale, [25] relies on frequent flow interpolations, requiring a high-powered GPU, and still is significantly slower than our CPU-only method. The paper is structured as follows: In Sect. 2 we introduce our DIS method. In Sect. 3 we describe the experiments, separately evaluate the patch-based correspondence search, and analyse the complete DIS algorithm with and without the variational refinement. In Sect. 4 we conclude the paper.

2 Proposed Method

In the following, we introduce our dense inverse search-based method (DIS) by describing: how we extract single point correspondences between two images in Sect. 2.1, how we merge a set of noisy point correspondences on each level s of a scale-pyramid into a dense flow field $\mathbf{U}_s$ in Sect. 2.2, how we refine $\mathbf{U}_s$ using variational refinement in Sect. 2.3, and possible extensions of DIS in Sect. 2.4.

2.1 Fast Inverse Search for Correspondences

The core component in our method to achieve high performance is the efficient search for patch correspondences. In the following we will detail how we extract one single point correspondence between two frames.

For a given template patch T in the reference image I_t, with a size of $\theta_{ps} \times \theta_{ps}$ pixels, centered on location $\mathbf{x} = (x, y)^T$, we find the best-matching sub-window of $\theta_{ps} \times \theta_{ps}$ pixels in the query image I_{t+1} using gradient descent. We are interested in finding a warping vector $\mathbf{u} = (u, v)$ such that we minimize the sum of squared differences over the sub-window between template and query location:

$$\mathbf{u} = \text{argmin}_{\mathbf{u}'} \sum_x \left[I_{t+1}(\mathbf{x} + \mathbf{u}') - T(\mathbf{x})\right]^2 . \tag{1}$$

Minimizing this quantity is non-linear and is optimized iteratively using the inverse Lukas-Kanade algorithm as proposed in [2]. For this method two steps are alternated for a number of iterations or until the quantity (1) converges. For the first step, the quantity (2) is minimized around the current estimate $\mathbf{u}$ for an update vector $\varDelta\mathbf{u}$ such that

$$\varDelta\mathbf{u} = \text{argmin}_{\varDelta\mathbf{u}'} \sum_x \left[I_{t+1}(\mathbf{x} + \mathbf{u} + \varDelta\mathbf{u}') - T(\mathbf{x})\right]^2 . \tag{2}$$

The first step requires extraction and bilinear interpolation of a sub-window $I_{t+1}(\mathbf{x} + \mathbf{u})$ for sub-pixel accurate warp updates. The second step updates the warping $\mathbf{u} \leftarrow \mathbf{u} + \varDelta\mathbf{u}$.

The original Lukas-Kanade algorithm [5] required expensive re-evaluation of the Hessian of the image warp at every iteration. As proposed in [2] the inverse objective function $\sum_x \left[T(\mathbf{x} - \varDelta\mathbf{u}) - I_{t+1}(\mathbf{x} + \mathbf{u})\right]^2$ can be optimized instead of (2), removing the need to extract the image gradients for $I_{t+1}(\mathbf{x} + \mathbf{u})$ and to re-compute the Jacobian and Hessian at every iteration. Due to the large speed-up this inversion has been used for point tracking in SLAM [50], camera pose estimation [51], and is covered in detail in [2] and our supplementary material.

In order to gain some robustness against absolute illumination changes, we mean-normalize each patch. One challenge of finding sparse correspondences with this approach is that the true displacements cannot be larger than the patch size θ_{ps}, since the gradient descent is dependent on similar image context in both patches. Often a coarse-to-fine approach with fixed window-size but changing image size is used [50,51], firstly, to incorporate larger smoothed contexts at coarser scales and thereby lessen the problem of falling into local optima, secondly, to find larger displacements, and, thirdly, to ensure fast convergence.

Algorithm 1. Dense Inverse Search (DIS)

```
Set initial flow field $\mathbf{U}_{\theta_{ss}+1} \leftarrow \mathbf{0}$
for $s = \theta_{ss}$ to $\theta_{sf}$ do
    (1.) Create uniform grid of $N_s$ patches
    (2.) Initialize displacements from $\mathbf{U}_{s+1}$
    for $i = 1$ to $N_s$ do
        (3.) Inverse search for patch $i$
    (4.) Densification: Compute dense flow field $\mathbf{U}_s$
    (5.) Variational refinement of $\mathbf{U}_s$
```

2.2 Fast Optical Flow with Multi-scale Reasoning

We follow such a multi-scale approach, but, instead of optimizing patches independently, we compute an intermediate dense flow field and re-initialize patches at each level. We do this because of two reasons: (1) the intermediate dense flow field smooths displacements and provides robustness, effectively filtering outliers and (2) it reduces the number of patches on coarser scales, thereby providing a speed-up. We operate in a coarse-to-fine fashion from a first (coarsest) level θ_{ss} in a scale pyramid with a downscaling quotient of θ_{sd} to the last (finest) level θ_{sf}. On each level our method consists of five steps, summarized in Algorithm 1, yielding a dense flow field $\mathbf{U}_s$ in each iteration s.

(1.) Creation of a grid: We initialize patches in a uniform grid over the image domain. The grid density and number of patches N_s is implicitly determined by the parameter $\theta_{ov} \in [0, 1)$ which specifies the overlap of adjacent patches and is always floored to an integer overlap in pixels. A value of $\theta_{ov} = 0$ denotes a patch adjacency with no overlap and $\theta_{ov} = 1 - \epsilon$ results in a dense grid with one patch centered on each pixel in the reference image.

