Efficient Sparse-to-Dense Optical Flow Estimation using a Learned Basis and Layers

Jonas Wulff Michael J. Black
Max Planck Institute for Intelligent Systems, Tübingen, Germany
{jonas.wulff,black}@tue.mpg.de

Abstract

We address the elusive goal of estimating optical flow both accurately and efficiently by adopting a sparse-to-dense approach. Given a set of sparse matches, we regress to dense optical flow using a learned set of full-frame basis flow fields. We learn the principal components of natural flow fields using flow computed from four Hollywood movies. Optical flow fields are then compactly approximated as a weighted sum of the basis flow fields. Our new PCA-Flow *algorithm robustly estimates these weights from sparse feature matches. The method runs in under 200ms/frame on the MPI-Sintel dataset using a single CPU and is more accurate and significantly faster than popular methods such as LDOF and Classic+NL. For some applications, however, the results are too smooth. Consequently, we develop a novel sparse layered flow method in which each layer is represented by PCA-Flow. Unlike existing layered methods, estimation is fast because it uses only sparse matches. We combine information from different layers into a dense flow field using an image-aware MRF. The resulting* PCA-Layers *method runs in 3.2s/frame, is significantly more accurate than PCA-Flow, and achieves state-of-the-art performance in occluded regions on MPI-Sintel.*

1. Introduction

Recent progress in optical flow estimation has led to increased accuracy, driven in part by benchmarks such as Middlebury [3], MPI-Sintel [10], and KITTI [16]. In particular, recent methods use either sparse or dense matching to capture long-range motions while exploiting traditional variational techniques to obtain high accuracy [9, 24, 26, 28, 29, 50, 53]. Still other methods use layered models or segmented regions to reason about occlusion relationships and better estimate motion at boundaries and in unmatched regions [24, 28, 43, 46]. In many applications, however, speed is at least as important. Most accurate methods require several seconds to many minutes per frame. Efficient methods are often less accurate or require a GPU (or both).

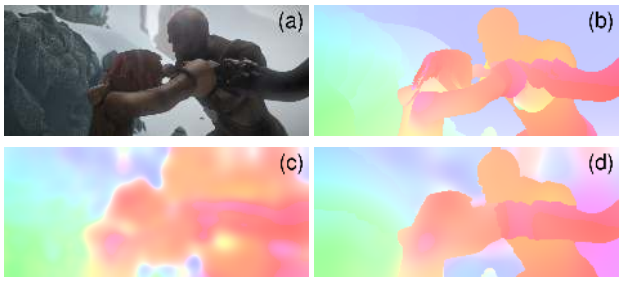


Figure 1: **Result overview.** (a) Image from MPI-Sintel; (b) Ground truth flow; (c) PCA-Flow; (d) PCA-Layers.

To address both accuracy and speed we propose a new sparse-to-dense approach that is based on sparse feature matching followed by interpolation. Sparse features are efficient to compute robustly and can capture long-range motions. By interpolating between these sparse matches, dense flow can be computed efficiently. However, due to outliers in the sparse matches and uneven covering of the images, generic interpolators do not work well. Instead, we learn an interpolant from training optical flow fields via principal component analysis (PCA).

The idea of learning linear models of flow is not new [7, 15], but previous work applied such models only in image patches, not to full images. To train our PCA model we use optical flow computed from 8 hours of video frames from four commercial movies using an existing flow algorithm (GPUflow [52]). To deal with noise in the training flow we use a robust PCA method that scales well to our huge training set [21].

Our method computes dense flow by estimating the location in the PCA subspace that best explains the sparse matches (Fig. 1(c)). At first it is not immediately obvious that one can represent generic flow fields using a low-dimensional PCA basis constructed from computed flow; we demonstrate that this indeed works.

This approach is very efficient. Our novel flow algorithm, called *PCA-Flow*, has a runtime of about 190 ms

per frame on a standard CPU; this is the fastest CPU-based method on both KITTI [16] and MPI-Sintel [10]. While there is a trade-off between accuracy and speed, and PCA-Flow cannot compete with the most accurate methods, it is significantly more accurate than the next fastest method on KITTI and is more accurate than recent, widely used methods such as LDOF [9] and Classic+NL [44] on MPI-Sintel. Interestingly, by learning from enough data, *we obtain a lower error than the algorithm used to train our PCA basis*.

While fast and sufficiently accurate for many tasks, PCA-Flow does not contain high-frequency spatial information and consequently over-smooths the flow at motion boundaries. To obtain sharp motion boundaries while retaining efficiency, we propose a novel layered flow model where each layer is a PCA-Flow field estimated from a subset of the sparse matches. Previous layered models are computationally expensive [45, 46]. By working with sparse matches and the learned PCA interpolator, the motion of each layer can be efficiently computed using Expectation Maximization (EM) [23].

To compute a final dense flow field, we must combine the flow fields estimated for each layer. We do so using a Markov Random Field (MRF) that incorporates image evidence to select among PCA-Flow fields at each pixel. This *PCA-Layers* method computes optical flow fields with much sharper motion boundaries and reduces the overall errors (Fig. 1(d)). At the same time, it is still reasonably fast, taking on average 3.2s/frame on MPI-Sintel. On Sintel it is more accurate than recent methods like MDP-Flow2 [53], EPPM [4], MLDP-OF [33], Classic+NLP [44] and the traditional layered approach, FC-2Layers-FF [46], which is a least two orders of magnitude slower. Most interestingly, PCA-Layers is particularly good in occluded (unmatched) regions, achieving lower errors there than DeepFlow [50] on Sintel.

For research purposes, the code for both methods and the learned optical flow basis are available at [1].

2. Previous work

Both optical flow and sparse feature tracking have long histories. Here we focus on the elements and combinations most related to our approach. Traditional variational methods for optical flow achieve good results for smooth and small motions, but fail in the presence of long-range motions. Feature matching, on the other hand, requires sufficient local image structure, and therefore only yields sparse results. They must therefore be "densified", either through explicit interpolation/regression, or through integration in a variational framework.

Sparse features in optical flow. The idea of using tracked features to estimate motion has its roots in early signal processing [5]. Early optical flow methods used correlation matching to deal with large motions [2].

Gibson and Spann [18] describe a two-stage method that first estimates sparse feature tracks, followed by an interpolation stage. Their tracking stage uses an MRF to enforce spatio-temporal smoothing, while the interpolation phase essentially optimizes a traditional dense optical flow objective. This makes the method computationally expensive.

Nicolescu and Medioni [34] use feature matching to get candidate motions and use tensor voting for interpolation. They then segment the flow into regions using only the flow information. Our PCA-Flow method has similar stages but uses a learned PCA basis for densification.

Lang et al. [26] first track sparse features over an image sequence and then use an image-guided filtering approach to interpolate between the features. They rely on temporal smoothness over multiple frames. Their results are visually appealing, but they report poor performance on Middlebury and do not evaluate on Sintel or KITTI.

Leordeanu et al. [28] use k-NN to find correspondences of features on a grid, and iteratively refine estimates of locally affine motion and occlusions. They follow this with a standard variational optical flow method [44]. Their algorithm requires 39 minutes per pair of frames on Sintel.

Several methods combine feature matching and traditional variational methods in a single optimization. Liu et al. [29] combine dense SIFT features with an optical flow-based regularization. Brox and Malik [9] match regions segmented in both frames using both region and HOG descriptors. These descriptor matches then form an additional data term in their dense flow optimization. Kennedy and Taylor [24] use a traditional data term in triangulated patches together with dense HOG matches; their method, TF+OFM performs well on MPI Sintel but is computationally expensive (350s on KITTI). Weinzaepfel et al. [50] use a similar approach, but propose a novel matching mechanism termed DeepFlow. Xu et al. [53] use sparse SIFT features to generate additional translational flow hypotheses. They then use a labeling approach to assign a pixel to one of those hypotheses, or to more traditional variational flow.

PatchMatch-based approaches fall between dense optical flow and sparse feature estimation [11, 30]. They compute a dense but approximate correspondence field, and refine this field using anisotropic diffusion [30] or segmentation into small surface patches [11].

Computing flow quickly. Zach et al. [54] were the first to demonstrate optical flow computation on a GPU. Using a traditional objective function they show how a total variation approach can be parallelized with a shader. They achieve realtime performance for small resolutions (320×240 pixels). Werlberger et al. [52] extend this approach to robust regularization. On a recent GPU, their algorithm takes approximately 2 seconds per frame at a resolution of 1024×436 pixels. Rannacher [35] presents an extremely fast method, but requires stereo images and a

pre-computed disparity field (for example, using an FPGA). Sundaram et al. [47] port the large-displacement optical flow method [9] to a GPU, with a runtime of 1.8 seconds for image pairs of 640×480 pixels. The reported runtimes depend on image resolution and the type of GPU.

Tao et al. [48] propose an algorithm that scales sublinearly with image input resolution by computing the flow on a selected subset of the image pixels and interpolating the remaining pixels. However, it still has a running time of around 2 seconds on Middlebury image pairs. Bao et al. [4] use a GPU to make their recent EPPM method run at about 0.25s/frame on Sintel. Our basic method is slightly less accurate, but around $60ms$ faster and does not require a GPU. Our layered method is more accurate but takes 3.2s/frame.

Non-local smoothing and layers. Optical flow regularization typically uses small neighborhoods but recent work suggests the value of non-local regularization. This can be done via median filtering [44, 51] or a densely connected MRF [25]. Here we achieve non-local smoothing using the learned PCA basis.

Layered models [13, 23, 42, 45, 46, 49] provide another approach. The advantage of layered models is that the segmentation is related to the scene geometry. The disadvantage of current methods, however, is that the runtime varies between tens of minutes [46] to tens of hours [45].

Learning spatial models of flow. Simple linear models (translational or affine) of optical flow in image patches have a long history [31]. As patch size grows, so does the complexity of the motion, and such models are no longer appropriate [23]. Consequently such linear models are typically used only locally.

Fleet et al. [15] extend the linear modeling approach by using PCA to learn basis flow fields from examples. They estimate the coefficients of these models in patches using a computationally expensive warping-based optimization scheme. Our approach of using sparse features is more efficient and can cope with long range correspondences. Chessa et al. [12] use basis flow fields in local patches, which accounts for affine motion plus deformation due to the geometric structure of the scene. Several authors have explored PCA models of walking humans [14, 20].

Roberts et al. [37] learn an optical flow subspace for egomotion using probabilistic PCA. Using this subspace, they estimate a dense flow field from sparse motion estimates. They restrict themselves to egomotion, train and test on similar sequences captured from the same platform, and use only a two-dimensional subspace with a low resolution of 45×13 pixels. Recent work extends this, but focuses on semantic classification into obstacle classes from this motion subspace, rather than accurate motion estimation [36].

Beyond these linear models, there is little work on learning spatial models for the full flow field. Roth and Black [39] learn a high-order MRF model for the spatial regularization of optical flow, but the neighborhood is still small (5×5 pixels). Rosenbaum et al. [38] learn a spatial model of flow and use this for flow *denoising* rather than estimation. Gibson and Marques [19] use an overcomplete dictionary of optical flow in patches to regularize the flow.

3. A learned basis for optical flow

Our basic assumption is that optical flow fields can be approximated as a weighted sum over a relatively small number of basis flow fields $\mathbf{b}_n, n = 1 \ldots N$, with corresponding weights w_n

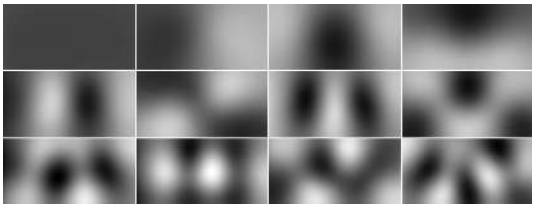
$$\mathbf{u} \approx \sum_{n=1}^{N} w_n \mathbf{b}_n. \tag{1}$$

Here, $\mathbf{u}$ and $\mathbf{b}_n$ are vectorized optical flow fields, containing the horizontal and vertical motions stacked as column vectors: $\mathbf{u} = \left(\mathbf{u}_x^\top, \mathbf{u}_y^\top\right)^\top$. We assume separate basis vectors for the horizontal and vertical flow components, so that the horizontal motion is spanned by $\{\mathbf{b}_n\}_{n=1,\ldots,\frac{N}{2}}$, and the vertical by $\{\mathbf{b}_n\}_{n=\frac{N}{2}+1,\ldots,N}$.