(2.) Initialization: For the first iteration ($s = \theta_{ss}$) we initialize all patches with the trivial zero flow. On each subsequent scale s we initialize the displacement of each patch $i \in N_s$ at its location $\mathbf{x}$ with the flow from the previous (coarser) scale: $\mathbf{u}_{i,\text{init}} = \mathbf{U}_{s+1}(\mathbf{x}/\theta_{sd}) \cdot \theta_{sd}$.

(3.) Inverse search: Optimal displacements are computed independently for all patches, as detailed in Sect. 2.1. The search time required for each patch lies in the range of 1–2 μs, as detailed in the supplementary material.

(4.) Densification: After step three we have updated displacement vectors $\mathbf{u}_i$. For more robustness against outliers, we reset all patches to their initial flow $\mathbf{u}_{i,\text{init}}$ for which the displacement update $\|\mathbf{u}_{i,\text{init}} - \mathbf{u}_i\|_2$ exceeds the patch size θ_{ps}. We create a dense flow field $\mathbf{U}_s$ in each pixel $\mathbf{x}$ by applying weighted averaging to displacement estimates of all patches overlapping at $\mathbf{x}$ in the reference image:

$$\mathbf{U}_s(\mathbf{x}) = \frac{1}{Z} \sum_{i}^{N_s} \frac{\lambda_{i,\mathbf{x}}}{\max(1, \|d_i(\mathbf{x})\|_2)} \cdot \mathbf{u}_i, \qquad (3)$$

where the indicator $\lambda_{i,\mathbf{x}} = 1$ iff patch i overlaps with location $\mathbf{x}$ in the reference image, $d_i(\mathbf{x}) = I_{t+1}(\mathbf{x} + \mathbf{u}_i) - T(\mathbf{x})$ denotes the intensity difference between template patch and warped image at this pixel, $\mathbf{u}_i$ denotes the estimated displacement of patch i, and normalization $Z = \sum_i \lambda_{i,\mathbf{x}} / \max(1, \|d_i(\mathbf{x})\|_2)$.

(5.) Variational energy minimization of flow $\mathbf{U_s}$, as detailed in Sect. 2.3.

2.3 Fast Variational Refinement

We use the variational refinement of [17] with three simplifications: (i) We use no feature matching term, (ii) intensity images only, and (iii) refine only on the current scale. The energy is a weighted sum of intensity and gradient data terms (E_I, E_G) and a smoothness term (E_S) over the image domain Ω:

$$E(\mathbf{U}) = \int_{\Omega} \sigma \Psi(E_I) + \gamma \Psi(E_G) + \alpha \Psi(E_S) \, d\mathbf{x} \tag{4}$$

We use a robust penalizer $\Psi(a^2) = \sqrt{a^2 + \epsilon^2}$, with $\epsilon = 0.001$ for all terms as proposed in [52]. We use a separate penalization of intensity and gradient constancy assumption, with normalization as proposed in [53]: With the brightness constancy assumption $(\nabla_3^T I)\mathbf{u} = 0$, where $\nabla_3 = (\partial x, \partial y, \partial z)^T$ denotes the spatio-temporal gradient, we can model the intensity data term as $E_I = \mathbf{u}^T \, \bar{\mathbf{J}}_0 \, \mathbf{u}$. We use the normalized tensor $\bar{\mathbf{J}}_0 = \beta_0 \, (\nabla_3 I)(\nabla_3^T I)$ to enforce brightness constancy, with normalization $\beta_0 = (\|\nabla_2 I\|^2 + 0.01)^{-1}$ by the spatial derivatives and a term to avoid division by zero as in [53].

Similarly, E_G penalizes the gradient constancy: $E_G = \mathbf{u}^T \, \bar{\mathbf{J}}_{xy} \, \mathbf{u}$ with $\bar{\mathbf{J}}_{xy} = \beta_x(\nabla_3 I_{dx})(\nabla_3^T I_{dx}) + \beta_y(\nabla_3 I_{dy})(\nabla_3^T I_{dy})$, and normalizations $\beta_x = (\|\nabla_2 I_{dx}\|^2 + 0.01)^{-1}$ and $\beta_y = (\|\nabla_2 I_{dy}\|^2 + 0.01)^{-1}$. The smoothness term is a penalization over the norm of the gradient of displacements: $E_S = \|\nabla u\|^2 + \|\nabla v\|^2$. The non-convex energy $E(\mathbf{U})$ is minimized iteratively with θ_{vo} fixed point iterations and θ_{vi} iterations of Successive-Over-Relaxation for the linear system, as in [54].

2.4 Extensions

Our method lends itself to five extensions as follows:

i. Parallelization of all time-sensitive parts of our method (step 3, 5 in Sect. 2.2) is trivially achievable, since patch optimization operates independently. In the variational refinement the linear systems per pixel are solved independently in each inner iteration. With OpenMP we receive an almost linear speed-up with number of cores. Since the overhead of thread creation and management is significant for fast run-times, we use only one core in all experiments.

ii. Using RGB color images, instead of intensity only, boosts the score in most top-performing optical flow methods. In our experiments, we found that using color is not worth the observed increase of the run-time.

iii. Merging forward-backward flow estimations increases the accuracy. We found that the boost is not worth the observed doubling of the run-time.