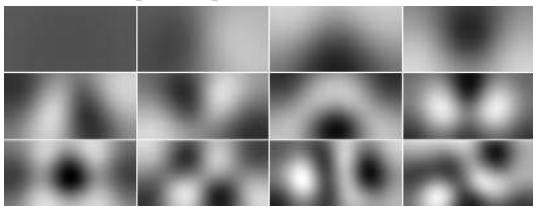
3.1. Learning the flow basis

To learn the basis flow fields, we use data from four Hollywood movies spanning several genres (*Star Wars*, *Babel*, *The Constant Gardener*, *Land of Plenty*). For each movie, we compute the optical flow using GPUFlow [52]. This method is not the most accurate (as we will see, it is less accurate than our PCA-Flow algorithm trained using it). However, it is the fastest method with an available reference implementation, and has a runtime of approximately 2 seconds per frame. Computing the optical flow takes approximately 4 days per movie. The computed flow is then resized to 512×256 pixels and the magnitudes of the flow values are scaled accordingly; this is the same resolution used in [50].

From the computed optical flow fields, we randomly select 180,000 frames, limited by the maximum amount of memory at our disposal. We first subtract the mean flow, which contains some consistent boundary artifacts caused by the GPUFlow method. Note that here, the dimensionality of our components is higher than the number of datapoints. However, compared to the theoretical dimensionality, we extract only a very small fraction of the principal components, here $N = 500$, 250 for the horizontal motion and 250 for the vertical. Since the computed optical flow contains outliers due to editing cuts and frames for which the optical flow computation fails, we use a robust PCA method to compute the principal components [21]. The total time required to extract 500 components is approximately 22 hours. However, this has to be done only once and offline; we make the learned basis available [1]. Figure 2 shows the first 12 flow components in the horizontal and vertical directions. Note that one could also train a com-



(a) Principal components for horizontal motion



(b) Principal components for vertical motion

Figure 2: First 12 components for horizontal and vertical motion. Contrast enhanced for visualization.

(a) Ground truth optical flow (b) Projected optical flow

Figure 3: Example of projecting Sintel ground truth flow onto the first 500 principal components.

bined basis for vertical and horizontal motion. In our experiments, however, separate bases consistently outperformed a combined basis. Note also that the first six components do not directly correspond to affine motion, in contrast with what was found for small flow patches [15].

Figure 2 reveals that the resulting principal components resemble the basis functions of a Discrete Cosine Transform (DCT). In order to achieve comparable performance to our learned basis with the same number of components, we generated a DCT basis with ten times more components and used basis pursuit to select the most useful ones. Despite this, the DCT basis gave slightly worse endpoint errors in our experiments and so we do not consider it further.

Figure 3 shows the projection of a ground truth flow field from Sintel onto the learned basis. Note that no Sintel training data was used in learning the basis, so this tests generalization. Also note that the Sintel sequences are quite complex and that the projected flow is much smoother; this is to be expected. For the impact of the number of principal components on the reconstruction accuracy, as well as for a quantitative comparison with a DCT basis, please see the **Supplemental Material** [1].

4. Estimating flow

Given an image sequence and the learned flow basis, we estimate the coefficients that define the optical flow. To that end, we first compute sparse feature matches to establish correspondences of key points between both frames. We then estimate the coefficients that produce a dense flow field that is consistent with both the matched scene motion and with the general structure of optical flow fields.

4.1. Sparse feature matching

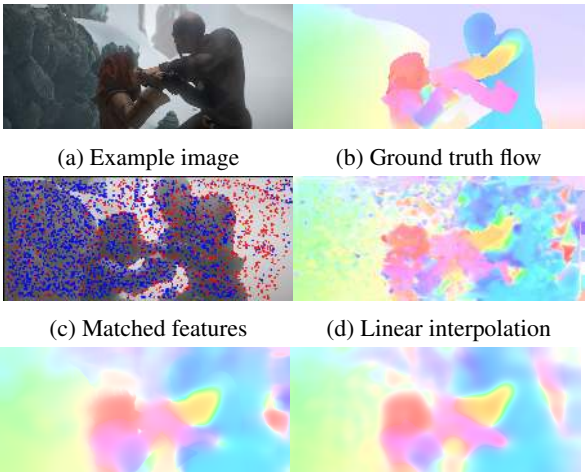
Our algorithm starts by estimating K sparse feature matches across neighboring frames; *i.e.* pairs of points $\{(\mathbf{p}_k, \mathbf{q}_k)\}, k = 1 \dots K$. $\mathbf{p}_k$ is the 2D location of a (usually visually distinct) feature point in frame 1, and $\mathbf{q}_k$ is the corresponding feature point location in frame 2. Each of these correspondences induces a displacement vector $\mathbf{v}_k = \mathbf{q}_k - \mathbf{p}_k = (v_{k,x}, v_{k,y})^\top$. Using sparse features has two main advantages. First, it provides a coarse estimate of image and object motions while being relatively cheap computationally. Second, it establishes long range correspondences, which are difficult for traditional, dense flow methods to estimate [9, 50, 53].

First, we normalize the image contrast using CLAHE [55] to increase detail that can be captured by the feature detectors. Then, we use the features from [17], which are designed for visual odometry applications. We found that to match features across *video* frames, these features work much better than image matching features such as SURF or SIFT. The latter are invariant against a wide range of geometric deformations which rarely occur in adjacent frames of a video, and hence return a large number of mismatches. Furthermore, the features we use are computationally much cheaper: currently, matching across two frames in MPI-Sintel takes on average 80 ms. Using sparse features creates the problem of low coverage in unstructured image regions. However, this problem also exists in classical optical flow: If image structure is missing, the data term becomes ambiguous, and flow computation relies on the regularization, just as our approach relies on the learned basis for interpolation.

Feature matches will always include outliers. We account for these in the matching process by formulating our optimization in a robust manner below. Figure 4(a) shows a frame and Fig. 4(c) the corresponding features. Features shown in blue have an error of less than 3 pixels; features with greater errors are red.

4.2. Regression: From sparse to dense

Extending the sparse feature matches to a dense optical flow field is a regression problem. Using our learned flow basis vectors $\mathbf{b}_n$, this can be formulated as finding the weighted linear combination of flow basis vectors that best



(a) Example image (b) Ground truth flow

(c) Matched features (d) Linear interpolation

(e) Guided interpolation (f) PCA-Flow (ours)

Figure 4: Sparse features and possible interpolations.

explains the detected feature matches. The weights then define the dense flow field.

First, consider a basic version of the method. This can be expressed as a simple least squares problem in the unknown $\mathbf{w} = (w_1, \ldots, w_N)^\top$:

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{A}\mathbf{w} - \mathbf{y}\|_2^2 \tag{2}$$

with

$$\mathbf{A} = \begin{pmatrix} \mathbf{b}_{1,x}(\mathbf{p}_1) & & \mathbf{b}_{N,x}(\mathbf{p}_1) \\ \vdots & & \vdots \\ \mathbf{b}_{1,x}(\mathbf{p}_K) & \cdots & \mathbf{b}_{N,x}(\mathbf{p}_K) \\ \mathbf{b}_{1,y}(\mathbf{p}_1) & & \mathbf{b}_{N,y}(\mathbf{p}_1) \\ \vdots & & \vdots \\ \mathbf{b}_{1,y}(\mathbf{p}_K) & & \mathbf{b}_{N,y}(\mathbf{p}_K) \end{pmatrix}. \tag{3}$$

We use nearest neighbor interpolation to compute the elements at fractional coordinates; better interpolation did not increase the accuracy in our experiments. $\mathbf{y} = (v_{1,x}, \ldots, v_{K,x}, v_{1,y}, \ldots, v_{K,y})^\top$ contains the motion of the matched points.

Solving Eq. (2) yields $\hat{\mathbf{w}}$, and thus the estimated dense optical flow field

$$\mathbf{u}_{est} = \sum_{n=1}^{N} \hat{w}_n \mathbf{b}_n. \tag{4}$$

Unfortunately, the sparse feature matches usually contain outliers. Since the search for feature matches is done across the whole image (*i.e.*, the spatial extent of feature motion is not limited), the errors caused by bad matches are often large, and thus can have a large influence on the solution of Eq. (2). Therefore, we solve a robust version of Eq. (2)

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \rho\left(\|\mathbf{A}\mathbf{w} - \mathbf{y}\|_2^2\right) \tag{5}$$

where $\rho(\cdot)$ is the robust Cauchy function

$$\rho(x^2) = \frac{\sigma^2}{2} \log\left[1 + \left(\frac{x}{\sigma}\right)^2\right]. \tag{6}$$

The parameter σ controls the sensitivity to outliers. Note that (6) is just one of many possible robust estimators [6]. We found the Cauchy estimator to work well.

If the input images have a different resolution than the flow basis, we first detect the features at full resolution, and scale their locations to the resolution of our flow basis. The weights are then estimated at this resolution, and the resulting optical flow field is upsampled and scaled again to the resolution of the input images.

Note that one could also estimate the coefficients using classical, dense estimation of parametric optical flow [31]. This is the approach used in [15]. We implemented this method and found, surprisingly, that its accuracy was comparable to our PCA-flow method with sparse feature matches. Because it is much slower, we do not consider it further here; see the **Supplemental Material** for results and comparisons [1].

4.3. Imposing a prior

Equation (5) does not take the distribution of $\mathbf{w}$ into account. The simplest prior on $\mathbf{w}$ is given by the eigenvalues computed during PCA on the training flow fields. New sequences may have quite different statistics, however. In KITTI, for example, the motion is caused purely by the egomotion of a car, and thus is less general than our training data. While KITTI and MPI-Sintel contain training data, the amount of data is insufficient to learn the full flow basis. We can, however, keep the basis fixed and adapt the prior. Since the prior lies in the 500-dimensional subspace defined by our flow basis, this requires much less training data. Given ground truth flow fields (*e.g.* from KITTI or Sintel), we project these onto our generic flow basis and compute $\mathbf{\Gamma}$, the inverse covariance matrix of the coefficients.

We express our prior using a Tikhonov regularizer on $\mathbf{w}$:

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \rho\left(\|\mathbf{A}\mathbf{w} - \mathbf{y}\|_2^2\right) + \lambda\|\mathbf{\Gamma}\mathbf{w}\|_2. \tag{7}$$

Intuitively, if a certain coefficient does not vary much in the projection of the training set onto the flow bases, we restrict this coefficient to small values during inference. When training data is available, this regularizer improves performance significantly.

We solve Eq. (7) using Iterative Reweighted Least Squares and refer to the method as **PCA-Flow**. Figure 4 shows the results of our method (4(f)) in comparison to two simpler methods that interpolate from sparse features to a dense flow field, nearest-neighbor interpolation (4(d)) and image-guided interpolation [22] (4(e)). These generic interpolation methods cannot detect and eliminate outliers

caused by wrong feature matches. Thus, their average endpoint errors on the Sintel test set (*linear*: 9.07 px; *guided*: 8.44 px) are higher than our basic method, PCA-Flow (7.74 px).

5. Dense motion from sparse layers

While the smooth flow fields generated by PCA-Flow may be appropriate for some applications, many applications require accurate motion boundaries. To that end, we develop a method that generates proposals using a layered method and combines them using image evidence.

5.1. Sparse layers

Here, we assume that a full optical flow field is composed of M simpler motions, where one of the motions is assigned to each pixel. The flow in each layer is represented by our learned basis as above with one modification: Since the motion of each layer should be simpler than for the full flow field, we change the prior. To obtain a layered representation for training, we first cluster the motion fields in our training set into layers with similar motions. Then, we compute $\mathbf{w}$ for each of the layers, compute the covariance matrix $\mathbf{\Sigma}$ from the weights of all segments across the whole training set, and use $\mathbf{\Gamma} = \mathbf{\Sigma}^{-1}$ in Eq. 7.