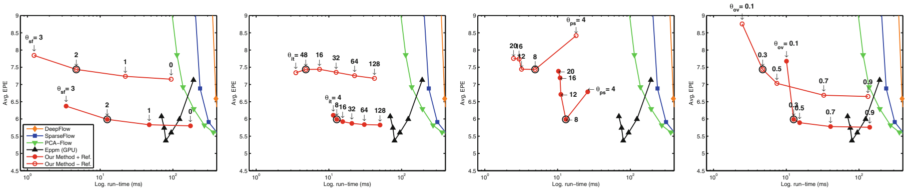


Fig. 2. Optical Flow result on Sintel with changing parameters. We set $\theta_{sf} = 2$, $\theta_{it} = 8$, $\theta_{ps} = 8$, $\theta_{ov} = 0.3$, marked with a black circle in all plots. From the left to right we vary the parameters θ_{sf}, θ_{it}, θ_{ps}, and θ_{ov} independently in each plot.

iv. Robust error norms, such as L1 and the Huber-norm [55], can be used instead of the L2-norm, implicit in the optimization of (1). Experimentally, we found that the gained robustness is not worth the slower convergence.

v. Using DIS for stereo depth, requires the estimation of the horizontal pixel displacement. Removing the vertical degree of freedom from DIS is trivial.

See the supplementary material for experiments on **i.-v.**

3 Experiments

In order to evaluate the performance of our method, we present three sets of experiments. Firstly, we conduct an analysis of our parameter selection in Sect. 3.1. Here, we also study the impact of variational refinement in our method. Secondly, we evaluate the inverse search (step 3 in Algorithm 1) in Sect. 3.2 without densification (step 4). The complete pipeline for optical flow is evaluated in Sects. 3.3 and 3.4. Thirdly, since the problem of recovering large displacements can also be handled by higher frame-rates combined with lower run-time per frame-pair, we conduct an experiment in Sect. 3.5 to analyse the benefit of higher frame-rates.

3.1 Implementation and Parameter Selection

We implemented[1] our method in C++ and run all experiments and baselines on a Core i7 CPU using a single core, and a GTX780 GPU for the EPPM [24] baseline. For *all* experiments on the Sintel and KITTI training datasets we report timings from which we exclude all operations which, in a typical robotics vision application, would be unnecessary, performed only once, or shared between multiple tasks: Disk access, creation of an image pyramid including image gradients with a downsampling quotient of 2, all initializations of the flow algorithms. We do this for our method and *all* baselines within their provided code. For EPPM, where only an executable was available, we subtracted the average overhead time of our method for fair comparison. Please see the supplementary material for variants of these experiments where preprocessing times are included for

[1] Source code available: http://www.vision.ee.ethz.ch/~kroegert/OFlow/.

Table 1. Parameters of our method. Parameters in **bold** have a significant impact on performance and are cross-validated in Sect. 3.1.

Parameter	Function
$\boldsymbol{\theta_{\mathbf{sf}}}$	**Finest scale in multi-scale pyramid**
$\boldsymbol{\theta_{\mathbf{it}}}$	**Number of gradient descent iterations per patch**
$\boldsymbol{\theta_{\mathbf{ps}}}$	**Rectangular patch size in (pixel)**
$\boldsymbol{\theta_{\mathbf{ov}}}$	**Patch overlap on each scale (percent)**
θ_{sd}	Downscaling quotient in scale pyramid
θ_{ss}	Coarsest scale in multi-scale pyramid
θ_{vo}, θ_{vi}	Number of outer and inner iterations for variational refinement
δ, γ, α	Intensity, gradient and smoothness weights for variational refinement

all methods. Our method requires 20 ms of preprocessing, spent on disk access (11 ms), image scaling and gradients (9 ms, unoptimized). For experiments on the Sintel and KITTI test datasets (Tables 3 and 4) we *include* this preprocessing time to be comparable with reported timings in the online benchmarks.

Parameter selection. Our method has four main parameters which affect speed and performance as explained in Sect. 2: θ_{ps} size of each rectangular patch, θ_{ov} patch overlap, θ_{it} number of iterations for the inverse search, θ_{sf} finest and final scale on which to compute the flow. We plot the change in the *average end-point error* (EPE) versus *run-time* on the Sintel (*training, final*) dataset [48] in Fig. 2. We draw three conclusions: Firstly, operating on finer scales (lower θ_{sf}), more patch iterations (higher θ_{it}), higher patch density (higher θ_{ov}) generally lowers the error, but, depending on the time budget, may not be worth it. Secondly, the patch size θ_{ps} has a clear optimum at 8 and 12 pixels. This also did not change when varying θ_{ps} at lower θ_{sf} or higher θ_{it}. Thirdly, using variational refinement always significantly reduced the error for a moderate increase in run-time.

In addition we have several parameters of lower importance, which are fixed for all experiments. We set $\theta_{sd} = 2$, *i.e.* we use an image pyramid, where the resolution is halved with each downscaling. We set the coarsest image scale $\theta_{ss} = 5$ for Sect. 3.3 and $\theta_{ss} = 6$ for Sect. 3.4 due to higher image resolutions. For different patch sizes and image pyramids the coarsest scale can be selected as $\theta_{ss} = \log_{\theta_{sd}}(2 \cdot width)/(f \cdot \theta_{ps})$ and raised to the nearest integer, to capture motions of at least $1/f$ of the image width. For the variational refinement we fix intensity, gradient and smoothness weights as $\delta = 5, \gamma = 10, \alpha = 10$ and keep iteration numbers fixed at $\theta_{vo} = 1 \cdot (s+1)$, where s denotes the current scale and $\theta_{vi} = 5$. In contrast to our comparison baselines [12,17,19], we do not fine-tune DIS for a specific dataset. We use a 20 percent subset of Sintel training to develop our method, and only the remaining training material is used for evaluation. All parameters are summarized in Table 1. If the flow is not computed up to finest scale ($\theta_{sf} = 0$), we scale-up the result (linearly interpolated) to full resolution for comparison for all methods. More details on implementation and timings of all parts of Algorithm 1 are provided in the supplementary material.