To compute the simpler motions at test time, we first cluster sparse feature matches using an EM algorithm with hard assignments[1]. To initialize, we cluster the features into M clusters using K-Means. The assignments of features to layers at iteration i are represented as assignment variables $a_k^{(i)}, k = 1 \dots K$, where $a_k^{(i)} = m$ means that feature point $\mathbf{p}_k$ is assigned to layer m. Given a set of layers, the distance of a feature point $\mathbf{p}_k$ to a layer m in iteration i is given as

$$d^{(i)}(\mathbf{p}_k, m) = \|\mathbf{u}_m^{(i-1)}(\mathbf{p}_k) - \mathbf{v}_k\|_2 + \alpha \|\mathbf{p}_k - \text{median}\left(\mathbf{p}_k | a_k^{(i-1)} = m\right)\|_2 \quad (8)$$

$\mathbf{u}_m$ is the optical flow field of layer m, $\mathbf{v}_k$ are the feature displacements as defined above. The right part is the distance of point $\mathbf{p}_k$ to the median of all features assigned to m in the previous iteration; initially, the medians are initialized to the center of the image. α is a weighting factor.

The features are then hard-assigned to the layers

$$a_k^{(i)} = \underset{\hat{m}}{\text{argmin}}\, d^{(i)}(\mathbf{p}_k, \hat{m}) \quad (9)$$

and the layers are updated to

$$\mathbf{w}_m^{(i)} = \text{estimate}\left(\left\{\mathbf{p}_k | a_k^{(i)} = m\right\}\right) \quad (10)$$

where $\text{estimate}(\cdot)$ solves Eq. (7) using a given subset of features. Since the motion of each layer is simpler than the motion of the whole image, the layers do not have to capture fine spatial detail. Consequently we reduce the number of linear coefficients from 500 to 100; this is sufficient for good results. We iteratively solve Eqs. (8)–(10) for 20 iterations, or until the assignments a_k do not change anymore.

[1] Soft assignments did not significantly change the results, and increased the runtime.

5.2. Combining the layers

The estimated layers give motion for their assigned features but we have created a new interpolation problem. We do not know which of the non-feature pixels correspond to which layer. Consequently we develop a method to combine the layers into a dense flow field. Several methods have been propose in the literature for related problems [27, 32]. Here we use a simple MRF model.

The layer estimation step generates M approximate optical flow fields, represented by their coefficients $\mathbf{w}_m$, and the final assignment variables a_k, denoting which sparse feature belongs to which motion model. We treat each layer's motion as a proposal. In addition to these M flow fields, we compute two additional flow proposals: a simple homography model, robustly fit to all matched features, and the full approximate flow field, *i.e.* solving Eq. (7) with all features and 500 principal components ("global model"). Therefore, $\tilde{M} = M + 2$.

At each image location $\mathbf{x}$, the task is now to assign a label $l(\mathbf{x}) \in 1 \dots \tilde{M}$, with the best flow model at this pixel. Then, the final optical flow field $\mathbf{u}_{final}(\mathbf{x})$ is given as

$$\mathbf{u}_{final}(\mathbf{x}) = \sum_{m=1}^{\tilde{M}} \delta[m = l(\mathbf{x})]\, \mathbf{u}_m(\mathbf{x}). \quad (11)$$

Finding $l(\mathbf{x})$ can be formulated as an energy minimization problem, which can be solved via multi-class graph cuts [8]:

$$\hat{l} = \underset{l}{\text{argmin}} \sum_{\mathbf{x}} E_u(\mathbf{x}, l(\mathbf{x})) + \gamma \sum_{\mathbf{y} \in n(\mathbf{x})} E_p(\mathbf{x}, \mathbf{y}, l(\mathbf{x}), l(\mathbf{y})) \quad (12)$$

where E_u and E_p are unary and pairwise energies, respectively, and $n(\mathbf{x})$ denotes the 4-neighborhood of $\mathbf{x}$. Omitting the arguments $\mathbf{x}, l(\mathbf{x})$, the unaries E_u are defined as

$$E_u = E_{warp} + \gamma_c E_{col} + \gamma_l E_{loc}. \quad (13)$$

Warping cost. The warping cost E_{warp} is a rectified brightness and gradient constancy term:

$$E_{warp}(\mathbf{x}, l(\mathbf{x})) = 1 - \exp\left(-\left(\frac{c(\mathbf{x}, l(\mathbf{x}))}{\sigma_w}\right)^2\right) \quad (14)$$

$$c(\mathbf{x}, l(\mathbf{x})) = \left\| \begin{pmatrix} I_1(\mathbf{x}) - I_2\left(\mathbf{x} + \mathbf{u}_{l(\mathbf{x})}\right) \\ \nabla_x I_1(\mathbf{x}) - \nabla_x I_2\left(\mathbf{x} + \mathbf{u}_{l(\mathbf{x})}\right) \\ \nabla_y I_1(\mathbf{x}) - \nabla_y I_2\left(\mathbf{x} + \mathbf{u}_{l(\mathbf{x})}\right) \end{pmatrix} \right\|_2 . \quad (15)$$

Color cost. We build an appearance model for each layer using the pixel colors at the feature points assigned to this layer. This helps especially in occluded regions, where the warping error is high, but the color provides a good cue about which layer to use.

$$E_{col}(\mathbf{x}, l(\mathbf{x})) = -\log p_{l(\mathbf{x})}\left(I_1\left(\mathbf{x}\right)\right) \tag{16}$$

$$p_{l(\mathbf{x})}\left(I_1\left(\mathbf{x}\right)\right) = \mathcal{N}\left(\mu_k, \mathbf{\Sigma}_k\right) \tag{17}$$

where $\mu_m, \mathbf{\Sigma}_m$ are computed from the pixels $\{I_1\left(\mathbf{p}_k\right) | a_k = m\}$ Here, we use simple multivariate Gaussian distributions as color models; we found these to perform as well as or better than multi-component Gaussian Mixture Models. For the homography model, we fit the distribution to all inlier features; for the global model, we fit it to all features.