Table 2. Error of sparse correspondences (pixels). Columns left to right: (i) average end-point error over complete flow field, (ii) error in displacement range < 10 px., (iii) 10–40 px., (iv) > 40 px.

	EPE all	s0–10	s10–40	s40+
NN	32.06	13.64	53.77	101.00
DIS w/o Densification	7.76	2.16	8.65	37.94
DIS	4.16	**0.84**	4.98	23.09
DeepMatching [17]	**3.60**	1.27	**3.91**	**16.49**

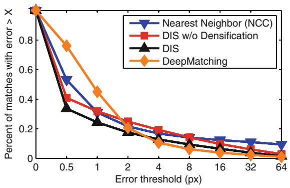


Fig. 3. Percent of sparse correspondences above error threshold.

3.2 Evaluation of Inverse Search

In this section we evaluate the sparse point correspondences created by inverse search on the Sintel training dataset. For each frame pair we initialized a sparse grid (given by Deep Matching [17]) in the first image and computed point correspondences in the second image. The correspondences are computed by (i) exhaustive *Nearest Neighbor* search on normalized cross-correlation (*NCC*), (ii) our method where we skip the densification step between each scale change (*DIS w/o Densification*), (iii) our method including the densification step (*DIS*), and using (iv) *DeepMatching* [17]. The results are shown in Fig. 3 and Table 2.

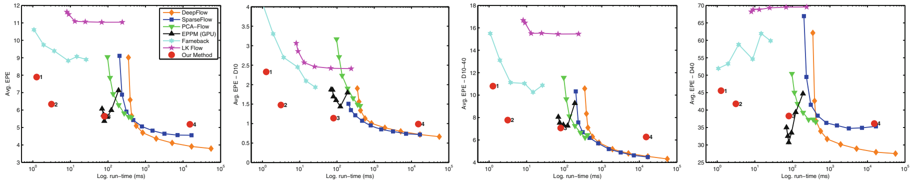
We have four observations: (i) Nearest Neighbor search has a low number of incorrect matches, but precise correspondences and is very prone to outliers. (ii) DeepMatching has a high percentage of erroneous correspondences (with small errors), but is very good at large displacements. (iii) In contrast to this, our method (DIS w/o Densification) generally performs well in the range of small displacements, but is strongly affected by outliers. This is due to the fact that the implicit SSD (sum of squared differences) error minimization is not invariant to changes in orientation, contrast, and deformations. (iv) Averaging all patches in each scale (DIS), taking into account their photometric error as described in Eq. (3), introduces robustness towards these outliers. It also decreases the error for approximately correct matches. Furthermore, it enables reducing the number of patches at coarser scales, leading to lower run-time.

3.3 MPI Sintel Optical Flow Results

Following our parameter evaluation in Sect. 3.1, we selected four operating points:

(1) $\theta_{sf} = 3, \theta_{it} = 016, \theta_{ps} = 08, \theta_{ov} = 0.30$, **at 600/46**[2] **Hz**,
(2) $\theta_{sf} = 3, \theta_{it} = 012, \theta_{ps} = 08, \theta_{ov} = 0.40$, **at 300/42 Hz**,
(3) $\theta_{sf} = 1, \theta_{it} = 016, \theta_{ps} = 12, \theta_{ov} = 0.75$, **at 10/8.3 Hz**,
(4) $\theta_{sf} = 0, \theta_{it} = 256, \theta_{ps} = 12, \theta_{ov} = 0.75$, **at 0.5/0.5 Hz**,

[2] Without/with image preprocessing: disk access, image gradients and re-scaling.



EPE, full range. EPE for < 10 px. EPE for 10 – 40 px. EPE for > 40 px.

Fig. 4. Sintel-training results: average end-point error (EPE, in pixels) versus run-time (millisecond) on various displacement ranges.

Table 3. Sintel test errors in pixels (http://sintel.is.tue.mpg.de/results), retrieved on 25th of July 2016 for *final* subset. Run-times are measured by us, except: †self-reported, and ‡on other datasets with same or smaller resolution.

	EPE all	s0–10	s10–40	s40+	Time (s)	CPU	GPU
FlowFields [16]	**5.81**	1.16	**3.74**	**33.89**	18 †	✓	
DeepFlow [17]	7.21	1.28	4.11	44.12	55	✓	
SparseFlow [12]	7.85	**1.07**	3.77	51.35	16	✓	
EPPM [24]	8.38	1.83	4.96	49.08	0.31		✓
PCA-Flow [19]	8.65	1.96	4.52	51.84	0.37	✓	
LDOF [9]	9.12	1.49	4.84	57.30	60 †‡	✓	
Classic+NL-fast [52]	10.09	1.09	4.67	67.81	120 †‡	✓	
DIS-Fast	10.13	2.17	5.93	59.70	**0.023**	✓	
SimpleFlow [23]	13.36	1.48	9.58	81.35	1.6 †‡		✓

We compare our method against a set of recently published baselines running on a single CPU core: DeepFlow [17], SparseFlow [12], PCA-Flow [19]; two older established methods: Pyramidal Lukas-Kanade Flow [5,56], Farneback's method [22]; and one recent GPU-based method: EPPM [24]. Since run-times for optical flow methods are strongly linked to image resolution, we incrementally speed-up all baselines by downscaling the input images by factor of 2^n, where n starting at $n = 0$ is increased in increments of 0.5. We chose this *non-intrusive* parameter of image resolution to analyse each method's trade-off between run-time and flow error. We bilinearly interpolate the resulting flow field to the original resolution for evaluation. We also experiment with temporal instead of spatial downsampling for the same purpose, as described in Sect. 3.5.

We run all baselines and DIS for all operating points on the Sintel [48] *final* training (Fig. 4) and testing (Table 3) benchmark. On the testing benchmark we report operating point **(2)** for DIS. As noted in Sect. 3.1, run-times for all methods are reported *without* preprocessing for the training dataset to facilitate comparison of algorithms running in the same environment at high speed, and *with* preprocessing for the online testing benchmark to allow comparison with self-reported times. From the experiments on the testing and training dataset,

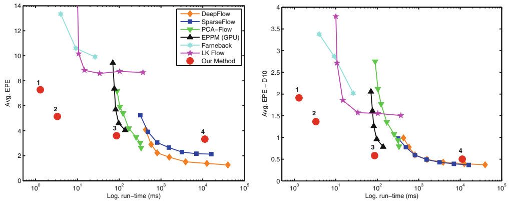


Fig. 5. KITTI (training) result. Average end-point error (px) versus run-time (ms) for all (left) and small displacements (right, s0–10). See supplementary material for large displacement errors.

Table 4. KITTI test results (http://www.cvlibs.net/datasets/kitti/eval_flow.php), retrieved on 25th of July 2016, for *all pixels, at 3px threshold*.

	Out-Noc	Out-All	Avg-Noc	Avg-All	Time (s)	CPU	GPU
PH-Flow [57]	**5.76 %**	**10.57 %**	**1.3 px**	**2.9 px**	800	✓	
DeepFlow [17]	7.22 %	17.79 %	1.5 px	5.8 px	17	✓	
SparseFlow [12]	9.09 %	19.32 %	2.6 px	7.6 px	10	✓	
EPPM [24]	12.75 %	23.55 %	2.5 px	9.2 px	0.25		✓
PCA-Flow [19]	15.67 %	24.59 %	2.7 px	6.2 px	0.19	✓	
eFolki [25]	19.31 %	28.79 %	5.2 px	10.9 px	0.026		✓
LDOF [9]	21.93 %	31.39 %	5.6 px	12.4 px	60	✓	
FlowNetS+ft [21]	37.05 %	44.49 %	5.0 px	9.1 px	0.08		✓
DIS-Fast	38.58 %	46.21 %	7.8 px	14.4 px	**0.024**	✓	
RLOF [39]	38.60 %	46.13 %	8.7 px	16.5 px	0.488		✓

we draw several conclusions: *Operating point* **(2)** points to the best trade-off between run-time and flow error. For the average EPE of around 6 pixels, DIS is approximately two orders of magnitude faster than the fastest CPU baseline (PCA-Flow [19]) and also more than one order of magnitude faster than the fastest GPU baseline (EPPM [24]). DIS can be further sped-up by removing the variational refinement as in *operating point* **(1)** while maintaining reasonable flow quality (see Fig. 6). We also tested using only the variational refinement without sparse initialization ($\theta_{it} = 0$), and found experimentally that the result is close to the trivial zero-flow solution. Finer resolution changes over scales and more iterations for the refinement will yield better results at significantly increased cost. *Operating point* **(3)** is comparable with the performance of EPPM, but slightly better for small displacements and worse for large displacements. If we use all available scales, and increase the number of iterations, we obtain *operating point* **(4)**. At the run-time of several seconds per frame pair, more complex methods, such as DeepFlow, perform better, in particular for large displacements. The supplementary material includes variants of Figs. 4 and 5, where preprocessing times are included, and flow error maps on Sintel, where typical failure cases of DIS at motion discontinuities and frame boundaries are observable.

3.4 KITTI Optical Flow Results

Complementary to the experiment on the synthetic Sintel dataset, we ran our method on the KITTI Optical Flow benchmark [49] for realistic driving scenarios. We use the same experimental setup and operating points as in Sect. 3.3. The result is presented in Figs. 5, 7 (training) and Table 4 (testing). Our conclusions from the Sintel dataset in Sect. 3.3 also apply for this dataset, suggesting a stable performance of our method, since we did not optimize any parameters for this dataset. On the online test benchmark, for which we *include* our preprocessing time, we are on par with RLOF [39] and the recently published FlowNet [21]. Even though both take advantage of a GPU, we are still significantly faster at comparable performance. In the supplementary material we include plots of more operating points on the training set of Sintel and KITTI, as well as the same plots as Figs. 4 and 5 where all preprocessing times are *included*.

3.5 High Frame-Rate Optical Flow

Often, a simpler and faster algorithm, combined with a higher temporal resolution in the data, can yield better accuracy than a more powerful algorithm, on lower temporal resolutions. This has been analysed in detail in [26] for the task of visual odometry. As noted in [3,29] this is also the case for optical flow, where large displacements, due to low-frame rate or strong motions are significantly more difficult to estimate than small displacements. In contrast to the recent focus on handling ever larger displacements [9,12,14,17,20,46], we want to analyse how *decreasing* the run-time while *increasing* the frame-rate affects our algorithm. For this experiment we selected a random subset of the Sintel training dataset, and synthesized new ground truth flow for lower frame-rates

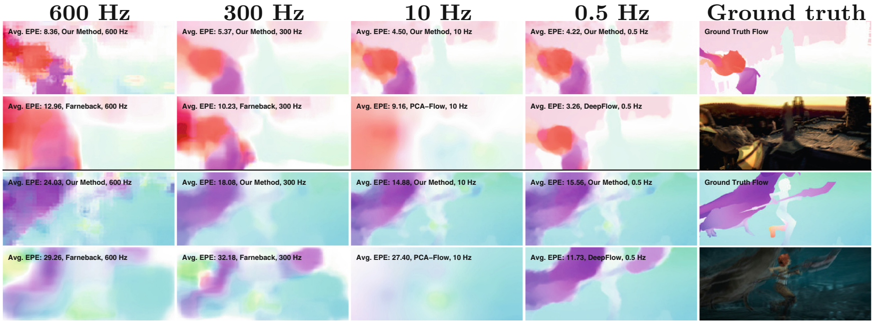


Fig. 6. Examplary results on Sintel (training). In each block of 2 × 6, top row, left to right: Our method for operating points (**1**)–(**4**), Ground Truth. Bottom row: Farneback 600 Hz, Farneback 300 Hz, PCA-Flow 10 Hz, DeepFlow 0.5 Hz, Original Image.