Feature location cost. Lastly, we note that the features assigned to a given layer are often spatially clustered, and the quality of the model decreases for regions far away from the features. Therefore, we encourage spatial compactness of the layers using $E_{loc}(\mathbf{x}, l(\mathbf{x})) =$

$$1 - \sum_{k|a_k=l(\mathbf{x})} \frac{1}{\sqrt{2\pi\sigma_l^2}} \exp\left(-\frac{\left(\mathbf{x}-\mathbf{p}_k\right)^2}{\sigma_l^2}\right) \tag{18}$$

For the homography model, we again use only the inlier features, for the global model we use all.

Image-modulated smoothness. To enforce spatial smoothness, we use the image-modulated pairwise Potts energy from GrabCut [40]:

$$E_p\left(\mathbf{x}, \mathbf{y}, l\left(\mathbf{x}\right), l\left(\mathbf{y}\right)\right) = -\delta\left[l\left(\mathbf{x}\right) = l\left(\mathbf{y}\right)\right] \exp\left(-\frac{\left(I_1\left(\mathbf{x}\right) - I_2\left(\mathbf{y}\right)\right)^2}{2\mathbb{E}\left[\|\nabla I_1\|_2\right]}\right) \tag{19}$$

with $\mathbb{E}\left[\cdot\right]$ denoting the expected value. This energy encourages spatial smoothness of the layer labels between pixels, unless there is a strong gradient in the image. It thus allows the layer labels to change at image boundaries.

6. Evaluation

This section describes the performance of our algorithm in terms of accuracy on standard optical flow datasets. Additionally, we provide runtime information, and relate this to other current optical flow algorithms. All parameters are determined using cross validation on the available training sets. For PCA-Layers, we use $M = 6$ layers. For the other parameter values, experiments on the impact of the number of principal components as well as the feature matches, and more visual results, please see the **Supplemental Material** [1].

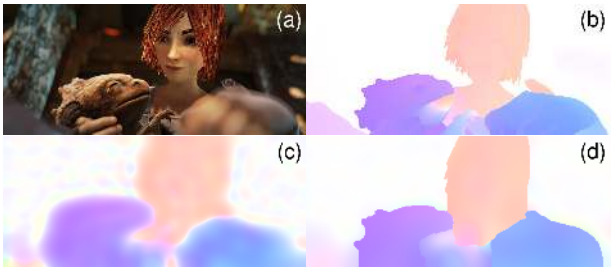


Figure 5: Results on MPI-Sintel: (a) Image; (b) Ground truth flow; (c) PCA-Flow; (d) PCA-Layers.

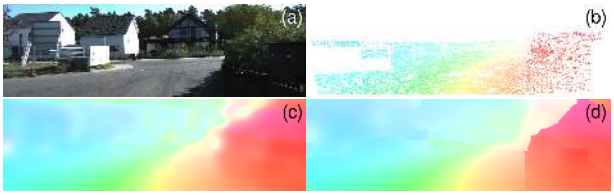


Figure 6: Results on KITTI: (a) Image; (b) Ground truth flow; (c) PCA-Flow; (d) PCA-Layers.

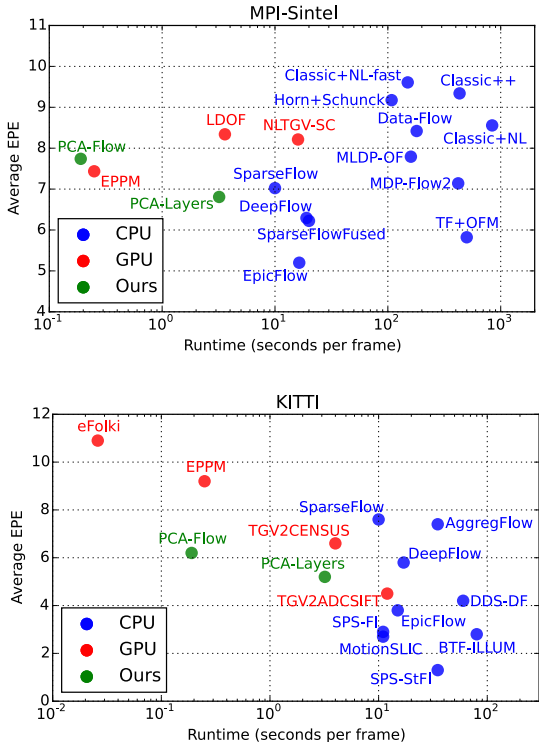


Figure 7: Avg. EPE vs. runtime on Sintel and KITTI

6.1. Evaluation on optical flow datasets

MPI-Sintel. Figure 5 shows an example from the clean pass of the training set of the MPI-Sintel optical flow benchmark for both PCA-Flow and PCA-Layers. PCA-Flow pro-

duces an oversmoothed optical flow field, but can correctly estimate most of the long-range motion. By computing and combining multiple layers, PCA-Layers is able to compute good motion and precisely locate motion boundaries.

On the Sintel test set, PCA-Flow currently ranks at place 22 of 36 on both the final pass (EPE = 8.65 px) and the clean pass (EPE = 6.83 px). While not the most accurate method, it only requires 190 ms per frame, while consistently outperforming the widely used methods LDOF and Classic+NL. Notably, we outperform GPUFlow [54], which we used to generate the training data. GPUFlow takes 2 s per frame, and achieves an average EPE of 12.64 px (clean) and 11.93 px (final).

Since the optical flow field generated by our method has a low spatial resolution, we compare it to Classic+NLP at an image resolution of 64x32 px. At this resolution, Classic+NLP achieves an EPE of 10.01 px, significantly worse than PCA-Flow, and requires 1.9 s per pair of frames.

PCA-Layers performs much better, with place 10 of 36 on the final pass, and 9 of 36 on the clean pass. It performs particularly well in the unmatched regions, where it ranks 5 of 36 on both passes. This demonstrates that our learned basis captures the structure of the optical flow well enough to make "educated guesses" about regions that are only visible in one frame of a pair.

KITTI. In addition to Sintel, we tested our method on the KITTI benchmark [16]. Since KITTI contains scenes recorded from a moving vehicle, we expect the subspace of possible motions to be relatively low-dimensional. Figure 6 shows an example. Note how we are able to accurately estimate the motion of the hedge on the right side, and how the boundaries are much sharper using PCA-Layers.