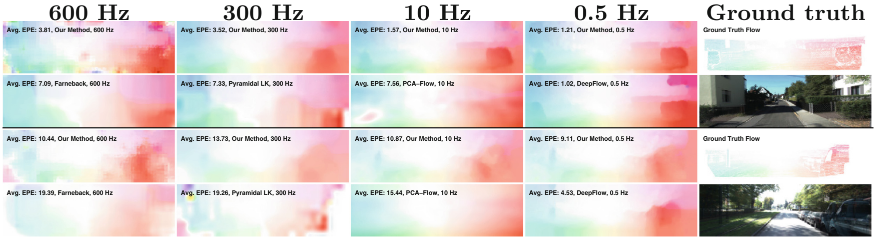

600 Hz
300 Hz
10 Hz
0.5 Hz
Ground truth
Avg. EPE: 3.81, Our Method, 600 Hz
Avg. EPE: 3.52, Our Method, 300 Hz
Avg. EPE: 1.57, Our Method, 10 Hz
Avg. EPE: 1.21, Our Method, 0.5 Hz
Ground Truth Flow
Avg. EPE: 7.09, Farneback, 600 Hz
Avg. EPE: 7.33, Pyramidal LK, 300 Hz
Avg. EPE: 7.56, PCA–Flow, 10 Hz
Avg. EPE: 1.02, DeepFlow, 0.5 Hz
Avg. EPE: 10.44, Our Method, 600 Hz
Avg. EPE: 13.73, Our Method, 300 Hz
Avg. EPE: 10.87, Our Method, 10 Hz
Avg. EPE: 9.11, Our Method, 0.5 Hz
Ground Truth Flow
Avg. EPE: 19.39, Farneback, 600 Hz
Avg. EPE: 19.26, Pyramidal LK, 300 Hz
Avg. EPE: 15.44, PCA–Flow, 10 Hz
Avg. EPE: 4.53, DeepFlow, 0.5 Hz

Fig. 7. Same as Fig. 6 but for KITTI (training) with Pyramidal LK as 300 Hz baseline.

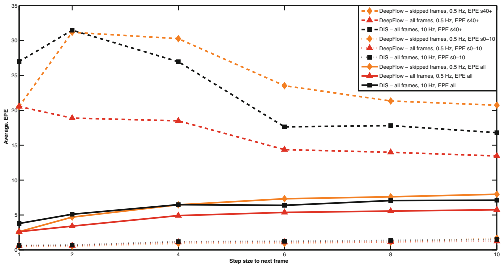


Fig. 8. Flow result on Sintel with low temporal resolution. Accuracy of DeepFlow on large displacements versus DIS on small displacements, tracked through *all intermediate* frames. As baseline we included the accuracy of DeepFlow for tracking small displacements. Note: While we use the same frame pairs to compute each vertical set of points, frame pairs differ over stepsizes.

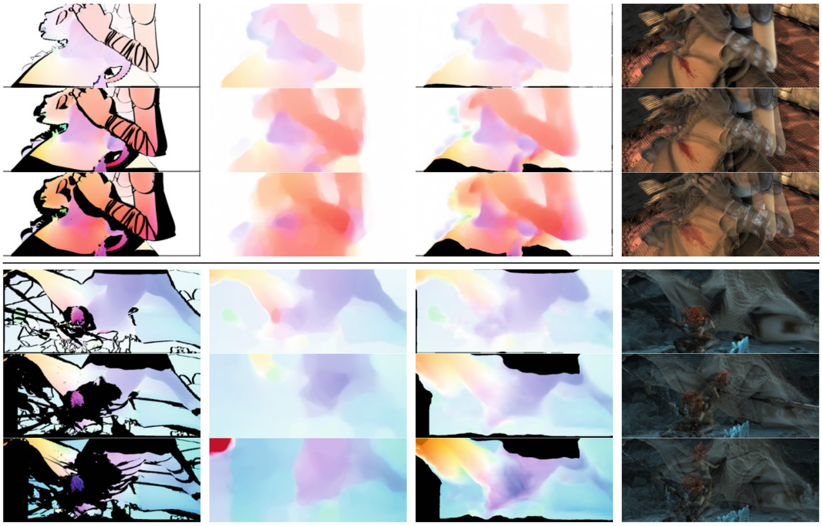


Fig. 9. Optical flow on Sintel with lower temporal resolution. In each block of 3 × 4: Rows, top to bottom, correspond to step sizes 1 (original frame-rate), 6, 10 frames. Columns, left to right, correspond to new ground truth, DeepFlow result, DIS result (through *all intermediate frames*), original images. Large displacements are significantly better preserved by DIS through higher frame-rates.

from the one provided in the dataset. We create new ground truth for 1/2 to 1/10 of the source frame-rate from the original ground truth and the additionally provided segmentation masks to invalidate occluded regions. We compare DeepFlow at a speed of 0.5 Hz on this lower temporal resolution against DIS (*operating point* **(3)**, 10 Hz), running through *all intermediate frames* at the original, higher frame-rate. Thus, while DeepFlow has to handle larger displacements in one frame pair, DIS has to handle smaller displacements, tracked through multiple frames and accumulates error drift. We observe (Fig. 8) that DIS starts to outperform DeepFlow when running at 2× the original frame-rate, notably for large displacements, while still being 10× faster. Figure 9 shows examples of the new ground truth, results of DeepFlow and DIS. We conclude, that it is advantageous to choose DIS over DeepFlow, aimed at recovering large displacements, when the combination of frame-rate and run-time per frame can be chosen freely.

4 Conclusions

In this paper we presented a novel and simple way of computing dense optical flow. The presented approach trades off a lower flow estimation error for large decreases in run-time: For the same level of error, the presented method is two orders of magnitude faster than current state-of-the-art approaches, as shown in experiments on synthetic (Sintel) and realistic (KITTI) optical flow benchmarks. In the future we will address open problems with our method: Due to the coarse-to-fine approach small and fast motions can sometimes get lost beyond recovery. A sampling-based approach to recover over-smoothed object motions at finer scales may alleviate this problem. The implicit minimization of the L2 matching error in our method is not invariant to many modes of change, such as in contrast, deformations, and occlusions. More robust error metrics may be helpful here. Furthermore, a GPU implementation may yield another significant speed-up.

Acknowledgments. This work was supported by ERC *VarCity (#273940)* and SNF *Tracking in the Wild (CRSII2_147693/1)*, and a NVIDIA GPU grant. We thank Richard Hartley for his pertinent input on this work.

References

1. Baker, S., Matthews, I.: Equivalence and efficiency of image alignment algorithms. In: CVPR (2001)
2. Baker, S., Matthews, I.: Lucas-Kanade 20 years on: a unifying framework. In: IJCV (2004)
3. Benosman, R., Clercq, C., Lagorce, X., Ieng, S.H., Bartolozzi, C.: Event-based visual flow. IEEE Trans. Neural Netw. Learn. Syst. **25**(2), 407–417 (2014)
4. Horn, B.K., Schunck, B.G.: Determining optical flow. In: Proceedings of SPIE 0281, Techniques and Applications of Image Understanding (1981)
5. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. IJCAI **81**, 674–679 (1981)

6. Black, M.J., Anandan, P.: The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. In: CVIU (1996)
7. Papenberg, N., Bruhn, A., Brox, T., Didas, S., Weickert, J.: Highly accurate optic flow computation with theoretically justified warping. IJCV **67**(2), 141–158 (2006)
8. Steinbrucker, F., Pock, T., Cremers, D.: Large displacement optical flow computation without warping. In: ICCV (2009)
9. Brox, T., Malik, J.: Large displacement optical flow: descriptor matching in variational motion estimation. IEEE Trans. PAMI **33**(3), 500–513 (2011)
10. Braux-Zin, J., Dupont, R., Bartoli, A.: A general dense image matching framework combining direct and feature-based costs. In: ICCV (2013)
11. Leordeanu, M., Zanfir, A., Sminchisescu, C.: Locally affine sparse-to-dense matching for motion and occlusion estimation. In: ICCV (2013)
12. Timofte, R., Van Gool, L.: Sparseflow: Sparse matching for small to large displacement optical flow. In: WACV, pp. 1100–1106, January 2015
13. Kennedy, R., Taylor, C.J.: Optical flow with geometric occlusion estimation and fusion of multiple frames. In: Tai, X.-C., Bae, E., Chan, T.F., Lysaker, M. (eds.) EMMCVPR 2015. LNCS, vol. 8932, pp. 364–377. Springer, Heidelberg (2015). doi:10.1007/978-3-319-14612-6_27
14. Menze, M., Heipke, C., Geiger, A.: Discrete optimization for optical flow. In: Gall, J., Gehler, P., Leibe, B. (eds.) GCPR 2015. LNCS, vol. 9358, pp. 16–28. Springer, Heidelberg (2015). doi:10.1007/978-3-319-24947-6_2
15. Revaud, J., Weinzaepfel, P., Harchaoui, Z., Schmid, C.: EpicFlow: edge-preserving interpolation of correspondences for optical flow. In: CVPR (2015)
16. Bailer, C., Taetz, B., Stricker, D.: Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In: ICCV (2015)
17. Weinzaepfel, P., Revaud, J., Harchaoui, Z., Schmid, C.: Deepflow: large displacement optical flow with deep matching. In: ICCV (2013)
18. Wills, J., Agarwal, S., Belongie, S.: A feature-based approach for dense segmentation and estimation of large disparity motion. IJCV **68**, 125–143 (2006)
19. Wulff, J., Black, M.J.: Efficient sparse-to-dense optical flow estimation using a learned basis and layers. In: CVPR, pp. 120–130 (2015)
20. Xu, L., Jia, J., Matsushita, Y.: Motion detail preserving optical flow estimation. IEEE Trans. PAMI **34**(9), 1744–1757 (2012)
21. Fischer, P., Dosovitskiy, A., Ilg, E., Häusser, P., Hazırbaş, C., Golkov, V., van der Smagt, P., Cremers, D., Brox, T.: Flownet: learning optical flow with convolutional networks. In: ICCV (2015)
22. Farnebäck, G.: Two-frame motion estimation based on polynomial expansion. In: Bigun, J., Gustavsson, T. (eds.) SCIA 2003. LNCS, vol. 2749, pp. 363–370. Springer, Heidelberg (2003). doi:10.1007/3-540-45103-X_50
23. Tao, M., Bai, J., Kohli, P., Paris, S.: Simpleflow: A non-iterative, sublinear optical flow algorithm. In: Computer Graphics Forum, vol. 31, pp. 345–353. Wiley Online Library (2012)
24. Bao, L., Yang, Q., Jin, H.: Fast edge-preserving patchmatch for large displacement optical flow. IEEE Trans. Image Process. **23**(12), 4996–5006 (2014)
25. Plyer, A., Le Besnerais, G., Champagnat, F.: Massively parallel lucas kanade optical flow for real-time video processing applications. J. Real-Time Image Proc. **11**(4), 1–18 (2014)
26. Handa, A., Newcombe, R.A., Angeli, A., Davison, A.J.: Real-Time camera tracking: when is high frame-rate best? In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7578, pp. 222–235. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33786-4_17