On the KITTI test set we obtain an average EPE (`Avg-All`) on all pixels of 6.2 px for PCA-Flow and 5.2 px for PCA-Layers. While the flow in KITTI is purely caused by the low-dimensional motion of the car, a segmentation into layers helps to better capture motion boundaries. All other published methods faster than 5 s per frame perform worse in average EPE, the next best being TGV2CENSUS with 6.6 px at a runtime of 4 s. No CPU-based method with similar accuracy to ours is faster than 10 s per frame.

In the `Out-Noc` metric (percentage of pixels with an error > 3 px), PCA-Flow ranks 40 of 63 (15.67%) and PCA-Layers ranks 34 of 63 (12.02%). These results reflect the approximate nature of our flow fields.

6.2. Timings

On a current CPU (Intel Xeon i7), the PCA-Flow algorithm takes on average 190 ms per frame on the MPI-Sintel dataset. 80 milliseconds are used for the feature matching. One advantage of our algorithm is that, when using longer sequences such as those from MPI-Sintel, the features for each frame have to be computed only once, which eliminates roughly 20 milliseconds runtime per frame. Fitting the flow basis itself requires approximately 90 milliseconds. PCA-Layers is significantly slower, requiring on average 3.2 seconds per pair of frames. Our implementation uses Python and its OpenCV bindings. The core IRLS algorithm is implemented in C++ using Armadillo [41]. Figure 7 plots the best and fastest published methods on Sintel and KITTI in the EPE-runtime plane[2]. Generally, all methods faster than PCA-Flow require a GPU, and achieve a much higher endpoint error. On the other hand, all methods that are more accurate than PCA-Layers are significantly slower.

7. Conclusion and future work

To summarize, this paper makes several contributions. First, we demonstrate the feasibility of computing a basis for global optical flow fields from a large amount of training data. Second, we show how this basis can be used with different datasets and scenarios, showing good generalization capabilities. Third, we propose an algorithm to efficiently estimate approximate optical flow, using sparse feature matches and the learned basis. Fourth, we develop a sparse layered optical flow method that is more efficient than existing dense layered methods. To do so, we combine several PCA-Flow fields using image evidence to improve accuracy and produce sharp motion boundaries. Fifth, we evaluate both algorithms on two current, challenging datasets for optical flow estimation. Our results suggest that sparse-to-dense methods can compete on accuracy with current non-sparse methods while achieving state-of-the-art efficiency using a standard CPU.

The existing basis vectors appear sufficient for the task and future work should focus on "assembling" coherent flow fields from the layer flows. Our current MRF is quite simple and much more sophisticated models exist for scene segmentation. In particular, including higher level scene classification and segmentation into the flow generation process holds promise. Still, even in its current form, the combination of speed and accuracy opens up opportunities to apply optical flow to new problems involving large video databases. In addition to speed, here the compactness of the optical flow descriptor that PCA-Flow provides is also beneficial, and we are currently exploring the use of PCA-Flow for large-scale video classification and indexing.

Lastly, our methods assume a linear subspace. While this does not necessarily reflect the reality of flow, preliminary experiments with other subspace extraction methods did not improve accuracy. Investigating the true structure of the subspace of optical flow remains future work.

Acknowledgements. We thank A. Geiger for providing the code for the features from [17].

[2]For Sintel, we used the timings as reported by the authors.

References

[1] http://pcaflow.is.tue.mpg.de. 2, 3, 4, 5, 7

[2] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2(3):283–310, 1989. 2

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

[4] L. Bao, Q. Yang, and H. Jin. Fast edge-preserving PatchMatch for large displacement optical flow. *Image Processing, IEEE Transactions on*, 23(12):4996–5006, Dec 2014. 2, 3

[5] D. I. Barnea and H. Silverman. A class of algorithms for fast digital image registration. *Computers, IEEE Transactions on*, C-21(2):179–186, Feb 1972. 2

[6] M. J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75 – 104, 1996. 5

[7] M. J. Black and Y. Yacoob. Recognizing facial expressions in image sequences using local parameterized models of image motion. *International Journal of Computer Vision*, 25(1):23–48, 1997. 1

[8] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, Nov 2001. 6

[9] T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(3):500–513, March 2011. 1, 2, 3, 4

[10] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, editors, *Computer Vision - ECCV 2012*, volume 7577 of *Lecture Notes in Computer Science*, pages 611–625. Springer Berlin Heidelberg, 2012. 1, 2

[11] Z. Chen, H. Jin, Z. Lin, S. Cohen, and Y. Wu. Large displacement optical flow from nearest neighbor fields. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2443–2450, June 2013. 2

[12] M. Chessa, F. Solari, S. P. Sabatini, and G. M. Bisio. Motion interpretation using adjustable linear models. In *BMVC*, pages 1–10, 2008. 3

[13] T. Darrell and A. Pentland. Robust estimation of a multi-layered motion representation. In *Visual Motion, 1991., Proceedings of the IEEE Workshop on*, pages 173–178, Oct 1991. 3

[14] R. Fablet and M. Black. Automatic detection and tracking of human motion with a view-based representation. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *Computer Vision - ECCV 2002*, volume 2350 of *Lecture Notes in Computer Science*, pages 476–491. Springer Berlin Heidelberg, 2002. 3

[15] D. J. Fleet, M. J. Black, Y. Yacoob, and A. D. Jepson. Design and use of linear models for image motion analysis. *International Journal of Computer Vision*, 36(3):171–193, 2000. 1, 3, 4, 5

[16] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research*, 32(11):1231–1237, Sept. 2013. 1, 2, 8

[17] A. Geiger, J. Ziegler, and C. Stiller. StereoScan: Dense 3D reconstruction in real-time. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 963–968, June 2011. 4, 8

[18] D. Gibson and M. Spann. Robust optical flow estimation based on a sparse motion trajectory set. *Image Processing, IEEE Transactions on*, 12(4):431–445, April 2003. 2

[19] J. Gibson and O. Marques. Sparse regularization of TV-L1 optical flow. In A. Elmoataz, O. Lezoray, F. Nouboud, and D. Mammass, editors, *Image and Signal Processing*, volume 8509 of *Lecture Notes in Computer Science*, pages 460–467. Springer International Publishing, 2014. 3