27. Dai, D., Kroeger, T., Timofte, R., Van Gool, L.: Metric imitation by manifold transfer for efficient vision applications. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
28. Srinivasan, N., Roberts, R., Dellaert, F.: High frame rate egomotion estimation. In: Chen, M., Leibe, B., Neumann, B. (eds.) ICVS 2013. LNCS, vol. 7963, pp. 183–192. Springer, Heidelberg (2013). doi:10.1007/978-3-642-39402-7_19
29. Barranco, F., Fermuller, C., Aloimonos, Y.: Contour motion estimation for asynchronous event-driven cameras. Proc. IEEE **102**(10), 1537–1556 (2014)
30. Fortun, D., Bouthemy, P., Kervrann, C.: Optical flow modeling and computation: a survey. Comput. Vis. Image Underst. **134**, 1–21 (2015). Image Understanding for Real-world Distributed Video Networks
31. Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., Van Gool, L.: A comparison of affine region detectors. IJCV **65**, 43–72 (2005)
32. Tola, E., Lepetit, V., Fua, P.: A fast local descriptor for dense matching. In: CVPR (2008)
33. Liu, C., Yuen, J., Torralba, A.: SIFT flow: Dense correspondence across scenes and its applications. TPAMI **33**(5), 978–994 (2011)
34. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR (2005)
35. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV **60**(2), 91–110 (2004)
36. Baya, H., Essa, A., Tuytelaarsb, T., Van Gool, L.: Speeded-up robust features (surf). CVIU **110**(3), 346–359 (2008)
37. Barnes, C., Shechtman, E., Goldman, D.B., Finkelstein, A.: The generalized patchmatch correspondence algorithm. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6313, pp. 29–43. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15558-1_3
38. Gadot, D., Wolf, L.: Patchbatch: a batch augmented loss for optical flow. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016
39. Senst, T., Eiselein, V., Sikora, T.: Robust local optical flow for feature tracking. IEEE Trans. Circ. Syst. Video Technol. **22**(9), 1377–1387 (2012)
40. Heitz, F., Bouthemy, P.: Multimodal estimation of discontinuous optical flow using markov random fields. TPAMI **15**(12), 1217–1232 (1993)
41. Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., Rother, C.: A comparative study of energy minimization methods for markov random fields with smoothness-based priors. TPAMI **30**(6), 1068–1080 (2008)
42. Chen, Q., Koltun, V.: Full flow: optical flow estimation by global optimization over regular grids. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016
43. Pauwels, K., Tomasi, M., Alonso, J.D., Ros, E., Van Hulle, M.: A comparison of FPGA and GPU for real-time phase-based optical flow, stereo, and local image features. IEEE Trans. Comput. **61**(7), 999–1012 (2012)
44. Zach, C., Pock, T., Bischof, H.: A duality based approach for realtime TV-L1 optical flow. In: Annual Symposium on German Association Pattern Recognition (2007)
45. Enkelmann, W.: Investigations of multigrid algorithms for the estimation of optical flow fields in image sequences. Comput. Vis. Graph. Image Process. **43**, 150–177 (1988)

46. Hu, Y., Song, R., Li, Y.: Efficient coarse-to-fine patchmatch for large displacement optical flow. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016
47. Lichtsteiner, P., Posch, C., Delbruck, T.: A 128×128 120 db 15 μs latency asynchronous temporal contrast vision sensor. IEEE J. Solid-State Circ. **43**(2), 566–576 (2008)
48. Butler, D.J., Wulff, J., Stanley, G.B., Black, M.J.: A naturalistic open source movie for optical flow evaluation. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7577, pp. 611–625. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33783-3_44
49. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: the KITTI dataset. In: IJRR (2013)
50. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: ISMAR (2007)
51. Forster, C., Pizzoli, M., Scaramuzza, D.: SVO: Fast semi-direct monocular visual odometry. In: ICRA, pp. 15–22, May 2014
52. Sun, D., Roth, S., Black, M.J.: Secrets of optical flow estimation and their principles. In: CVPR (2010)
53. Zimmer, H., Bruhn, A., Weickert, J.: Optic flow in harmony. IJCV **93**, 368–388 (2011)
54. Brox, T., Bruhn, A., Papenberg, N., Weickert, J.: High accuracy optical flow estimation based on a theory for warping. In: Pajdla, T., Matas, J. (eds.) ECCV 2004. LNCS, vol. 3024, pp. 25–36. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24673-2_3
55. Werlberger, M., Trobin, W., Pock, T., Wedel, A., Cremers, D., Bischof, H.: Anisotropic huber-L1 optical flow. In: BMVC (2009)
56. Bouguet, J.Y.: Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. Intel Corporation **5**, 1–10 (2001)
57. Yang, J., Li, H.: Dense, accurate optical flow estimation with piecewise parametric model. In: CVPR, pp. 1019–1027 (2015)