[20] T. Guthier, J. Eggert, and V. Willert. Unsupervised learning of motion patterns. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, volume 20, pages 323–328, Bruges, April 2012. 3

[21] S. Hauberg, A. Feragen, and M. Black. Grassmann averages for scalable robust PCA. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3810–3817, June 2014. 1, 3

[22] K. He, J. Sun, and X. Tang. Guided image filtering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(6):1397–1409, June 2013. 5

[23] A. Jepson and M. Black. Mixture models for optical flow computation. In *Computer Vision and Pattern Recognition (CVPR), 1993 IEEE Conference on*, pages 760–761, Jun 1993. 2, 3

[24] R. Kennedy and C. Taylor. Optical flow with geometric occlusion estimation and fusion of multiple frames. In X.-C. Tai, E. Bae, T. Chan, and M. Lysaker, editors,

Energy Minimization Methods in Computer Vision and Pattern Recognition, volume 8932 of *Lecture Notes in Computer Science*, pages 364–377. Springer International Publishing, 2015. 1, 2

[25] P. Krähenbühl and V. Koltun. Efficient nonlocal regularization for optical flow. In A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, editors, *Computer Vision - ECCV 2012*, volume 7572 of *Lecture Notes in Computer Science*, pages 356–369. Springer Berlin Heidelberg, 2012. 3

[26] M. Lang, O. Wang, T. Aydin, A. Smolic, and M. Gross. Practical temporal consistency for image-based graphics applications. *ACM Trans. Graph.*, 31(4):34:1–34:8, July 2012. 1, 2

[27] V. Lempitsky, S. Roth, and C. Rother. Fusionflow: Discrete-continuous optimization for optical flow estimation. In *Computer Vision and Pattern Recognition (CVPR), 2008 IEEE Conference on*, pages 1–8, June 2008. 6

[28] M. Leordeanu, A. Zanfir, and C. Sminchisescu. Locally affine sparse-to-dense matching for motion and occlusion estimation. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1721–1728, Dec 2013. 1, 2

[29] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(5):978–994, May 2011. 1, 2

[30] J. Lu, H. Yang, D. Min, and M. Do. Patch Match Filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1854–1861, June 2013. 2

[31] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981. 3, 5

[32] O. Mac Aodha, A. Humayun, M. Pollefeys, and G. Brostow. Learning a confidence measure for optical flow. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(5):1107–1120, May 2013. 6

[33] M. Mohamed, H. Rashwan, B. Mertsching, M. Garcia, and D. Puig. Illumination-robust optical flow using a local directional pattern. *Circuits and Systems for Video Technology, IEEE Transactions on*, 24(9):1499–1508, Sept 2014. 2

[34] M. Nicolescu and G. Medioni. Layered 4D representation and voting for grouping from motion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(4):492–501, April 2003. 2

[35] J. Rannacher. Realtime 3D motion estimation on graphics hardware. Undergraduate Thesis, Heidelberg University, 2009. 2

[36] R. Roberts and F. Dellaert. Direct superpixel labeling for mobile robot navigation using learned general optical flow templates. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 1032–1037, Sept 2014. 3

[37] R. Roberts, C. Potthast, and F. Dellaert. Learning general optical flow subspaces for egomotion estimation and detection of motion anomalies. In *Computer Vision and Pattern Recognition (CVPR), 2009 IEEE Conference on*, pages 57–64, June 2009. 3

[38] D. Rosenbaum, D. Zoran, and Y. Weiss. Learning the local statistics of optical flow. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2373–2381. MIT Press, 2013. 3

[39] S. Roth and M. J. Black. On the spatial statistics of optical flow. *International Journal of Computer Vision*, 74(1):33–50, 2007. 3

[40] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, Aug. 2004. 7

[41] C. Sanderson. Armadillo: An open source C++ linear algebra library for fast prototyping and computationally intensive experiments. Technical report, NICTA, 2010. 8

[42] T. Schoenemann and D. Cremers. A coding-cost framework for super-resolution motion layer decomposition. *Image Processing, IEEE Transactions on*, 21(3):1097–1110, March 2012. 3

[43] D. Sun, C. Liu, and H. Pfister. Local layering for joint motion estimation and occlusion detection. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1098–1105, June 2014. 1

[44] D. Sun, S. Roth, and M. Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision*, 106(2):115–137, 2014. 2, 3

[45] D. Sun, E. Sudderth, and M. Black. Layered segmentation and optical flow estimation over time. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1768–1775, June 2012. 2, 3

[46] D. Sun, J. Wulff, E. Sudderth, H. Pfister, and M. Black. A fully-connected layered model of foreground and background flow. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2451–2458, June 2013. 1, 2, 3

[47] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. Technical Report UCB/EECS-2010-104, EECS Department, University of California, Berkeley, Jul 2010. 3

[48] M. Tao, J. Bai, P. Kohli, and S. Paris. Simpleflow: A non-iterative, sublinear optical flow algorithm. *Computer Graphics Forum*, 31(2pt1):345–353, 2012. 3

[49] J. Wang and E. Adelson. Representing moving images with layers. *Image Processing, IEEE Transactions on*, 3(5):625–638, Sep 1994. 3

[50] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. DeepFlow: Large displacement optical flow with deep matching. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1385–1392, Dec 2013. 1, 2, 3, 4

[51] M. Werlberger, T. Pock, and H. Bischof. Motion estimation with non-local total variation regularization. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2464–2471, June 2010. 3

[52] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof. Anisotropic Huber-L1 optical flow. In *BMVC*, London, UK, September 2009. 1, 2, 3

[53] L. Xu, J. Jia, and Y. Matsushita. Motion detail preserving optical flow estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(9):1744–1757, Sept 2012. 1, 2, 4

[54] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-L1 optical flow. In F. Hamprecht, C. Schnrr, and B. Jhne, editors, *Pattern Recognition*, volume 4713 of *Lecture Notes in Computer Science*, pages 214–223. Springer Berlin Heidelberg, 2007. 2, 8

[55] K. Zuiderveld. Contrast limited adaptive histogram equalization. In P. S. Heckbert, editor, *Graphics Gems IV*, pages 474–485. Academic Press Professional, Inc., San Diego, CA, USA, 1994. 4